LightMoE: Reducing Mixture-of-Experts Redundancy through Expert Replacing

Jiawei Hao[1], **Zhiwei Hao**[2*], **Jianyuan Guo**[2], **Li Shen**[3], **Yong Luo**[4], **Han Hu**[1*], **Dan Zeng**[5]

[1]School of information and Electronics, Beijing Institute of Technology.
[2]Department of Computer Science, City University of Hong Kong.
[3]School of Cyber Science and Technology, Shenzhen Campus of Sun Yat-sen University.
[4]School of Computer Science, Wuhan University.
[5]School of Communication and Information Engineering, Shanghai University.

{haojiawei7, hhu}@bit.edu.cn, {zhiwei.hao, jianyguo}@cityu.edu.hk,
shenli6@mail.sysu.edu.cn, luoyong@whu.edu.cn, dzeng@shu.edu.cn

Abstract

Mixture-of-Experts (MoE) based Large Language Models (LLMs) have demonstrated impressive performance and computational efficiency. However, their deployment is often constrained by substantial memory demands, primarily due to the need to load numerous expert modules. While existing expert compression techniques like pruning or merging attempt to mitigate this, they often suffer from irreversible knowledge loss or high training overhead. In this paper, we propose a novel expert compression paradigm termed expert replacing, which replaces redundant experts with parameter-efficient modules and recovers their capabilities with low training costs. We find that even a straightforward baseline of this paradigm yields promising performance. Building on this foundation, we introduce LightMoE, a framework that enhances the paradigm by introducing adaptive expert selection, hierarchical expert construction, and an annealed recovery strategy. Experimental results show that LightMoE matches the performance of LoRA fine-tuning at a 30% compression ratio. Even under a more aggressive 50% compression rate, it outperforms existing methods and achieves average performance improvements of 5.6% across five diverse tasks. These findings demonstrate that LightMoE strikes a superior balance among memory efficiency, training efficiency, and model performance.

1 Introduction

LLMs leveraging Sparse Mixture-of-Experts (MoE) architectures, exemplified by models such as DeepSeek-MoE [1, 2] and OLMoE [3], have recently received significant attention. Their design offers excellent performance and notable efficiency in both training and inference processes. However, a primary challenge associated with these powerful models is their substantial memory footprint. Loading numerous expert modules demands considerable memory resources, which constrains their practical applicability and impedes widespread deployment in real-world scenarios. While expert offloading techniques [4, 5] mitigate GPU memory limits, they introduce prohibitive inference latency due to the frequent transfer of weights from CPU memory or disk. Consequently, direct parameter compression has become a critical research frontier.

Current MoE compression methods largely follow two paradigms: expert pruning [6, 7] and expert merging [8, 9, 10]. Expert pruning methods, such as MoE-Pruner, which prunes weights based on

*Corresponding author.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

activation frequencies and router importance scores, aim to remove less critical experts. However, a significant drawback of pruning is the irreversible loss of pruned expert knowledge, which results in substantial performance degradation. On the other hand, expert merging seeks to combine multiple experts into a single, more compact representation. However, merging experts inherently diminishes the model's representational diversity, and determining an optimal merging strategy remains a significant challenge in the field. Despite these challenges, recent findings [11, 6] reveal that current MoEs contain significant parameter redundancy. Some studies [12] also indicate a high concentration of active experts in fine-grained MoE models when applied to specific tasks.

Based on these challenges and observations, we explore a straightforward solution: *Can we simply replace less critical experts with parameter-efficient modules and subsequently recover their capabilities with low-cost training*? This idea raises three fundamental questions: (1) *expert selection*: How to select the less important experts for replacement? (2) *module construction*: How to design and initialize the parameter-efficient modules? (3) *efficient recovery*: How to restore model performance with minimal training overhead? A simple way to materialize this idea is to replace these less frequently activated experts with Low-Rank Adaptation (LoRA) [13], and subsequently fine-tune the modified model. As illustrated in Figure 1, this directly replacing strategy achieves performance comparable to the existing method MC-SMoE [8]. However, both approaches suffer from significant performance degradation, particularly at higher compression ratios. This suggests that even experts deemed "inactive" or less critical for a specific task may still harbor fundamental abilities and knowledge crucial for overall model capabilities. Attempting to restore this lost knowledge and alleviate performance degradation solely through fine-tuning proves to be a difficult endeavor.

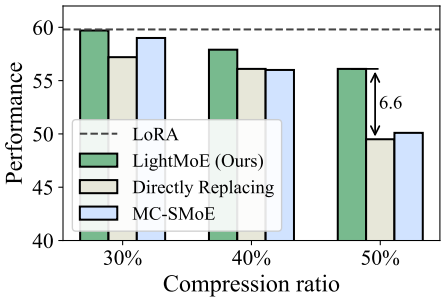


Figure 1: Performance comparison across different compression ratios on the Math task. While the directly replacing strategy performs comparably to MC-SMoE, both suffer from significant performance degradation, particularly at higher ratios

In this work, we propose LightMoE, a framework for compressing redundant experts in MoE models via expert replacing. The framework comprises three key stages: selecting less important experts through adaptive thresholding, constructing hierarchical experts, and progressively replacing the original experts through annealing. First, LightMoE calculates the relative importance of experts within and across layers to establish an adaptive threshold. This threshold serves as a dynamic criterion to select less critical experts. Next, the selected experts are grouped and replaced with a smaller number of shared bases, each equipped with task-specific low-rank adaptation parameters to preserve specialization. Finally, during the fine-tuning phase, the original experts are gradually annealed into their corresponding shared bases, allowing the model to adapt smoothly to the compressed structure. Experimental results across five diverse tasks show that LightMoE achieves performance comparable to LoRA fine-tuning at a 30% compression ratio. Even under a more aggressive 50% compression rate with the same training budget, it delivers average performance improvements of 5.6% and 3.8% over existing methods and the directly replacing baseline, respectively.

2 Related Works

Mixture-of-Experts LLMs. In recent years, autoregressive language models have demonstrated remarkable capabilities across a wide range of language understanding and generation tasks[14]. In contrast to traditional dense Large Language Models (LLMs) such as the LLaMA series [15, 16], Mixture of Experts (MoE) architectures offer a compelling approach to expanding model capacity while reducing training and inference costs. The majority of existing MoE [17, 18, 19] implementations employ coarse-grained architectures with relatively few experts. For instance, the Mixtral series [20] activate only 2 out of 8 available experts during computation. This limitation necessitates that each expert must handle diverse patterns across multiple domains simultaneously.

More recently, fine-grained expert segmentation [1, 21, 22] has gained significant attention, as it enables a greater variety of expert combinations and demonstrates superior performance. In the OLMoE [3], there are as many as 64 experts, with 8 active experts. However, these fine-grained MoE

architectures inherit substantial memory footprints from storing all expert weights. Interestingly, recent studies [12] reveal that this fine-grained division ensures a high degree of specialization among the experts. We argue that this distinct specialization presents opportunities for task-specific model compression, enabling more efficient deployment for downstream applications.

MoE Compression Methods. Given that experts constitute a significant portion of memory requirements, prior work on MoE model efficiency can be categorized into these principal approaches: expert offloading [4, 5], expert compression [6, 7, 8, 9], and more general compression strategies such as quantization [23, 24] and knowledge distillation [25, 26, 27].

In this work, we focus on the expert compression technique, which can be broadly categorized into expert pruning [6, 7] and expert merging [8, 9, 10]. Expert pruning methods focus on pruning dense matrices and removing redundant experts. For instance, MoE-Pruner [28] prunes weights based on activation frequencies, while NAEE [6] searches for expert combinations that minimize output deviation. However, these approaches often lead to substantial performance degradation due to the permanent loss of expert knowledge. Furthermore, search-based methods like NAEE are ill-suited for fine-grained MoE architectures due to combinatorial explosion. In the case of OLMoE with 64 experts, a 50% reduction results in an overwhelming search space of $C(64, 32) \approx 1.8 \times 10^{18}$ combinations per layer. Expert merging methods aim to consolidate multiple experts into fewer ones. For example, MC-SMoE [8] clusters experts according to their routing statistics and subsequently merges each cluster into a single representative expert. However, this approach inherently diminishes the model's representational diversity, and identifying an optimal merging strategy is non-trivial. Furthermore, while MC-SMoE employs progressive low-rank decomposition during retraining for further expert compression, it introduces substantial training overhead. This is because the decomposition process requires computing gradients from the original experts. Recent efforts have also explored dynamic inference optimization. Specifically, He et al. [29] propose replacing auxiliary activated experts with compressed modules to reduce active parameters. While this effectively lowers inference computational costs, it does not address the substantial memory footprint stemming from the total parameters, as the non-active experts still require storage.

Distinct from the aforementioned approaches, we propose a novel expert compression paradigm termed *expert replacing*. We further propose optimizations across three key dimensions: expert selection, module construction, and efficient recovery. From a broader perspective, expert merging can be viewed as a special case of our expert replacing paradigm. Expert merging does not reduce the number of experts indexed by the router, but instead maps multiple indices to the same parameters. In essence, this is equivalent to replacing multiple original experts with a single shared expert.

Low-Rank Adaptation. The immense complexity and computational demands of LLMs with billions of parameters pose substantial challenges for their adaptation to specific downstream tasks. Parameter-efficient fine-tuning (PEFT) [30, 31, 32] aims to minimize the fine-tuning parameters while achieving performance comparable to full fine-tuning. Typical PEFT methods include adapter [33, 34, 35, 36], soft prompt [37, 38, 39], and low-rank adaptation [13, 40, 41, 42, 43]. LoRA [13] decomposes the original weight matrices into low-rank components. ScaleNet [44] combines shared weights with layer-specific adapters for parameter-efficient model expansion. Inspired by this, we propose replacing multiple experts with parameter-efficient modules, which effectively reduce the number of expert parameters while preserving the diversity of the experts. Moreover, our approach incurs significantly lower training overhead compared to full fine-tuning the entire compressed model.

Hierarchical PEFT and continual learning. TreeLoRA [45] organizes layer-wise LoRA adapters in a task-level hierarchy based on gradient similarity. GainLoRA [46] gates task-specific LoRA branches, while InfLoRA [47] constrains low-rank updates to interference-free subspaces. These methods support continual adaptation while retaining the pretrained backbone. LightMoE constructs an expert-level hierarchy within each MoE layer and permanently replaces selected full-rank experts, reducing total model parameters and deployment memory.

3 Method

In a MoE layer, a router module selects one or more experts to process each input based on its routing decision, and the outputs of the selected experts are aggregated to form the output. To enable redundant expert compression, the proposed LightMoE framework first measures the importance of each expert and selects less critical candidates for compression (Section 3.1). These candidate

experts are then grouped, and each group is assigned a shared base equipped with low-rank adaptation terms to retain the specialization of the original experts (Section 3.2). Finally, during training, the original candidates are gradually replaced with their corresponding shared bases through an annealed replacement strategy (Section 3.3). Figure 2 provides an overview of the proposed framework.

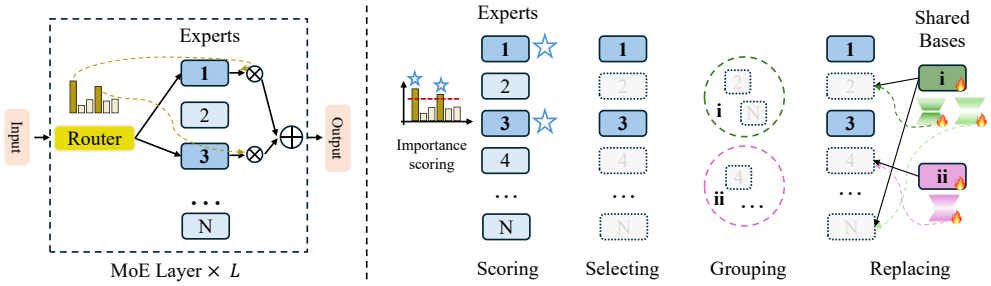


Figure 2: Overview of the proposed LightMoE framework. (*Left*) A standard MoE layer. (*Right*) The LightMoE workflow, comprising three key steps: (1) scoring experts and selecting those with lower scores as compression candidates, (2) grouping the selected candidates, and (3) replacing each group with a shared base augmented with lightweight, expert-specific adaptation parameters.

3.1 Adaptive Expert Selection

In MoE models, redundancy manifests in two dimensions: intra-layer imbalance, where some experts contribute significantly less than others, and inter-layer variability, where different layers exhibit varying degrees of impact on model performance. Building on these observations, we define expert importance based on activation frequency and propose an adaptive strategy to select redundant experts. Specifically, this process involves two steps: computing the importance score for each expert, and identifying less important experts in each layer using an adaptive threshold that incorporates both expert-level and layer-level importance.

Importance scoring. A typical MoE layer comprises multiple feed-forward network experts and a routing mechanism that dynamically selects the active experts for each input token. The output of the MoE layer is computed as:

$$y = \sum_{i=1}^{N} G(x)_i E_i(x), \tag{1}$$

where x and y denote the input and output of the MoE layer, respectively, N is the total number of experts, E_i represents the i-th expert, and $G(x)_i$ is the gating value assigned to expert i, indicating its relevance to the input token x.

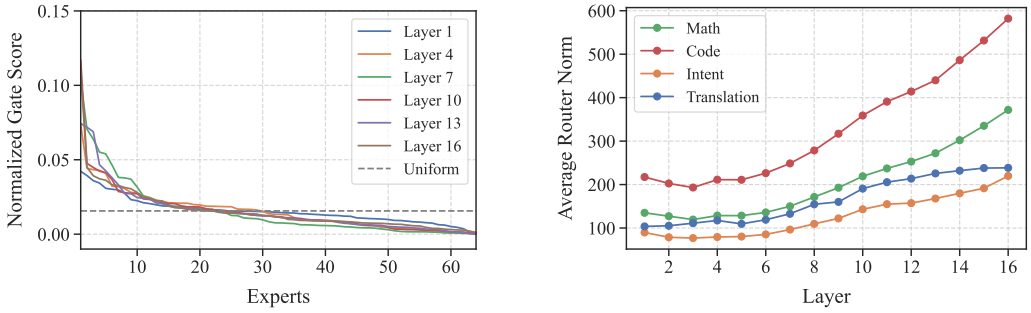
To quantify the importance of each expert, we sample a subset of the training data and aggregate gate values across all tokens. This provides a data-driven estimate of each expert's contribution. To normalize these contributions and enable comparison across experts, we define the normalized gate score for expert i as:

$$G_i = \frac{\sum_{x \in \mathcal{X}} G(x)_i}{\sum_{i=1}^{N} \sum_{x \in \mathcal{X}} G(x)_i}, \tag{2}$$

where $\mathcal{X}$ denotes the evaluation subset. This score reflects the relative importance of each expert based on its cumulative gate activation across the sampled data.

In our preliminary experiment, we used samples from the training set, totaling 2^{17} tokens, to evaluate expert importance. The MoE model used is OLMoE-1B-7B-SFT [3]. Figure 3a presents the sorted importance scores of experts in the selected layers. The non-uniform distribution of scores confirms the existence of differences in expert importance, supporting the compression of less important experts in our proposed method.

Adaptive thresholding. The importance scoring results reveal that experts within the same layer exhibit varying degrees of significance. This suggests that less important experts can be compressed without substantial performance degradation. A direct approach might apply a fixed compression ratio across all experts. However, such a coarse strategy risks unnecessary performance loss.



(a) Sorted expert-wise normalized gate scores. Experts exhibit varying importance across layers, highlighting *intra-layer* differences in specialization.

(b) Average output norms of router per layer. The increasing norm with depth suggests growing *inter-layer* importance in deeper layers.

Figure 3: Analysis of expert importance in OLMoE-1B-7B-SFT.

Motivated by the observed variation in importance among experts within a layer, we hypothesize that similar differences exist across layers in MoE models. To test this, we evaluate the average output norms of router at each layer using a subset of training samples. The underlying intuition is that more important layers tend to produce stronger activations, which reflected in larger output norms. Similar ideas have also been adopted in prior studies [48, 49]. Figure 3b presents the results, which show that as the model depth increases, the average router norm also rises, suggesting that deeper layers play a more critical role. Building on this insight, we propose a compression strategy that varies across layers, where shallower layers are assigned higher compression ratios, while deeper layers are preserved more conservatively. Specifically, based on the previously defined normalized gating scores, we sort experts in ascending order and select the top subset whose cumulative score just exceeds a layer-specific compression threshold. This subset is deemed less important and subject to compression. Formally, given a base threshold $\hat{p}$, we define the threshold for the j-th layer as:

$$\hat{p}_j = \text{clip}\left(\hat{p} \cdot \mathrm{e}^{-\alpha(\text{norm}_j - 1)}, p_{\min}, p_{\max}\right), \tag{3}$$

where norm_j is the ratio of the norm of layer j to the average norm across all layers. Thus, if the norm of layer j exceeds the mean, the exponential term drives $\hat{p}_j$ below the base threshold $\hat{p}$. α is an exponential decay coefficient that modulates the strength of the adjustment; we set $\alpha = 0.3$ in all experiments. $\text{clip}(p, p_{\min}, p_{\max})$ truncates the adjusted threshold to lie within $[p_{\min}, p_{\max}]$ for numerical stability. In our experiments we set $p_{\min} = 0.8\hat{p}$ and $p_{\max} = 1.2\hat{p}$. This adaptive thresholding mechanism allows the proposed method to account for both intra-layer and inter-layer variations in expert importance, resulting in a more effective compression approach.

3.2 Hierarchical Expert Construction

After selecting less important experts in each layer, we aim to compress them to reduce memory consumption. However, directly removing or merging them may lead to a loss of expert-specific knowledge. To solve this, we propose a hierarchical expert representation that decomposes each expert into a shared base and a expert-specific low-rank adapter. This design allows the model to capture common patterns via the shared base while preserving specific capabilities through efficient parameterization. Specifically, let N' denote the number of experts to be compressed in a given layer, where each expert is parameterized by a weight matrix $W_1, W_2, \ldots, W_{N'} \in \mathbb{R}^{n \times m}$. We first construct the shared base W_{share} as the weighted average of these experts, utilizing their normalized gate scores G_i as weights[2]:

$$W_{\text{share}} = \frac{\sum_{i=1}^{N'} G_i W_i}{\sum_{i=1}^{N'} G_i}. \tag{4}$$

Subsequently, each original expert $W_{n'}$ is reconstructed hierarchically as $W_{\text{share}} + B_{n'} A_{n'}$, where $B_{n'} \in \mathbb{R}^{n \times r}$ and $A_{n'} \in \mathbb{R}^{r \times m}$ represent the expert-specific low-rank adaptation terms, with $r \ll \min(n, m)$.

[2]For simplicity, we represent each expert using a single weight matrix. In practice, all parameter matrices in each expert are processed in the same manner.

While a single shared base can serve as a common foundation, it may not adequately capture the diverse knowledge embedded across multiple less important experts. To address this, we extend the framework to support multiple shared bases. Suppose we choose to use M shared bases, where $M < N'$. The N' experts are partitioned into M groups, each represented by a shared base. Following the approach in [8], we select the top M experts with the highest normalized gate scores G_i as dominant experts. Each remaining expert is then assigned to the dominant expert with which it shares the highest semantic similarity. Specifically, given b calibration tokens, we collect the router logits in $\mathbb{R}^{b\times N}$ and use the column corresponding to each expert as its routing vector. We then compute the cosine similarity between the routing vector of each non-dominant expert and those of the dominant experts, and assign it to the most similar dominant expert. Once grouped, each set of experts is compressed into a shared base following the weighted average procedure described above.

Expert-Level Compression Ratio. Under the proposed scheme, multiple full-rank experts are represented using a smaller number of full-rank shared bases, each augmented with several low-rank adaptation terms. The resulting expert compression ratio ρ is given by:

$$\rho = 1 - \frac{(N - N' + M)nm + N'r(n+m)}{N \cdot nm}. \tag{5}$$

Since $M < N'$ and $r \ll \min(n, m)$, this strategy leads to a substantial reduction in parameter count while preserving model capacity and diversity.

3.3 Annealed Expert Replacement

Directly replacing the original experts with shared bases and low-rank adaptation parameters can lead to significant performance degradation due to the abrupt change in the model parameter space. To mitigate this, we introduce an annealed expert replacement strategy, which gradually transitions each expert from its original form to its compressed representation during fine-tuning. This smooth transition helps preserve performance by maintaining continuity in the optimization trajectory. A detailed empirical analysis of the training dynamics is provided in appendices (Appendix C).

Concretely, consider a less important expert $W_{n'}$ selected for compression. During fine-tuning, its effective parameter matrix $W^*_{n'}$ is computed as a weighted combination of three components: the original expert $W_{n'}$, the shared base W_{share}, and the expert-specific low-rank adaptation term $B_{n'}A_{n'}$. The combined parameter is given by:

$$W^*_{n'} = \beta W_{n'} + (1-\beta)W_{\text{share}} + B_{n'}A_{n'}, \tag{6}$$

where $\beta \in [0, 1]$ is an annealing factor that is gradually decays from 1 to 0 over the course of fine-tuning. At the beginning of training, $\beta = 1$, so the model behaves identically to the original MoE. As training progresses and β decays, the model incrementally shifts towards using the compressed representation. This progressive interpolation ensures a smooth adaptation to the shared and low-rank parameter space. At the end of fine-tuning, $\beta = 0$, and the original expert parameters $W_{n'}$ are no longer used. They can thus be safely removed during inference, reducing the model size and achieving the desired compression. We utilize a simple yet effective decay strategy for β:

$$\beta = \max\left(1 - \frac{t}{\epsilon T}, 0\right), \tag{7}$$

where T is the total steps of training and t is the current step. The parameter $\epsilon \in [0, 1]$ is the end ratio that controls when β completely decays to 0. For instance, when $\epsilon = 0.4$, β reaches zero at $t = 0.4T$. When $\epsilon = 0$, β is set to zero at the beginning of training, which means directly replacing the original experts with the parameter-efficient modules without any annealing period.

4 Experiments

4.1 Experiment Setup

Datasets and evaluation. Following ESFT [12], we evaluate our LightMoE method in two LLM customization scenarios: (1) preserving **specific ability in a domain** in which the model already demonstrates reasonable performance, and (2) adapting the model to a **narrow but unfamiliar specialized task** while achieving compression. For preservation, we target Math, Coding, and

Commonsense Reasoning domains. For the Math domain, we train on MetaMathQA [50] and evaluate using GSM8K [51]. For the Code domain, we use CodeFeedback [52] for training and HumanEval [53] for evaluation. Evaluation protocols are consistent with those used in existing works [3, 54, 55]. For the Commonsense Reasoning domain, we utilize the Cleaned Alpaca Dataset [56] for training. Following standard evaluation protocols [56, 41], we report the averaged accuracy on eight representative commonsense reasoning tasks, such as ARC [57], BoolQ [58], PIQA [59], and WinoGrande [60]. For adaptation, we focus on intent recognition and low-resource translation. The intent recognition task comes from the BDCI-21 Smart HCI NLU Challenge.[3] The low-resource translation task utilizes the ChrEn dataset [61], which requires translating Cherokee into English. In line with ESFT [12], during evaluation, we compute the exact match rate between model predictions and reference answers on intent recognition task. For the translation task, we use GPT-4 to assign a score between 0 and 10 according to output quality relative to the reference.[4] For expert importance scoring and adaptive threshold selection, we sample a calibration subset from the training set of each target task, with a default budget of 2^{17} tokens unless otherwise specified.

Backbone model. We use OLMoE-1B-7B-SFT [3] with 6.9B total and 1.3B active parameters as the pretrained model. The model includes a fine-grained set of 64 experts for each MoE layer.

Baselines. We compare with representative expert-compression methods, including MC-SMoE [8], HC-SMoE [10], and MoBE [62]. For MC-SMoE, we report both its original setting and a LoRA-based variant for computationally fair comparison. We also evaluate two direct expert-replacement baselines, "Replace (w/o shared)" and "Replace (w. shared)", and include full fine-tuning and LoRA fine-tuning on the original model as uncompressed references. Detailed baseline configurations are provided in Appendix E.

Training details. All methods use the AdamW optimizer [63] with a batch size of 32, and a learning rate of 1e-4. For our method, we adopt a low-rank of 16 for all expert parameters and set the group size to 3. The low-rank matrices are initialized following the standard strategy [13]. For a fair comparison, we adjust baseline ranks to match the number of trainable parameters. For model preservation, we adopt a training step setting of 2000. For model adaptation, training is limited to 500 steps due to smaller datasets. The optimal end ratio ϵ is obtained by grid search from $\{0.1, 0.2, 0.3, 0.4, 0.5\}$. Unless otherwise specified, experiments are conducted on NVIDIA RTX 4080 SUPER 32GB GPUs and AMD EPYC 9654 CPUs.

4.2 Main Results

Overall performance. Table 1 reports the main results on five tasks at three compression ratios. Our method consistently achieves the best or second-best performance across various compression ratios. Remarkably, at a 30% compression ratio, our method performs comparably to LoRA, even surpassing it on some tasks. This suggests that replacing extremely unimportant experts with parameter-efficient modules even benefits downstream task performance. At a 50% compression ratio with the same training budget, our method significantly outperforms existing methods, achieving average performance improvements of 5.6% and 3.8% over existing methods and the directly replacing baseline, respectively. Besides, even with over three times the trainable parameters, MC-SMoE* still lags behind our method by 2.8%. This demonstrates that our method strikes a superior balance among compression efficiency, training efficiency, and model performance.

For model preservation tasks, our method maintains robust performance even under high compression ratios. Notably, on the Math task, our approach preserves 94% of LoRA's performance while reducing the model parameters by 50%. This demonstrates that our method can effectively preserve the existing ability of the model. For model adaptation tasks, although performance fluctuates across different specialized tasks and compression ratios, our method outperforms alternative approaches. This indicates that our technique successfully enables model adaptation to an unfamiliar downstream task while simultaneously achieving model compression.

Furthermore, the "Replace (w/o shared)" method achieves performance comparable to MC-SMoE at a 30% ratio and surpasses it at 40%. Notably, even compared to MC-SMoE*, it maintains comparable performance at both 40% and 50% ratios. This shows that directly replacing less important experts is a strong baseline for compressing MoE models.

[3] `https://www.datafountain.cn/competitions/511`

[4] The exact version used is `gpt-4-1106-preview`. The scores are normalized to match the scale of other reported metrics.

Table 1: Main performance comparison across methods and tasks at different compression ratios. '*' indicates the full method as described in the original paper. "# Params" is the number of trainable parameters. Best results are shown in **bold** and second-best results are underlined. LightMoE consistently achieves good performance among all tasks under different compression settings.

Ratio	Method	# Params	Math	Code	Common.	Specialized Tasks		Avg.
			GSM8K	HumanEval	8 Sub-Tasks	Intent	Translation	
0%	Original	-	49.3	55.0	60.6	9.8	13.6	37.7
0%	Full FT	6.92B	61.2	49.7	54.2	79.2	37.1	56.3
0%	LoRA	0.45B	59.8	54.2	61.4	73.0	29.2	55.5
30%	MoBE	0.45B	52.9	41.3	**58.3**	69.4	30.3	50.4
	HC-SMoE	0.45B	40.6	41.7	55.5	64.0	**32.4**	46.8
	MC-SMoE	0.45B	59.0	51.7	55.1	70.2	30.9	53.4
	Replace (w/o shared)	0.45B	57.2	53.8	55.6	69.4	27.0	52.6
	Replace (w. shared)	0.45B	57.8	52.0	56.7	71.4	23.6	52.3
	LightMoE	0.45B	**59.7**	**55.7**	56.3	**74.6**	30.4	**55.3**
40%	MC-SMoE*	1.65B	57.0	50.6	40.3	**74.5**	28.4	50.2
	MoBE	0.45B	37.2	28.6	**54.0**	65.4	29.9	43.0
	HC-SMoE	0.45B	31.2	33.5	51.5	58.8	16.9	38.4
	MC-SMoE	0.45B	56.0	48.4	50.4	70.8	10.8	47.3
	Replace (w/o shared)	0.45B	56.1	**51.3**	52.5	65.8	24.7	50.1
	Replace (w. shared)	0.45B	57.7	50.1	53.9	59.0	25.9	49.3
	LightMoE	0.45B	**57.9**	**51.3**	53.2	72.6	**30.2**	**53.0**
50%	MC-SMoE*	1.65B	50.8	43.8	40.0	73.4	18.5	45.3
	MoBE	0.45B	18.7	9.7	46.4	53.0	**21.6**	29.9
	HC-SMoE	0.45B	26.2	26.5	47.8	58.0	11.6	34.0
	MC-SMoE	0.45B	50.1	43.1	44.8	67.2	7.2	42.5
	Replace (w/o shared)	0.45B	49.5	46.3	47.1	65.4	13.0	44.3
	Replace (w. shared)	0.45B	51.4	44.5	**49.3**	56.2	6.7	41.6
	LightMoE	0.45B	**56.1**	**48.4**	48.1	**74.2**	13.5	**48.1**

Generalization to other tasks. To examine the general ability of LightMoE, we evaluate OLMoE at a 50% compression ratio by training on one task and testing on other tasks. Specifically, we train the model on Math or Code data and evaluate it on Code, Math, and Commonsense Reasoning. The results are shown in Table 2.

Table 2: Cross-task transfer results on OLMoE at a 50% compression ratio.

Method	Math→Code	Math→Common.	Code→Math	Code→Common.	Avg.
MoBE	0.0	42.9	4.2	45.7	23.2
HC-SMoE	4.7	**45.7**	5.0	**47.3**	25.7
MC-SMoE	6.3	42.3	8.1	42.1	24.7
LightMoE	**37.7**	43.1	**13.8**	43.1	**34.4**

Overall, LightMoE achieves the best average score across the four transfer settings and is best or second-best in most cases. The advantage is particularly clear on Math→Code, where the baselines collapse to 0.0, 4.7, and 6.3, while LightMoE reaches 37.7. These results suggest that LightMoE maintains strong generalization when the training and evaluation tasks differ.

Efficiency analysis. We further summarize the training and inference efficiency of LightMoE, with full details provided in Appendix B, including Tables 14 and 15. For training, LightMoE achieves a favorable cost–performance trade-off. Compared with MC-SMoE* (8x4), LightMoE reduces peak training memory from 77.83 GB to 56.61 / 56.60 GB, mean step time from 14.10 / 14.03s to 5.89 / 6.03s, and end-to-end cost from 7.88 / 7.84 to 3.33 / 3.41 USD at 40% / 50% compression. Compared with MoBE under the 10000-epoch setting, LightMoE reduces end-to-end cost from 7.12 / 6.08 / 5.11 to 3.21 / 3.33 / 3.41 USD at 30% / 40% / 50% compression.

For inference, LightMoE substantially reduces the model footprint while preserving comparable latency. At a 50% compression ratio, the total parameter count decreases from 6.92B to 3.47B, and GPU memory usage drops from 12.89 GB to 6.63 GB. Meanwhile, inference latency remains comparable to the original model, changing from 8.09 ms/token to 8.01 ms/token. Overall, LightMoE achieves a strong balance among memory efficiency, training efficiency, and model performance.

4.3 Ablation Study

Ablation of main components. We assess the effectiveness of the three proposed designs: expert selection, hierarchical expert construction, and annealed expert replacement. For expert selection, we compare our Adaptive strategy with a Uniform baseline that uses the constant threshold across layers. As shown in Table 3, at lower compression ratios, adaptive selection improves performance over uniform selection, while the gain from the hierarchical structure is relatively small. At higher compression ratios, the hierarchical structure and annealed replacement become increasingly important. At 50% compression, the hierarchical module improves the score from 50.0 to 53.3 over LoRA replacement, and annealed replacement further raises it to 56.1. These results show that the three designs provide complementary gains, especially in high-compression settings.

Impact of decay ratio. Figure 4 illustrates how the end ratio hyperparameter affects model performance. Directly replacing (blue points) is consistently sub-optimal, which shows the effectiveness of our annealed expert replacement strategy. Furthermore, we find that that lower end ratios, specifically 0.1 or 0.2, can often yield the best results.

Impact of adaptation rank. To investigate the impact of adaptation rank, we conduct an ablation study across various ranks using the Math task, with results presented in Figure 5. Overall, we observe that lower ranks generally yield superior performance. Specifically, at a 30% compression ratio, performance peaks at rank 16, with a slight drop at higher ranks. Similarly, at 40% and 50% compression ratios, the best performance is achieved at relatively low ranks, while increasing the rank leads to performance degradation. These findings suggest that smaller ranks are generally more effective at higher compression levels.

Table 3: Component ablation on OLMoE for the Math task. DR denotes direct replacement, and AR denotes annealed replacement.

Selection Method	Replacement Module	Training Strategy	30%	40%	50%
Uniform	LoRA	DR	57.2	56.1	49.5
Adaptive	LoRA	DR	58.5	56.7	50.0
Adaptive	Hierarchical	DR	58.4	57.2	53.3
Adaptive	**Hierarchical**	**AR**	**59.7**	**57.9**	**56.1**

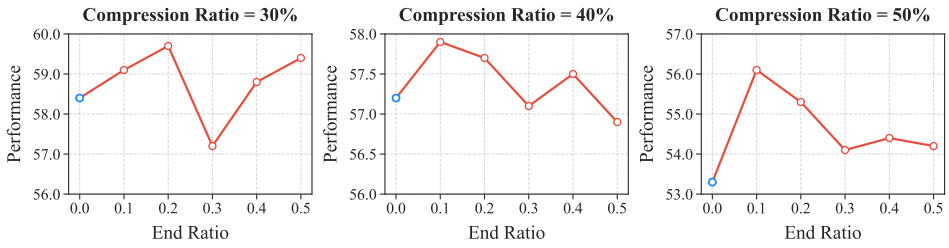


Figure 4: Comparison of different end ratios for LightMoE on the Math task. Directly replacing (blue points) is consistently sub-optimal, which shows the effectiveness of our annealed expert replacement strategy (red points).

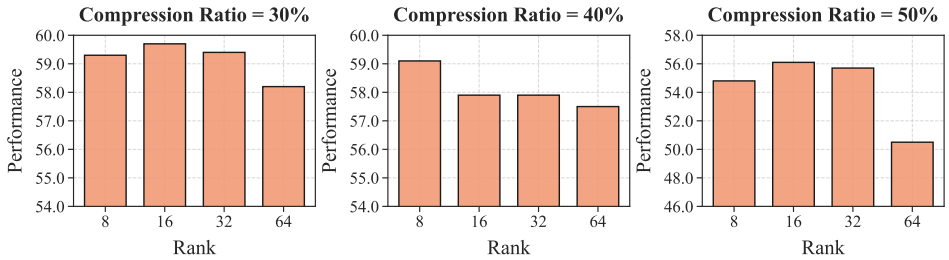


Figure 5: Comparison of different ranks at different compression ratios on the Math task.

5 Conclusion

In this paper, we introduce a novel expert compression paradigm termed expert replacing. Our empirical findings demonstrate that even a simple baseline of this paradigm yields promising results. Building on this foundation, we propose LightMoE, a framework that enhances the paradigm by

introducing adaptive expert selection, hierarchical expert construction, and an annealed recovery strategy. Experimental results across diverse tasks show that our method reduces the memory footprint of MoE models with minimal training overhead while preserving performance. Beyond providing a practical MoE compression solution, this work highlights expert replacing as a promising direction. Future work could explore advanced initialization methods and adaptive rank allocation strategies to further unlock the potential of this paradigm.

Limitation. Our method relies on score statistics from a small validation set, which may not fully reflect real-world expert usage patterns.Additionally, our experiments are limited to language tasks.

Broader impacts. By reducing MoE memory requirements, LightMoE can broaden LLM access and support more efficient AI deployment, but performance should be monitored across tasks.

Acknowledgment

This work was supported in part by the Joint Funds of the NSFC under Grants U2336211 and U23A20318, the Beijing Natural Science Foundation under Grant No. L2607010, the NSFC under Grant No. 62576364, the Guangdong Basic and Applied Basic Research Foundation under Grant No. 2026B1515020071, the Hong Kong RGC under Grant No. 21218026, and the City University of Hong Kong under Grants No. 9382010, 9048367 and 7020171.

References

[1] 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. Deepseekmoe: Towards ultimate expert specialization in mixture-of-experts language models. In *ACL*, pages 1280–1297, 2024.

[2] Aixin Liu, Bei Feng, Bin Wang, Bingxuan Wang, Bo Liu, Chenggang Zhao, Chengqi Dengr, Chong Ruan, Damai Dai, Daya Guo, et al. Deepseek-v2: A strong, economical, and efficient mixture-of-experts language model. *arXiv preprint arXiv:2405.04434*, 2024.

[3] Niklas Muennighoff, Luca Soldaini, Dirk Groeneveld, Kyle Lo, Jacob Morrison, Sewon Min, Weijia Shi, Pete Walsh, Oyvind Tafjord, Nathan Lambert, et al. Olmoe: Open mixture-of-experts language models. In *International Conference on Learning Representations*, volume 2025, pages 62061–62121, 2025.

[4] Yechan Kim, Hwijoon Lim, and Dongsu Han. Scaling beyond the gpu memory limit for large mixture-of-experts model training. In *ICML*, 2024.

[5] Artyom Eliseev and Denis Mazur. Fast inference of mixture-of-experts language models with offloading. *arXiv preprint arXiv:2312.17238*, 2023.

[6] Xudong Lu, Qi Liu, Yuhui Xu, Aojun Zhou, Siyuan Huang, Bo Zhang, Junchi Yan, and Hongsheng Li. Not all experts are equal: Efficient expert pruning and skipping for mixture-of-experts large language models. In *ACL*, pages 6159–6172, 2024.

[7] Zeliang Zhang, Xiaodong Liu, Hao Cheng, Chenliang Xu, and Jianfeng Gao. Diversifying the expert knowledge for task-agnostic pruning in sparse mixture-of-experts. In *ACL (Findings)*, pages 86–102, 2025.

[8] Pingzhi Li, Zhenyu Zhang, Prateek Yadav, Yi-Lin Sung, Yu Cheng, Mohit Bansal, and Tianlong Chen. Merge, then compress: Demystify efficient smoe with hints from its routing policy. In *ICLR*, 2024.

[9] Enshu Liu, Junyi Zhu, Zinan Lin, Xuefei Ning, Matthew B. Blaschko, Shengen Yan, Guohao Dai, Huazhong Yang, and Yu Wang. Efficient expert pruning for sparse mixture-of-experts language models: Enhancing performance and reducing inference costs. *arXiv preprint arXiv:2407.00945*, 2024.

[10] I-Chun Chen, Hsu-Shen Liu, Wei-Fang Sun, Chen-Hao Chao, Yen-Chang Hsu, and Chun-Yi Lee. Retraining-free merging of sparse moe via hierarchical clustering. In *ICML*, 2025.

[11] Boan Liu, Liang Ding, Li Shen, Keqin Peng, Yu Cao, Dazhao Cheng, and Dacheng Tao. Diversifying the mixture-of-experts representation for language models with orthogonal optimizer. In *ECAI*, volume 392 of *Frontiers in Artificial Intelligence and Applications*, pages 2966–2973, 2024.

[12] Zihan Wang, Deli Chen, Damai Dai, Runxin Xu, Zhuoshu Li, and Yu Wu. Let the expert stick to his last: Expert-specialized fine-tuning for sparse architectural large language models. In *EMNLP*, pages 784–801, 2024.

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

[14] Zhongyu Xiao, Zhiwei Hao, Jianyuan Guo, Yong Luo, Jia Liu, Jie Xu, and Han Hu. Streaming-dllm: Accelerating diffusion llms via suffix pruning and dynamic decoding. *arXiv preprint arXiv:2601.17917*, 2026.

[15] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

[16] Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

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

[18] Damai Dai, Li Dong, Shuming Ma, Bo Zheng, Zhifang Sui, Baobao Chang, and Furu Wei. Stablemoe: Stable routing strategy for mixture of experts. *arXiv preprint arXiv:2204.08396*, 2022.

[19] Yikang Shen, Zhen Guo, Tianle Cai, and Zengyi Qin. Jetmoe: Reaching llama2 performance with 0.1 m dollars. *arXiv preprint arXiv:2404.07413*, 2024.

[20] Albert Q Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, et al. Mixtral of experts. *arXiv preprint arXiv:2401.04088*, 2024.

[21] An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, Guanting Dong, Haoran Wei, Huan Lin, Jialong Tang, Jialin Wang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Ma, Jin Xu, Jingren Zhou, Jinze Bai, Jinzheng He, Junyang Lin, Kai Dang, Keming Lu, Keqin Chen, Kexin Yang, Mei Li, Mingfeng Xue, Na Ni, Pei Zhang, Peng Wang, Ru Peng, Rui Men, Ruize Gao, Runji Lin, Shijie Wang, Shuai Bai, Sinan Tan, Tianhang Zhu, Tianhao Li, Tianyu Liu, Wenbin Ge, Xiaodong Deng, Xiaohuan Zhou, Xingzhang Ren, Xinyu Zhang, Xipin Wei, Xuancheng Ren, Yang Fan, Yang Yao, Yichang Zhang, Yu Wan, Yunfei Chu, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zhihao Fan. Qwen2 technical report. *arXiv preprint arXiv:2407.10671*, 2024.

[22] An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2024.

[23] Pingzhi Li, Xiaolong Jin, Yu Cheng, and Tianlong Chen. Examining post-training quantization for mixture-of-experts: A benchmark. *arXiv preprint arXiv:2406.08155*, 2024.

[24] Wei Huang, Yue Liao, Jianhui Liu, Ruifei He, Haoru Tan, Shiming Zhang, Hongsheng Li, Si Liu, and Xiaojuan Qi. Mc-moe: Mixture compressor for mixture-of-experts llms gains more. *arXiv preprint arXiv:2410.06270*, 2024.

[25] Haiyang Xu, Haoxiang Liu, Wei Gong, Xianjun Deng, and Hai Wang. Sparse mixture of experts language models excel in knowledge distillation. In *NLPCC*, pages 80–91. Springer, 2024.

[26] Gyeongman Kim, Gyouk Chu, and Eunho Yang. Every expert matters: Towards effective knowledge distillation for mixture-of-experts language models. *arXiv preprint arXiv:2502.12947*, 2025.

[27] Zhiwei Hao, Zhongyu Xiao, Yong Luo, Jianyuan Guo, Jing Wang, Li Shen, and Han Hu. Primkd: Primary modality guided multimodal fusion for rgb-d semantic segmentation. In *Proceedings of the 32nd ACM International Conference on Multimedia*, pages 1943–1951, 2024.

[28] Yanyue Xie, Zhi Zhang, Ding Zhou, Cong Xie, Ziang Song, Xin Liu, Yanzhi Wang, Xue Lin, and An Xu. Moe-pruner: Pruning mixture-of-experts large language model using the hints from its router. *arXiv preprint arXiv:2410.12013*, 2024.

[29] Yifei He, Yang Liu, Chen Liang, and Hany Hassan Awadalla. Efficiently editing mixture-of-experts models with compressed experts. In *EMNLP (Findings)*, pages 7227–7238, 2025.

[30] Lingling Xu, Haoran Xie, Si-Zhao Joe Qin, Xiaohui Tao, and Fu Lee Wang. Parameter-efficient fine-tuning methods for pretrained language models: A critical review and assessment. *arXiv preprint arXiv:2312.12148*, 2023.

[31] Zeyu Han, Chao Gao, Jinyang Liu, Jeff Zhang, and Sai Qian Zhang. Parameter-efficient fine-tuning for large models: A comprehensive survey. *Transactions on Machine Learning Research*, 2024.

[32] Zhiwei Hao, Jianyuan Guo, Li Shen, Yong Luo, Han Hu, and Yonggang Wen. Adem-vl: Adaptive and embedded fusion for efficient vision-language tuning. *International Journal of Computer Vision*, 133(8):5527–5543, 2025.

[33] Andreas Rücklé, Gregor Geigle, Max Glockner, Tilman Beck, Jonas Pfeiffer, Nils Reimers, and Iryna Gurevych. Adapterdrop: On the efficiency of adapters in transformers. In *EMNLP*, pages 7930–7946, 2021.

[34] Tao Lei, Junwen Bai, Siddhartha Brahma, Joshua Ainslie, Kenton Lee, Yanqi Zhou, Nan Du, Vincent Y. Zhao, Yuexin Wu, Bo Li, Yu Zhang, and Ming-Wei Chang. Conditional adapters: Parameter-efficient transfer learning with fast inference. In *NeurIPS*, 2023.

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

[36] Zhiwei Hao, Zhongyu Xiao, Jianyuan Guo, Li Shen, Yong Luo, Han Hu, and Dan Zeng. Mixprompt: Efficient mixed prompting for multimodal semantic segmentation. *Advances in Neural Information Processing Systems*, 38:34937–34964, 2026.

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

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

[39] Zhen Wang, Rameswar Panda, Leonid Karlinsky, Rogerio Feris, Huan Sun, and Yoon Kim. Multitask prompt tuning enables parameter-efficient transfer learning. *arXiv preprint arXiv:2303.02861*, 2023.

[40] Fanxu Meng, Zhaohui Wang, and Muhan Zhang. Pissa: Principal singular values and singular vectors adaptation of large language models. In *NeurIPS*, 2024.

[41] Shih-Yang Liu, Chien-Yi Wang, Hongxu Yin, Pavlo Molchanov, Yu-Chiang Frank Wang, Kwang-Ting Cheng, and Min-Hung Chen. Dora: Weight-decomposed low-rank adaptation. In *ICML*, 2024.

[42] Xun Wu, Shaohan Huang, and Furu Wei. Mixture of lora experts. *arXiv preprint arXiv:2404.13628*, 2024.

[43] Soufiane Hayou, Nikhil Ghosh, and Bin Yu. LoRA+: Efficient low rank adaptation of large models. In *ICML*, 2024.

[44] Zhiwei Hao, Jianyuan Guo, Li Shen, Kai Han, Yehui Tang, Han Hu, and Yunhe Wang. Scalenet: Scaling up pretrained neural networks with incremental parameters. *IEEE Transactions on Image Processing*, 34:7109–7122, 2025.

[45] Yu-Yang Qian, Yuan-Ze Xu, Zhen-Yu Zhang, Peng Zhao, and Zhi-Hua Zhou. Treelora: Efficient continual learning via layer-wise loras guided by a hierarchical gradient-similarity tree. *arXiv preprint arXiv:2506.10355*, 2025.

[46] Yan-Shuo Liang, Jia-Rui Chen, and Wu-Jun Li. Gated integration of low-rank adaptation for continual learning of large language models. *Advances in Neural Information Processing Systems*, 38:68659–68689, 2025.

[47] Yan-Shuo Liang and Wu-Jun Li. Inflora: Interference-free low-rank adaptation for continual learning. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 23638–23647. IEEE, 2024.

[48] Aditya Dawadikar, Pooja Shyamsundar, Rashmi Vishwanath Bhat, and Navrati Saxena. Activation-guided layer selection for lora. *Information*, 17(3):283, 2026.

[49] Zichen Song, Sitan Huang, Yuxin Wu, and Zhongfeng Kang. Layer importance and hallucination analysis in large language models via enhanced activation variance-sparsity. *arXiv preprint arXiv:2411.10069*, 2024.

[50] Longhui Yu, Weisen Jiang, Han Shi, Jincheng Yu, Zhengying Liu, Yu Zhang, James T. Kwok, Zhenguo Li, Adrian Weller, and Weiyang Liu. Metamath: Bootstrap your own mathematical questions for large language models. In *ICLR*, 2024.

[51] Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

[52] Tianyu Zheng, Ge Zhang, Tianhao Shen, Xueling Liu, Bill Yuchen Lin, Jie Fu, Wenhu Chen, and Xiang Yue. Opencodeinterpreter: Integrating code generation with execution and refinement. In *ACL (Findings)*, pages 12834–12859, 2024.

[53] Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

[54] Hamish Ivison, Yizhong Wang, Valentina Pyatkin, Nathan Lambert, Matthew Peters, Pradeep Dasigi, Joel Jang, David Wadden, Noah A. Smith, Iz Beltagy, and Hannaneh Hajishirzi. Camels in a changing climate: Enhancing lm adaptation with tulu 2, 2023.

[55] Yizhong Wang, Hamish Ivison, Pradeep Dasigi, Jack Hessel, Tushar Khot, Khyathi Raghavi Chandu, David Wadden, Kelsey MacMillan, Noah A. Smith, Iz Beltagy, and Hannaneh Hajishirzi. How far can camels go? exploring the state of instruction tuning on open resources. In *NeurIPS*, 2023.

[56] Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. Stanford alpaca: An instruction-following llama model. `https://github.com/tatsu-lab/stanford_alpaca`, 2023.

[57] Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*, 2018.

[58] Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. Boolq: Exploring the surprising difficulty of natural yes/no questions. In *NAACL-HLT*, pages 2924–2936, 2019.

[59] Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. PIQA: reasoning about physical commonsense in natural language. In *AAAI*, pages 7432–7439, 2020.

[60] Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. Winogrande: an adversarial winograd schema challenge at scale. *Commun. ACM*, 64(9):99–106, 2021.

[61] Shiyue Zhang, Benjamin Frey, and Mohit Bansal. Chren: Cherokee-english machine translation for endangered language revitalization. In *EMNLP*, pages 577–595, 2020.

[62] Xiaodong Chen, Mingming Ha, Zhenzhong Lan, Jing Zhang, and Jianguo Li. Mobe: Mixture-of-basis-experts for compressing moe-based llms. In *ICLR*, 2026.

[63] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *ICLR*, 2019.

[64] An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

A Supplemental Experimental Results

A.1 Results on DeepSeek Model

To further verify the effectiveness of LightMoE, we extend our evaluation to the DeepSeek-V2-Lite model [2], with 15.7B total and 2.4B active parameters. With the exception of the first layer, this model natively consists of 64 routed experts and 2 shared experts per layer. We maintain the same experimental setup and training configurations as those used for the OLMoE model to ensure consistency. Experiments are conducted on RTX PRO 6000 GPUs and Intel Xeon Platinum 8470Q CPUs. The performance comparison across diverse tasks is presented in Table 4.

Table 4: Performance comparison on DeepSeek model across methods and tasks.

Ratio	Method	# Params	Math	Code	Common.	Specialized Tasks		Average
			GSM8K	HumanEval	8 Sub-Tasks	Intent	Translation	
0%	Original	-	3.0	53.9	61.8	0.0	12.2	26.2
0%	LoRA	0.96B	38.9	58.0	62.3	81.0	19.9	52.0
30%	MoBE	0.96B	**54.3**	47.7	**59.7**	80.2	**27.0**	**53.8**
	HC-SMoE	0.96B	36.8	51.6	57.8	75.8	26.9	49.8
	MC-SMoE	0.96B	48.4	53.7	57.0	60.6	26.8	49.3
	LightMoE	0.96B	33.3	**58.1**	59.2	**81.2**	25.3	51.4
40%	MoBE	0.96B	**50.9**	40.6	**57.6**	71.2	24.5	49.0
	HC-SMoE	0.96B	43.0	47.8	55.1	72.2	26.5	48.9
	MC-SMoE	0.96B	47.6	50.2	54.7	61.8	27.5	48.4
	LightMoE	0.96B	33.7	**55.8**	57.3	**80.0**	**27.7**	**50.9**
50%	MoBE	0.96B	39.3	19.2	50.7	63.8	19.9	38.6
	HC-SMoE	0.96B	34.3	43.5	53.0	72.4	**25.8**	45.8
	MC-SMoE	0.96B	**42.2**	38.2	51.3	47.6	22.9	40.4
	LightMoE	0.96B	32.8	**51.5**	**54.3**	**79.4**	24.0	**48.4**

Overall, LightMoE demonstrates robust performance, achieving the best or second-best scores across most tasks. This advantage becomes particularly pronounced at higher compression ratios. While MoBE performs slightly better at a mild 30% compression ratio, it suffers from a drastic performance collapse as compression increases, dropping to an average of 38.6% at the 50% ratio. In contrast, LightMoE maintains superior stability, securing the highest average performance at both 40% and 50% ratios. This highlights the robustness of our approach under aggressive compression constraints.

A critical observation in Table 4 is the performance on the Math task. Notably, LightMoE (33.3%) lags behind the standard LoRA baseline (38.9%), while other compression methods like MoBE (54.3%) and MC-SMoE (48.4%) significantly outperform LoRA. We attribute this anomaly primarily to the rank capacity. Under the constraint of equal trainable parameters, LightMoE operates with a rank of 16, whereas LoRA uses a rank of 54, and other baselines utilize ranks exceeding 80. This trend suggests that complex mathematical reasoning is highly sensitive to low-rank bottlenecks. While our design ensures overall stability, the reduced rank limits the specific capacity for the math task in DeepSeek model. Future work could address this trade-off by exploring adaptive rank allocation strategies.

A.2 Results on Qwen3-30B-A3B-Base

We evaluate LightMoE on Qwen3-30B-A3B-Base [64] to assess its effectiveness on a larger MoE backbone. LoRA and LightMoE use identical training data, training steps, and evaluation settings.

Table 5: Performance comparison on Qwen3-30B-A3B-Base for the Math task. "Total Params" denotes the total model parameter count. GSM8K accuracy is reported in percent.

Ratio	Method	Total Params	GSM8K
0%	Original	30.55B	35.33
0%	LoRA	30.55B	83.85
50%	**LightMoE**	15.27B	82.71

A.3 Calibration Distribution Analysis

Mixed calibration. On OLMoE, we compare task-matched calibration with a 50:50 Math–Code calibration set under the same budget of 2^{17} tokens. Each calibrated model is separately fine-tuned and evaluated on Math or Code.

Table 6: Comparison of calibration compositions on OLMoE. Values are reported as 30% / 40% / 50% compression. Math and Code are evaluated on GSM8K and HumanEval, respectively.

Calibration Data	Math	Code
50:50 Math–Code	56.4 / 57.2 / 55.1	54.3 / 50.8 / **49.7**
Task-matched	**59.7** / **57.9** / **56.1**	**55.7** / **51.3** / 48.4

Calibration-domain shift. We fix fine-tuning and evaluation to Math on OLMoE at 50% compression and vary the calibration source among Math, Code, Commonsense, and a mixture of these domains. All methods use the same calibration data for each setting.

Table 7: GSM8K accuracy on OLMoE at 50% compression under different calibration distributions.

Calibration Data	LightMoE	HC-SMoE	MC-SMoE
Math	**56.1**	26.2	50.1
Code	**40.3**	26.5	38.7
Commonsense	**30.4**	16.7	12.4
Mixed	**51.5**	28.7	47.9

A.4 Impact of Expert Selection Mechanism

Selection strategy. To assess the effectiveness of our adaptive thresholding approach, we conduct an ablation study on different expert selection methods: "Uniform" (constant threshold per layer), "Average" (same number of selected experts per layer), and "Adaptive" (ours). Experimental results are produced on the Math and Code tasks, as shown in Table 8. At a low compression ratio, our method performs comparably to the "Average" selection method but outperforms the "Uniform" selection method. At a high compression ratio, our method performs similarly to the "Uniform" but significantly outperforms the "Average". This indicates that our method can select the most appropriate experts for replacing, adapting to different compression ratios effectively.

Sample size for importance scoring. To evaluate the amount of data needed to select the less important experts for a task, we assess the model's performance on the Math task using varying token counts, ranging from 2^{13} to 2^{19}. As shown in Table 9, we observe that the performance improves as the number of tokens increases from 2^{13} to 2^{17}, highlighting the necessity of adequate data for accurate expert selection. However, performance gains saturate as the sample size reaches 2^{17}. Specifically, increasing the sample size from 2^{17} to 2^{19} yields minimal gains across all compression ratios. This indicates that the sample size of 2^{17} is sufficiently large to identify the less important experts.

Adaptive threshold sensitivity. To provide a systematic justification for the hyperparameter choice in our adaptive thresholding mechanism, we introduce a parameter $\max\Delta$ to control the bounds of

Table 8: Comparison of different expert selection methods. Our adaptive thresholding method can select the most appropriate experts for replacing, adapting to different compression ratios.

Ratio	Selection Method	Math	Code
		GSM8K	HumanEval
30%	Uniform	58.1	55.1
	Average	<u>59.4</u>	<u>55.5</u>
	Adaptive	**59.7**	**55.7**
50%	Uniform	**57.1**	<u>47.9</u>
	Average	54.6	43.0
	Adaptive	<u>56.1</u>	**48.4**

Table 9: Ablation study on the impact of sample size on OLMoE performance.

Ratio	Number of Tokens			
	2^{13}	2^{15}	2^{17}	2^{19}
30%	57.4	59.1	**59.7**	59.6
40%	55.1	56.6	<u>57.9</u>	**58.4**
50%	53.9	54.5	<u>56.1</u>	**56.3**

the layer-specific thresholds. Specifically, the minimum and maximum thresholds are defined as:

$$p_{\min} = (1 - \max\Delta)\hat{p}, \quad p_{\max} = (1 + \max\Delta)\hat{p}. \tag{8}$$

We evaluate the model performance on the Math task with $\max\Delta \in \{0, 0.1, 0.15, 0.2, 0.25, 0.3, \infty\}$. Notably, $\max\Delta = 0$ is equivalent to the "Uniform" strategy, while $\max\Delta = \infty$ removes the clipping bounds entirely.

Table 10: Sensitivity analysis of the clipping range parameter $\max\Delta$ for OLMoE on the Math task.

Ratio	Clipping Range $\max\Delta$						
	0	0.1	0.15	0.2	0.25	0.3	∞
30%	58.1	59.4	**59.7**	**59.7**	59.3	58.6	58.6
50%	**57.1**	55.6	55.8	<u>56.1</u>	54.1	54.3	54.3

The results presented in Table 10 highlight two key observations. First, at a moderate 30% compression ratio, our adaptive approach significantly outperforms the uniform baseline. Introducing a reasonable clipping range improves performance from 58.1% to 59.7%, confirming the benefit of varying thresholds based on layer importance. Second, the results demonstrate that moderate clipping is essential. Removing the bounds entirely leads to performance degradation compared to the optimal range (e.g., dropping from 59.7% to 58.6% at the 30% ratio). This suggests that while flexibility is beneficial, preventing the thresholds from becoming too extreme is necessary for stability.

In conclusion, values between 0.15 and 0.2 consistently yield robust performance across different compression ratios. This validates our default choice of $\max\Delta = 0.2$ as a systematic and effective setting that balances adaptivity with numerical stability.

A.5 Impact of Expert Grouping Mechanism

Expert grouping strategy. To explore different ways of grouping experts, we compare our method with the K-means clustering strategy, which is adapted from HC-SMoE [10]. As shown in Table 11, both strategies perform similarly at a mild 30% ratio. However, as the compression becomes more aggressive, our method demonstrates a clear advantage. At the 40% and 50% ratios, our approach significantly outperforms K-means. We attribute this performance gap to the distinct mechanisms of group formation. Our method focuses on preserving the "dominant" experts and grouping similar experts around them. In contrast, K-means computes average centroids for all experts. When the compression is high, K-means risks inadvertently merging critical experts into a general group, leading to the irreversible loss of expert-specific knowledge. This suggests that explicitly preserving dominant experts is crucial to maintaining performance under high compression.

Table 11: Comparison of expert grouping strategies for OLMoE on the Math task.

Ratio	K-means	Dominant (Ours)
30%	**60.0**	59.7
40%	56.1	**57.9**
50%	54.9	**56.1**

Group size. We explore the impact of incrementally increasing the group size of LightMoE from 1 to 4. Figure 6 illustrates performance across various group sizes and compression ratios on the Math task. At a mild 30% compression ratio, the performance of all group sizes vary by less than 0.9%, showing that multiple shared bases has little effect when ample parameters remain.

In contrast, as compression becomes more aggressive, the benefit of larger group sizes becomes pronounced. In particular, at a 50% compression ratio, performance improves with increasing group size. This trend suggests that under constrained parameter budgets, constructing multiple shared bases by grouping is crucial for preserving the model's knowledge and capabilities.

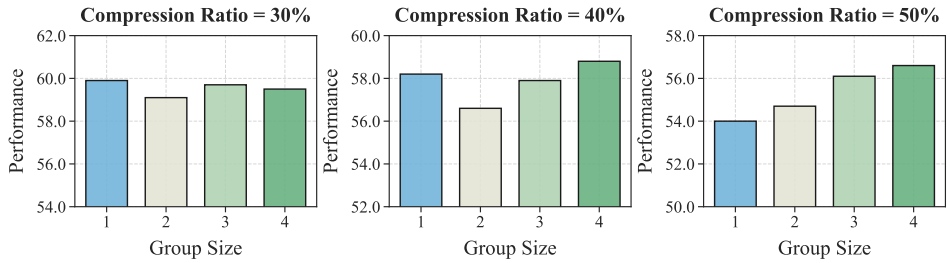


Figure 6: Comparison of different group sizes for OLMoE on the Math task.

A.6 Exploration of Annealing Schedules

To investigate the impact of the annealing schedule on model performance, we compare our default linear decay strategy against an exponential decay strategy. While the linear strategy decreases β at a constant rate, the exponential strategy introduces a curvature controlled by a hyperparameter γ. Formally, maintaining consistency with the notation in Section 3.3, the annealing factor β under the exponential schedule is defined as:

$$\beta = \max\left(\frac{e^{-\gamma\tau} - e^{-\gamma}}{1 - e^{-\gamma}}, 0\right), \quad \text{where } \tau = \frac{t}{\epsilon T}. \tag{9}$$

Here, τ represents the normalized training progress relative to the annealing period, and γ modulates the decay shape. A larger γ causes β to drop more rapidly initially, whereas a smaller γ maintains a higher value for a longer duration. Visualizations of these decay trajectories are provided in Figure 7.

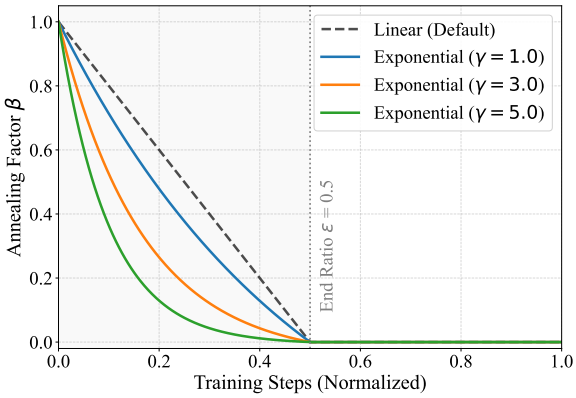


Figure 7: Comparison of different annealing schedules. The dashed line represents the linear decay strategy, while solid lines depict exponential decay strategies with varying γ. All schedules converge to zero at the end ratio ϵ.

Table 12: Comparison of linear and exponential annealing schedules for OLMoE on the Math task.

Ratio	Linear	Exponential Decay		
		$\gamma = 1.0$	$\gamma = 3.0$	$\gamma = 5.0$
30%	<u>59.7</u>	**59.8**	59.6	58.7
40%	57.9	56.1	<u>58.2</u>	**58.4**
50%	**56.1**	54.5	54.4	<u>56.0</u>

We evaluate this strategy with γ values of 1.0, 3.0, and 5.0. The results on the Math task are summarized in Table 12. Overall, the performance differences between the linear and exponential schedules are marginal. However, the linear schedule demonstrates remarkable robustness across different compression levels. It achieves the best performance at the aggressive 50% compression ratio and ranks second-best at the 30% ratio. Even at 40%, where it falls slightly behind the exponential setting, the gap is negligible. In contrast, the exponential strategy proves sensitive to the hyperparameter γ. A lower γ performs well at low compression but degrades at high compression, whereas a higher γ shows the opposite trend. No single γ value yields optimal results across all scenarios. Consequently, we adopt the Linear schedule as a simple yet sufficiently effective solution, as it delivers consistently high performance without the need for additional hyperparameter tuning.

A.7 Multi-seed Robustness

To further validate the robustness of our method, we report OLMoE results at 50% compression across three random seeds in Table 13. Values are reported as mean and standard deviation.

Table 13: OLMoE results at 50% compression across three seeds.

Method	Math	Code	Common.	Intent	Translation	Avg.
MoBE	22.1 ± 3.0	13.7 ± 3.5	46.8 ± 0.4	55.3 ± 2.1	26.4 ± 4.8	32.9 ± 2.6
HC-SMoE	26.1 ± 4.6	27.6 ± 1.0	47.8 ± 0.1	57.1 ± 0.8	16.3 ± 4.3	35.0 ± 1.5
MC-SMoE	50.2 ± 0.3	44.7 ± 1.6	45.4 ± 0.5	65.3 ± 3.9	9.4 ± 1.9	43.0 ± 0.9
LightMoE	54.8 ± 1.3	48.5 ± 0.2	48.3 ± 0.2	73.5 ± 0.8	15.3 ± 2.0	$\mathbf{48.1 \pm 0.3}$

LightMoE remains the strongest method on average, with a score of 48.1 ± 0.3, outperforming MC-SMoE (43.0 ± 0.9), HC-SMoE (35.0 ± 1.5), and MoBE (32.9 ± 2.6). It also has the smallest standard deviation in average performance, demonstrating stronger robustness across random seeds.

B Detailed Efficiency Analysis

Training Efficiency. We conduct a system-level training-cost comparison on the OLMoE Math task using a single RTX PRO 6000 GPU. We report the training pre-cost, peak training GPU memory, mean step time over 2,000 training steps, total training time, and end-to-end cost at $1/GPU-hour. Following the main experimental setting, HC-SMoE, MC-SMoE, MoBE, and LightMoE use batch size 8 and gradient accumulation 4, giving an effective batch size of 32. For MC-SMoE*, we additionally report 4x8 and 8x4 settings, where the first number denotes batch size and the second denotes gradient accumulation steps. For MoBE, the 1000-epoch and 10000-epoch settings refer to the number of optimization epochs used in the reconstruction-loss-based expert decomposition stage before downstream training, and the 10000-epoch setting is the released-code setting used in our OLMoE experiments.

The results in Table 14 show that LightMoE provides a favorable training-cost trade-off. Compared with MC-SMoE* (8x4), LightMoE reduces peak training memory from 77.83 GB to 56.61 / 56.60 GB, mean step time from 14.10 / 14.03s to 5.89 / 6.03s, and end-to-end cost from 7.88 / 7.84 to 3.33 / 3.41 USD at 40% / 50% compression. Compared with MoBE under the 10000-epoch setting, LightMoE reduces end-to-end cost from 7.12 / 6.08 / 5.11 to 3.21 / 3.33 / 3.41 USD at 30% / 40% / 50%. Compared with MC-SMoE and HC-SMoE, LightMoE remains in the same 0.45B trainable-parameter regime while achieving substantially stronger task performance in the main results. Overall, these results suggest that LightMoE achieves a strong balance among memory efficiency, training efficiency, and model performance.

Table 14: Training-cost comparison for OLMoE on the Math task. Values are reported as 30% / 40% / 50% compression.

Method	Pre-cost (min)	Trainable Params (B)	Peak Mem. (GB)	Step Time (s)	Train Time (h)	Total Cost ($)
MC-SMoE	3 / 3 / 3	0.45/ 0.45 / 0.45	58.46 / 56.40 / 54.66	4.83 / 4.70 / 4.42	2.68 / 2.61 / 2.46	2.73 / 2.66 / 2.50
HC-SMoE	1 / 1 / 1	0.45 / 0.45 / 0.45	58.11 / 56.80 / 54.34	4.76 / 4.68 / 4.42	2.65 / 2.60 / 2.46	2.66 / 2.61 / 2.47
MoBE (1000 ep)	24 / 18 / 12	0.45 / 0.45 / 0.45	59.83 / 53.69 / 47.26	5.61 / 5.55 / 5.60	3.12 / 3.08 / 3.11	3.52 / 3.38 / 3.31
MoBE (10000 ep)	240 / 180 / 120	0.45 / 0.45 / 0.45	59.83 / 53.69 / 47.26	5.61 / 5.55 / 5.60	3.12 / 3.08 / 3.11	7.12 / 6.08 / 5.11
MC-SMoE* (4x8)	– / 3 / 3	– / 1.65 / 1.65	– / 71.81 / 71.83	– / 16.24 / 16.18	– / 9.02 / 8.99	– / 9.07 / 9.04
MC-SMoE* (8x4)	– / 3 / 3	– / 1.65 / 1.65	– / 77.83 / 77.83	– / 14.10 / 14.03	– / 7.83 / 7.79	– / 7.88 / 7.84
LightMoE	4 / 4 / 4	0.45 / 0.45 / 0.45	56.59 / 56.61 / 56.60	5.68 / 5.89 / 6.03	3.16 / 3.27 / 3.35	3.21 / 3.33 / 3.41

Inference Efficiency. To assess the inference efficiency of our approach, we evaluate the model across four key metrics: total parameter count, GPU memory usage, MoE average active parameters per token, and inference latency. The results for the OLMoE model on the Math task are summarized in Table 15.

Table 15: Inference efficiency comparison on the Math task under different compression ratios.

Ratio	Total Params	Memory Usage	MoE Active Params	Latency (ms/token)
0%	6.92 B	12.89 GB	0.81 B	8.09
30%	4.82 B	9.09 GB	0.77 B	8.71
40%	4.14 B	7.85 GB	0.73 B	8.32
50%	3.47 B	6.63 GB	0.70 B	8.01

As the compression ratio increases, we observe a substantial reduction in memory requirements. Specifically, the memory footprint decreases by nearly half, dropping from 12.89 GB in the original model to 6.63 GB at the 50% compression ratio. In terms of computational cost, the decrease in the number of parameters activated within the MoE layers is less pronounced. This is expected, as our method primarily targets and compresses redundant experts that are less frequently activated, while preserving the critical experts that contribute most to the active parameter count. Moreover, although replacing experts with shared bases and adapters introduces slight architectural complexity, our method maintains an inference latency comparable to the original model. These findings demonstrate that LightMoE effectively reduces parameter redundancy and memory usage without compromising inference efficiency.

C Training Dynamics Analysis

To validate the intuition behind our annealing strategy, we visualize the training dynamics of LightMoE compared to the Directly Replacing baseline in Figure 8.

As illustrated in the inset of Figure 8, the Directly Replacing baseline suffers from a severe optimization instability, evidenced by a sharp initial spike in loss. This indicates that the abrupt parameter substitution destabilizes the model, forcing the optimization trajectory to recover from a high-loss, sub-optimal state. In contrast, LightMoE ensures a smooth start with minimal loss, effectively mitigating the destructive initial shock.

As training progresses and β decays, LightMoE exhibits a slight rise in loss. This phenomenon represents the critical transfer of capabilities from the original experts to the compressed modules. Effectively, the annealing phase acts as a warm-up period, allowing the compressed modules to implicitly align with the original experts' behavior. Crucially, despite this transient rise, the loss trajectory of LightMoE remains lower than that of the baseline throughout the training process. This suggests that the smooth transition keeps the model in a better region of the parameter space, avoiding the suboptimal local minima that the baseline falls into.

Moreover, although our current strategy is effective, the temporary rise in loss implies a brief gap in capability during the transition. Future work could explore more adaptive schedules and better initialization methods to further mitigate this fluctuation, thereby enhancing the training stability of the expert replacing paradigm.

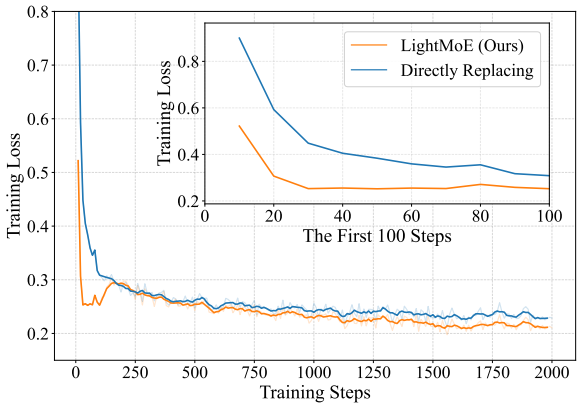


Figure 8: Training loss trajectories of LightMoE versus the Directly Replacing baseline on the Math task with OLMoE at a 50% compression ratio.

D Base Threshold Settings

To facilitate reproducibility, we provide the specific hyperparameter configurations used to achieve the target compression ratios for both the OLMoE and DeepSeek models. In the LightMoE framework, the global compression ratio is controlled by the base threshold $\hat{p}$ defined in Equation (3). This threshold determines the subset of experts selected for replacement. A higher $\hat{p}$ results in more experts being replaced, thereby increasing the compression ratio.

Since the distribution of importance scores varies across domains, the base threshold required to select these experts differs. Therefore, for each task and target compression ratio, we determine the optimal $\hat{p}$ via a binary search process on the calibration dataset. Table 16 presents these $\hat{p}$ values for the five evaluated tasks, alongside the corresponding average number of experts selected per layer.

A distinct pattern emerges from the results. For preservation tasks, including Math, Code, and Commonsense Reasoning, the base thresholds are relatively high. This suggests that in these domains, even the less critical experts selected for replacement retain a moderate level of activation and contribution. Conversely, for adaptation tasks like Intent Recognition and Translation, the thresholds are significantly lower. This indicates that the expert activation in these specialized downstream tasks is highly sparse, allowing a large number of redundant experts to be identified even with a very low threshold.

Table 16: Base threshold $\hat{p}$ settings calibrated for OLMoE and DeepSeek models. The average number of selected experts per layer (Avg # Exp.) corresponds to the target compression ratio.

Ratio (%)	**Avg # Exp.**	**Base Threshold $\hat{p}$** Math	Code	Common.	Intent	Trans.
Model: OLMoE-1B-7B-SFT						
30	≈ 24	0.110	0.095	0.185	0.055	0.036
40	≈ 31	0.180	0.159	0.275	0.098	0.059
50	≈ 38	0.268	0.240	0.382	0.168	0.092
Model: DeepSeek-V2-Lite						
30	≈ 26	0.160	0.200	0.192	0.085	0.058
40	≈ 33	0.240	0.285	0.280	0.142	0.098
50	≈ 40	0.330	0.385	0.380	0.215	0.146

E Baseline Details

Baseline configurations. We compare our proposals to a comprehensive set of baselines. First, we evaluate state-of-the-art expert compression methods. MC-SMoE [8] is evaluated in two variants. The original version initializes by merging experts to a 30% compression ratio and then uses progressive low-rank decomposition to reach 40% and 50% compression. Since this version requires gradients from the original experts and introduces substantial overhead, we also test a modified version using LoRA fine-tuning to ensure a fair computational comparison. Additionally, we include HC-SMoE [10], which employs hierarchical clustering based on expert outputs to ensure robust merging independent of routing decisions. Furthermore, we evaluate MoBE [62], which decomposes expert weights into shared basis matrices and expert-specific transformations. Second, due to the lack of prior work on expert replacing, we introduce two strong baselines: "Replace (w/o shared)" and "Replace (w. shared)", which directly replace less important experts with LoRA adapters, either without adding shared base or with one shared base, respectively. Both use LoRA fine-tuning to recover performance. Finally, full fine-tuning and LoRA fine-tuning on the original model are included as uncompressed references.

Distinction from MoBE. Although MoBE can be viewed as one possible instantiation of replacing original experts with compact surrogates, it differs from LightMoE in several key aspects. Although MoBE can be viewed as one possible instantiation of replacing original experts with compact surrogates, it differs from LightMoE in several key aspects. First, for expert selection, LightMoE selectively identifies less important experts and preserves important ones, whereas MoBE factorizes all experts in each layer. Second, for module construction, LightMoE uses a grouped shared-expert plus low-rank-adapter form, e.g., $W^* = W_{\text{share}} + BA$, to preserve diversity. In contrast, MoBE factorizes every expert as $W^* = BA$, with A further reparameterized by basis matrices shared within a layer. Third, for efficient recovery, LightMoE introduces annealed expert replacement to stabilize the transition from the original experts to their compressed forms, whereas MoBE relies on reconstruction-loss-based decomposition and does not introduce a dedicated transition strategy during downstream adaptation. These design differences are especially important under aggressive compression, where LightMoE substantially outperforms MoBE in our main results.

Pruning-based baselines. Expert pruning is also an important paradigm, but we do not include these methods mainly for reproducibility. Several representative pruning methods [28, 7] are difficult to evaluate because complete implementations are not publicly available. Although NAEE [6] provides public code, its combinatorial explosion makes it impractical for fine-grained MoEs. Therefore, we focus our main comparison on MC-SMoE, HC-SMoE, and MoBE, which are open-source, reproducible, and strong prior methods that can be compared under a consistent pipeline.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction clearly state the claims made.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: It is discussed in the conclusion.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [N/A]

Justification: The paper does not include theoretical results.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: The paper fully disclose all the information needed to reproduce the main experimental results.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: The code will be released after the paper is accepted.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: All the training and test details are introduced.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No]

   Justification: We follow common practice in this field and do not include error bars in the results.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.

- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We report the default GPU/CPU hardware used in the experimental setup and specify the separate hardware used for DeepSeek experiments. We also provide a training-cost analysis in the appendix, including peak GPU memory, mean step time, total training time, and end-to-end cost.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: It is discussed in the conclusion.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The paper poses no such risks.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: The creators or original owners of assets are properly credited.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.

- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [N/A]

    Justification: The paper does not release new assets.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [N/A]

Justification: The core method development in this research does not involve LLMs as any important, original, or non-standard components.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.