SlimMoE: Structured Compression of Large MoE Models via Expert Slimming and Distillation

Zichong Li[1†*], **Chen Liang**[2†], **Zixuan Zhang**[1], **Ilgee Hong**[1], **Young Jin Kim**[2], **Weizhu Chen**[2], **Tuo Zhao**[1]
[1]Georgia Institute of Technology [2]Microsoft

Abstract

The Mixture of Experts (MoE) architecture has emerged as a powerful paradigm for scaling large language models (LLMs) while maintaining inference efficiency. However, their substantial memory requirements make them prohibitively expensive to fine-tune or deploy in resource-constrained environments. To address this challenge, we propose *SlimMoE*, a multi-stage compression framework that transforms large MoE models into significantly smaller and more efficient variants without the cost of training from scratch. Our method systematically reduces parameter counts by slimming experts and transferring knowledge through intermediate stages, effectively mitigating the performance degradation typical of one-shot pruning. Using SlimMoE, we compress Phi-3.5-MoE (41.9B total / 6.6B activated parameters) into two smaller models: Phi-mini-MoE (7.6B total / 2.4B activated) and Phi-tiny-MoE (3.8B total / 1.1B activated), using only 400B tokens – less than 10% of the original training data. These models can be fine-tuned on a single GPU (A100 for Phi-mini-MoE, A6000 for Phi-tiny-MoE), making them well suited for academic and resource-limited settings. Our experiments show that the compressed models outperform others of similar size and remain competitive with larger models. For example, Phi-mini-MoE matches or exceeds the performance of Phi-3-mini while using only two-thirds of the activated parameters and achieves comparable MMLU scores to LLaMA 3.1 8B with significantly lower latency. These results highlight that structured pruning combined with multi-stage distillation is an effective strategy for building high-quality, compact MoE models, enabling broader adoption of MoE architectures across diverse computational environments. We release our models at https://huggingface.co/microsoft/Phi-mini-MoE-instruct and https://huggingface.co/microsoft/Phi-tiny-MoE-instruct.

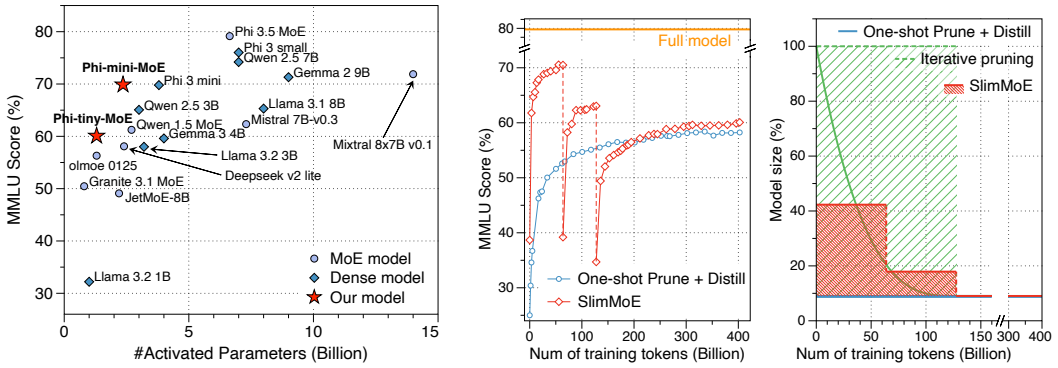
1 Introduction

Large language models (LLMs) have demonstrated remarkable capabilities across various domains (Dubey et al., 2024; DeepSeek-AI et al., 2024b). The Mixture of Experts (MoE) architecture has emerged as a particularly effective approach in open-source LLMs (DeepSeek-AI et al., 2024b; Abdin et al., 2024; Jiang et al., 2024; Team, 2024), offering superior performance compared to dense models while maintaining greater inference efficiency through sparse parameter activation.

Despite these advantages, state-of-the-art MoE models developed by industry leaders typically contain an enormous number of parameters and rely on abundant computational resources. Examples include Phi-3.5-MoE (Abdin et al., 2024) and DeepSeek-V3 (DeepSeek-AI et al., 2024b), which are prohibitively large and extremely costly to train and deploy. These high costs limit their practical use in resource-constrained environments such as academic

[*]Work is done during internship at Microsoft.
[†]Correspondence to zli911@gatech.edu and chenliang1@microsoft.com.

research, making it difficult for researchers to fine-tune such models and highlighting the need for smaller, more efficient MoE alternatives.



(a) (b) (c)

Figure 1: (a) Comparison of MMLU scores with recent open-sourced MoE and dense LLMs. (b) Comparison of MMLU scores between one-shot approach and SlimMoE multi-stage approach on Phi-tiny-MoE. (c) Illustration of model size change throughout the compression process. Shaded area indicate computation overhead of approaches over one-shot approach.

Training smaller MoE models from scratch is prohibitively expensive in terms of time, data, and computational resources (Team, 2024; Muennighoff et al., 2024). Given the availability of well-trained large MoE models, this work aims to obtain smaller, more efficient MoE models by compressing larger ones while preserving performance. This approach leverages existing models without incurring full-scale training costs. Specifically, we use the pre-trained Phi-3.5-MoE model (Abdin et al., 2024) as our starting point and scale it down to two target sizes: 7.6B total parameters (2.4B active) and 3.8B total parameters (1.1B active). These sizes are strategically chosen to enable fine-tuning on widely available hardware, e.g., the 7.6B model can be fine-tuned on a single A100 80GB GPU using memory-efficient optimizers such as 8-bit Adam (Dettmers et al., 2021) or Muon (Jordan et al., 2024), while the 3.8B model can be fine-tuned on an A6000 48GB GPU.

Among various compression methods, we focus on *structured pruning* (Ma et al., 2023; Xia et al., 2024; Muralidharan et al., 2024; Xia et al., 2022; Liang et al., 2023), which reduces parameter counts by systematically removing blocks of weights. However, compressing MoE models at high pruning ratios while preserving performance remains a significant challenge. A common strategy is to apply *one-shot pruning* to shrink the model to the target size in a single step, followed by *knowledge distillation* to recover performance (Muralidharan et al., 2024). Yet, pruning a large number of parameters at once can result in substantial performance degradation (Figure 1(b)), especially when entire architectural components, such as experts, are removed (Muzio et al., 2024; Chen et al., 2022; Chowdhury et al., 2024). Such severe degradation may hinder the effectiveness of distillation in recovering the original performance (Cho & Hariharan, 2019; Mirzadeh et al., 2020). An existing approach to address this challenge is iterative pruning (Liang et al., 2023), which gradually removes parameters during knowledge distillation to avoid a large performance drop at once and facilitate progressive recovery over time. However, iterative pruning is expensive, as it uses masks to simulate pruning and requires loading the full model throughout the process (Figure 1 (c)), making it impractical for compressing LLMs.

To address these challenges, we propose *SlimMoE*, a multi-stage expert slimming and distillation framework that mitigates the performance degradation of one-shot pruning with only modest additional computation. *SlimMoE* compresses MoE models at high ratios while preserving performance, using less than 10% of the original training data. It introduces two key design components: (1) Instead of pruning entire experts, SlimMoE retains all experts and prunes only redundant neurons within each one, preserving expert-specific knowledge critical to model performance (Section 4.5). (2) Rather than pruning directly to the target size, SlimMoE first applies one-shot pruning to an intermediate size to avoid drastic performance degradation, followed by knowledge distillation to restore accuracy

(Figure 1(b)). This prune-and-distill process is repeated until the target size is reached, with extended distillation applied at the final stage. As intermediate models retain sufficient capacity for knowledge transfer and recover quickly, this multi-stage approach ensures a smoother and more stable compression trajectory. Compared to iterative pruning, SlimMoE avoids pruning masks and full-model loading, resulting in significantly lower computational overhead (see Figure 1(c) for illustration and Section 4.2 for details).

Using our proposed method, we introduce Phi-mini-MoE (7.6B total / 2.4B active parameters) and Phi-tiny-MoE (3.8B total / 1.1B active parameters), which are compressed from Phi-3.5-MoE (41.9B total/6.6B activated parameters) with 400B tokens of continual pre-training[1]. As shown in Figure 1(a), both models substantially outperform dense and MoE baselines with similar active parameter counts and achieve competitive performance compared to models with higher inference costs. For example, Phi-mini-MoE reaches similar MMLU scores to Phi-3-mini and LLaMA 3.1 8B, while Phi-tiny-MoE outperforms LLaMA 3.2 3B and matches the performance of Qwen 1.5 MoE (See Figure 4 for comparison on inference costs). In summary, our contributions are as follows:

• We propose SlimMoE, a multi-stage expert slimming and distillation framework that enables high-ratio compression of MoE models while preserving competitive performance.

• We release two compact MoE models, Phi-mini-MoE and Phi-tiny-MoE, which achieve competitive performance among models of comparable size.

• We conduct extensive ablation studies to validate the effectiveness of SlimMoE and show that MoE architectures can be more robust to pruning than their dense counterparts.

2 Background

2.1 Mixture of Experts Models

The Mixture of Experts (MoE) architecture enhances the standard Transformer by replacing its feed-forward network (FFN) layers with MoE layers, where each input token activates only a subset of experts. This sparse activation enables the model to scale in capacity without a proportional increase in computational cost (Fedus et al., 2022; Lepikhin et al., 2020; Shazeer et al., 2017; Jiang et al., 2024; Abdin et al., 2024; Liu et al., 2024b).

An MoE layer consists of n_{expert} expert networks and a router. Given an input $\mathbf{x} \in \mathbb{R}^{d_{\text{model}}}$, each expert typically adopts a Gated Linear Unit (GLU) structure (Shazeer, 2020):

$$\text{Expert}_e(\mathbf{x}) = (\text{Act}(\mathbf{x}W_{1e}^E) \odot \mathbf{x}W_{2e}^E)W_{3e}^E,$$

where $W_{1e}^E, W_{2e}^E \in \mathbb{R}^{d_{\text{model}} \times d_{\text{expert}}}$, $W_{3e}^E \in \mathbb{R}^{d_{\text{expert}} \times d_{\text{model}}}$, $e \in \{1, \ldots, n_{\text{expert}}\}$, $\text{Act}(\cdot)$ is an activation function (e.g., GELU), and $\odot$ denotes element-wise multiplication.

The router is parameterized by $W^G \in \mathbb{R}^{d_{\text{model}} \times n_{\text{expert}}}$ and produces n_{expert}-dimentional scores:

$$G(\mathbf{x}) = \text{Gating}(\text{TopK}(\mathbf{x}W^G)),$$

where $\text{TopK}(\mathbf{x}W^G)_i = (\mathbf{x}W^G)_i$ if i is among the top-k highest routing logits, and $-\infty$ otherwise. $\text{Gating}(\cdot)$ is a gating function (e.g., softmax). The output of the MoE layer is:

$$\text{MoE}(\mathbf{x}) = \sum_{e=1}^{n_{\text{expert}}} G(\mathbf{x})_e \cdot \text{Expert}_e(\mathbf{x}).$$

In this paper, we focus on Phi-3.5-MoE (Abdin et al., 2024), which sets $n_{\text{expert}} = 16$, selects the top-2 experts, and uses the SparseMixer-v2 routing mechanism (Liu et al., 2023c;b; 2024b). SparseMixer-v2 enables gradient flow through routing by replacing hard TopK with stochastic expert selection and applying a third-order approximation to the routing gradient, improving training stability and scalability.

[1]The released models (instruct version) are further post-trained with supervised fine-tuning (SFT) and direct preference optimization (DPO) for instruction following and preference alignment.

The self-attention mechanism operates on an input sequence $\mathbf{X} \in \mathbb{R}^{N \times d_{\text{model}}}$, where N is the sequence length (Vaswani et al., 2017). Multi-head self-attention is computed as:

$$\text{MHA}(\mathbf{X}) = \text{Concat}(\text{head}_1, \ldots, \text{head}_H)W^O, \qquad \text{head}_h = \text{Attn}(\mathbf{X}W_h^Q, \mathbf{X}W_h^K, \mathbf{X}W_h^V),$$

where H is the number of heads; $W_h^Q, W_h^K, W_h^V \in \mathbb{R}^{d_{\text{model}} \times d_{\text{head}}}$ are the query, key, and value projection matrices for head h; and $W^O \in \mathbb{R}^{H \cdot d_{\text{head}} \times d_{\text{model}}}$ is the output projection matrix.

Further, grouped-query attention (GQA, Ainslie et al. (2023)) offers an efficient alternative to full multi-head attention. It partitions the H query heads into multiple groups and lets every head in a group share the same key and value projection.

2.2 Weight Pruning

Weight pruning is an effective compression technique that removes parameters from a model based on predefined importance criteria (Han et al., 2015b;a; Louizos et al., 2017). Commonly used metrics are *sensitivity* and its variants (Molchanov et al., 2016; 2019; Sanh et al., 2020; Zhang et al., 2022), which estimate the impact of removing individual parameters. Let W denote the model weights and $L(\mathbf{X}; W)$ the loss function. The sensitivity score for weight $W_{i,j}$ is computed as:

$$s_{i,j} = \left| \frac{\partial L}{\partial W_{i,j}} \odot W_{i,j} \right|,$$

which approximates the change in loss when setting $W_{i,j}$ to zero, using a first-order Taylor expansion of L around W. Weights with low sensitivity are considered redundant.

Pruning methods are typically categorized into *unstructured* and *structured* approaches. Unstructured pruning removes individual weights, resulting in sparse weight matrices. While this often causes minimal performance degradation, it does not yield practical speedup without specialized hardware support and tensor structure (Fang et al., 2024). Structured pruning, on the other hand, removes entire architectural components (e.g., neurons, attention heads, or experts), which enables real efficiency gains on standard hardware but may cause greater disruption to model performance and has been widely adopted for compressing LLMs in recent years (Ma et al., 2023; Xia et al., 2024; Muralidharan et al., 2024; Xia et al., 2022; Liang et al., 2023).

2.3 Knowledge Distillation

Knowledge Distillation (KD) is a widely used compression technique that trains a smaller student model to mimic the behavior of a larger teacher model by minimizing the discrepancy between their output distributions (Hinton et al., 2015; Muralidharan et al., 2024). The objective is typically formulated as the Kullback–Leibler (KL) divergence:

$$L_{\text{KD}}(\mathbf{X}; W) = \frac{1}{N} \sum_{n=1}^{N} \text{KL}\left(p_{\text{teacher}}^n(\mathbf{X}) \parallel p_W^n(\mathbf{X})\right), \tag{1}$$

where W denotes the student's parameters, $p_{\text{teacher}}^n(\mathbf{X})$ and $p_W^n(\mathbf{X})$ are the teacher and student next-token distributions for the n-th token in input $\mathbf{X}$.

To improve the efficiency and robustness of knowledge distillation, Peng et al. (2024) proposed to minimize the discrepancy over the k tokens with the highest predicted probabilities instead of the full vocabulary distribution, reducing computational requirements and mitigating noise from low-probability tokens.

3 Method

SlimMoE employs a multi-stage framework that progressively reduces model size by alternating expert and attention pruning with knowledge distillation. We begin by describing the target MoE architecture, followed by our pruning and distillation strategies.

3.1 Target Architecture

Expert Slimming. Since MoE layers account for over 95% of the model's parameters, we focus on pruning these layers. Specifically, we reduce the expert dimension d_{expert} by pruning redundant neurons within each expert network, i.e., $\{W^E_{1e}, W^E_{2e}, W^E_{3e}\}_{e=1}^{n_{\text{expert}}}$ in Eq. 2.1, while keeping the number of experts fixed. As experts often serve specialized roles for subsets of tokens, this approach better preserves expert functionality and results in smaller accuracy degradation than pruning entire experts (Section 4.5). We apply uniform slimming across all experts to maintain equal expert size for architectural consistency and deployment.

Attention Pruning. We also prune attention layers as they begin to dominate the parameter count and inference time at smaller scales. Since Phi-3.5-MoE adopts GQA that ties four query heads to a shared key/value projection, we prune at the granularity of entire GQA groups, eliminating both the shared KV pair and its four associated queries.

Target Architecture. We reduce the expert dimension of Phi-3.5-MoE (41.9B total / 6.6B activated parameters) to 15% of its original size to yield Phi-mini-MoE (7.6B total / 2.4B activated parameters). To obtain Phi-tiny-MoE (3.8B total / 1.1B activated parameters), we further reduce the expert dimension to 7% and prune 50% of the GQA groups. Table 6 in the Appendix summarizes the detailed architecture configurations.

3.2 Multi-stage Pruning and Distillation

We prune and distill the model over T stages, with each stage producing a smaller model than the previous one. At each stage, we apply one-shot pruning to reach an intermediate size, followed by distillation from the original full model.

One-shot Pruning. We use the top-8 logits distillation loss to compute the sensitivity score for each parameter $W_{i,j}$:

$$s_{i,j} = \left| \frac{\partial L^{\text{MoE}}_{\text{KD}}(\mathbf{X}; W)}{\partial W_{i,j}} \odot W_{i,j} \right|, \tag{2}$$

where the loss is defined as:

$$L^{\text{MoE}}_{\text{KD}}(\mathbf{X}; W) = \frac{1}{N} \sum_{n=1}^{N} \text{KL}\left(p^n_{\text{teacher, top-8}}(\mathbf{X}) \parallel p^n_W(\mathbf{X}) \right) + \text{Aux}(W). \tag{3}$$

Here, $p^n_{\text{teacher, top-8}}(\mathbf{X})$ is the teacher's next-token prediction probability distribution for the n-th token with all but the top-8 probabilities masked to 0, and $p^n_W(x)$ is the student distribution as defined in Eq. 1. $\text{Aux}(\cdot)$ is the auxiliary load-balancing loss (Liu et al., 2024b).

We compute the sensitivity score using the knowledge distillation loss as it captures the discrepancy between the student and teacher. This allows us to identify redundant neurons that have negligible impact on the gap, yielding better pruning performance than alternative loss metrics (Section 4.5).

For the e-th expert network, we first compute the sensitivity score for each parameter in the down-projection matrix W^E_{3e} of the GLU. We then aggregate these into a neuron-level score by taking the ℓ_2-norm across each row:

$$s_i = \sqrt{\sum_j s^2_{i,j}}. \tag{4}$$

Neurons with the lowest scores are considered least important. We prune these by removing the corresponding rows and columns in W^E_{1e}, W^E_{2e}, and W^E_{3e}. For attention pruning, we apply Eq. 4 to the output projection matrix W^O (Eq. 2.1) and average the scores of all neurons within heads in a GQA group to obtain the score for each group. We uniformly sample 16K training examples as calibration data to compute sensitivity scores.

Distillation. After pruning, we distill the model using the original full model as the teacher to recover performance. The student model is optimized using a gradient-based method (Loshchilov & Hutter, 2017):

$$W \leftarrow W - \eta \nabla_W L_{\text{KD}}^{\text{MoE}}(x; W), \tag{5}$$

where η is the learning rate. To reduce computational overhead, we apply early stopping once performance improvements begin to plateau, rather than training each intermediate model to full convergence. In practice, distillation at intermediate stages consumes only 30-35% of the total training steps (Figure 2).

Multi-stage Schedule. We use two stages (T=2) for Phi-mini-MoE, and for the more aggressively compressed Phi-tiny-MoE, we use three stages (T=3).

To ensure balanced pruning across stages, we follow a geometric compression schedule. Given a target overall compression ratio α, we reduce the model size at each stage by a factor of approximately $\alpha^{1/T}$. The full architectural specifications for each intermediate model are provided in Appendix A.3. This progressive strategy allows each intermediate model to retain sufficient capacity for effective knowledge transfer, resulting in a smoother and more stable transition to the final compact model.

4 Experiments

We compress the pre-trained Phi-3.5-MoE model (Abdin et al., 2024; Liu et al., 2024b) to obtain Phi-mini-MoE and Phi-tiny-MoE base models through continual pre-training. To better evaluate on downstream tasks, we further post-train the two base models by supervised fine-tuning (SFT) to enhance the models' instruction-following capabilities, followed by Direct Preference Optimization (DPO, Rafailov et al. (2023)) to steer the model away from unwanted behavior.

Data and Training. For continual pre-training, we perform multi-stage distillation using a 400B-token subset of the Phi-3.5-MoE (Abdin et al., 2024; Liu et al., 2024b) pre-training corpus, leveraging the top-8 logits predicted by the Phi-3.5-MoE teacher. For post-training, we apply SFT and DPO using the GRIN-MoE (Liu et al., 2024b) post-training corpus[2]. During SFT, we adopt a top-8 logits distillation objective using the post-trained GRIN-MoE as the teacher. The training hyperparameter configurations are provided in the Appendix A.5.

Evaluation. We evaluate our models against other similarly-sized models, including both MoE and dense architectures. For MoE models, we include Qwen 1.5 MoE (Team, 2024), DeepSeek V2 Lite (DeepSeek-AI et al., 2024a), OL-MoE (Muennighoff et al., 2024), and Granite 3.0 (Granite Team, 2024). For dense models, we include the Phi-3 series (Abdin et al., 2024), Llama 3 series (Dubey et al., 2024), Qwen 2.5 series (Yang et al., 2024a), and Gemma 3 series (Team et al., 2025).

We evaluate these models across a diverse set of downstream tasks. For commonsense and knowledge assessment, we employ MMLU (Hendrycks et al., 2021), MMLU-pro (Wang et al., 2024), Bigbench-Hard (Suzgun et al., 2023), Arc-Challenge (Clark et al., 2018), HellaSwag (Zellers et al., 2019), OpenbookQA (Mihaylov et al., 2018), PIQA (Bisk et al., 2020), BoolQ (Clark et al., 2019), and Winogrande (Sakaguchi et al., 2020). To evaluate coding capabilities, we include HumanEval (Chen et al., 2021) and MBPP (Austin et al., 2021), while for reasoning and mathematical tasks, we utilize GSM8K (Cobbe et al., 2021) and GPQA (Rein et al., 2023). Additionally, we report MT-bench (Zheng et al., 2023) scores to assess general instruction-following abilities. We provide the detailed settings for evaluations in Appendix A.6.

4.1 Performance of Compressed Models

Table 1 shows the evaluation results comparing the post-trained Phi-mini-MoE and Phi-tiny-MoE against other MoE and dense models of comparable sizes. More benchmark results are shown in Table 4, and the performance of the pretrained models is reported in Table 5.

[2]We follow GRIN-MoE's post-training setup for its simplicity, as Phi-3.5-MoE employs more complexity to build up long-context and multilingual capabilities.

Table 1: Comparison of Phi-mini-MoE and Phi-tiny-MoE against other models.

Model	# Total param	# Act. param	MMLU	MMLU pro	BBH	Arc-C (chat)	Human-eval	GSM8K	MT-bench
Mixture-of-Experts (MoE) Models									
Phi-3.5-MoE	42B	6.6B	78.36	59.38	63.93	91.38	81.70	87.87	8.34
Qwen 1.5 MoE	14B	2.7B	60.73	26.49	42.65	67.24	46.30	53.07	6.55
DeepSeek V2 Lite	16B	2.4B	56.69	17.89	36.30	61.09	54.40	63.23	6.82
OL-MoE	6.9B	1.3B	54.27	20.87	38.00	55.63	37.80	71.49	6.60
Granite 3.0 MoE	3.4B	0.8B	50.06	4.82	39.65	56.06	51.80	60.12	6.91
Dense Models									
LLaMA 3.1 8B	8B	8B	68.71	45.28	50.86	82.42	69.50	84.84	8.03
Qwen 2.5 7B	7.6B	7.6B	73.47	56.24	53.74	88.82	81.70	84.84	8.34
Phi-3-small	7.4B	7.4B	75.35	52.06	62.07	84.30	70.10	84.84	8.03
Gemma 3 4B	4.3B	4.3B	59.49	40.13	49.45	75.85	67.10	78.92	8.28
Phi-3-mini	3.8B	3.8B	69.94	45.65	54.94	85.58	72.60	84.61	7.46
LLaMA 3.2 3B	3.2B	3.2B	61.73	36.70	45.46	75.77	52.40	77.41	7.46
Qwen 2.5 3B	3.1B	3.1B	65.06	41.00	46.61	80.20	73.80	76.57	7.60
Gemma 3 1B	1B	1B	40.80	14.70	34.80	37.46	41.50	41.77	6.67
LLaMA 3.2 1B	1.2B	1.2B	46.30	18.67	35.18	49.91	35.40	44.96	5.23
Our Models									
Phi-mini-MoE	7.6B	2.4B	70.68	49.68	55.27	84.91	73.80	84.89	7.59
Phi-tiny-MoE	3.8B	1.1B	60.83	36.34	45.58	76.37	58.50	78.47	7.05

Our compressed models demonstrate strong performance while maintaining parameter efficiency. Phi-mini-MoE matches or outperforms Phi-3-mini across tasks with only two-thirds of its activated parameters, and achieves performance on par with Phi-3-small on ARC-Challenge, HumanEval, and GSM8K using just one-third of the activation size. Compared to other public dense models with similar total parameter counts, Phi-mini-MoE outperforms LLaMA 3.1 8B on most benchmarks and surpasses Qwen 2.5 7B on BBH, HellaSwag, and OpenBookQA, while using only one-third of their activated parameters. Among public MoE models, Phi-mini-MoE outperforms Qwen 1.5 MoE and DeepSeek V2 Lite MoE, while having less total parameters. Phi-tiny-MoE also shows strong results, outperforming OL-MoE and Granite 3.0 MoE with similar activation parameter counts and achieving performance comparable to LLaMA 3.2 3B at similar total parameter counts.

We note that the Phi-mini-MoE and Phi-tiny-MoE base models prior to post-training maintain similar relative performance compared to their respective baselines (Table 5), confirming that SlimMoE effectively preserves the model's capabilities.

4.2 Comparing Multi-stage with One-stage and Iterative Pruning

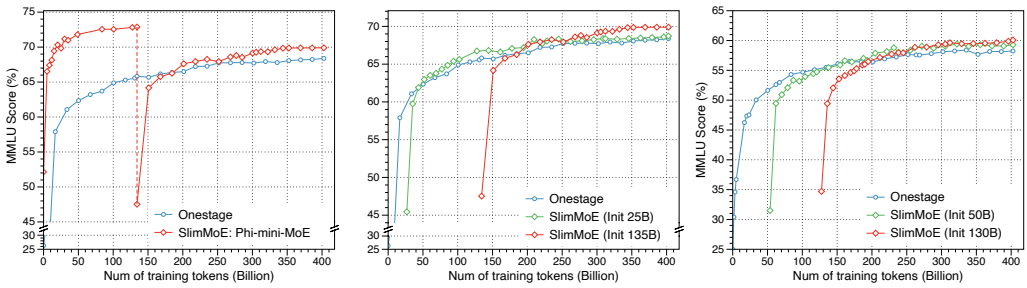
We compare SlimMoE with the conventional one-stage baseline (Muralidharan et al. (2024)) and iterative pruning (Liang et al. (2023) without the layerwise distillation objective) to demonstrate the effectiveness of our multi-stage design.

To ensure fair comparison, we control the total number of training tokens throughout the process to match between baselines. For iterative pruning, we set the pruning schedule to reach the target size at the same number of tokens as SlimMoE's initialization stages (stages before the final stage). As shown in Table 2, our multi-stage approach consistently outperforms the one-stage method across all evaluated benchmarks. On Phi-tiny-MoE, our approach achieves similar performance to iterative pruning despite the latter's much higher computational costs, as iterative pruning requires loading the full-size model at the pruning stage, resulting in approximately 2.4× computation time during initialization phase.

Figure 2 (a) illustrates how the MMLU scores evolve throughout the compression process for Phi-mini-MoE, while the corresponding plot for Phi-tiny-MoE is shown in Figure 1 (b). We can observe that the one-stage approach causes model collapse at the initial pruning step, whereas SlimMoE yields a smaller drop. The gradual transition through intermediate model sizes enables the knowledge distillation process to be more effective at each stage, resulting in superior overall performance in the final compressed model. While the multi-stage approach incurs additional computation during the initialization phases, it reaches the final convergence performance of the one-stage baseline earlier, resulting in reduced computation time to achieve the same performance: 0.74× for Phi-mini-MoE and 0.91× for Phi-tiny-MoE.

Table 2: Performance comparison of multi-stage, iterative and one-stage pruning approaches. Models evaluated here are pretrained version.

Model Type	Approach	MMLU	Arc-C	WinoGrande	HellaSwag	GSM8K
Phi-mini-MoE	One-Stage	68.58	60.84	70.85	75.93	74.52
	Multi-Stage	**69.87**	**62.29**	**75.85**	**76.03**	**77.48**
Phi-tiny-MoE	One-Stage	58.40	56.99	70.80	66.24	62.89
	Iterative	60.05	56.71	**72.20**	**67.52**	69.60
	Multi-Stage	**60.08**	**57.68**	71.90	67.33	**69.90**



(a) (b) (c)

Figure 2: MMLU performance analysis across compression stages. (a) Comparison between SlimMoE and one-stage approach on Phi-mini-MoE. (b) Impact of initialization token counts on Phi-mini-MoE. (c) Impact of initialization token counts on Phi-tiny-MoE.

4.3 When to Stop Previous Stages

A critical question for the proposed SlimMoE framework is determining the timing to stop current stage and proceed to the next. To address this question, we experiment with different token allocations for the initialization phases (where "initialization" refers to all training conducted before the final stage). As illustrated in Figure 2 (b) and (c), we compare both shorter and longer initialization approaches. For Phi-mini-MoE, we evaluate initializations of 25B and 135B tokens, while for Phi-tiny-MoE, we test 50B and 130B tokens. We observe that longer initialization consistently leads to better final performance on both model sizes. This indicates that extended earlier stages facilitate more effective knowledge transfer to subsequent stages. In practice, we recommend to initiate the next stage when the performance improvement in the current stage becomes minimal.

4.4 Are MoE Models Easier to Prune than Dense Models?

In this subsection, we investigate whether MoE architecture is more robust to pruning compared to dense model by conducting one-stage prune-and-distill on both MoE and dense architectures. For a fair comparison, we choose Phi-3-medium as the dense counterpart because it achieves performance comparable to Phi-3.5-MoE, and both models adopt the same pre-training data sources. We then compress Phi-3-medium based on the compression ratio of the activated parameters of our MoE models: 35% for Phi-mini-MoE and 16% for Phi-tiny-MoE. Detailed architecture configurations can be found in Appendix A.3.

Figure 3 presents the MMLU scores for both model types across different numbers of training tokens. The results show a consistent performance advantage for pruned MoE models over their dense counterparts. The improvements become even more pronounced at the more aggressive 16% compression level. These findings suggest that MoE architectures may indeed be more amenable to pruning, potentially due to their inherent sparse activation patterns and distributed knowledge representation across experts.

4.5 Pruning Architecture and Criterion

SlimMoE employs top-8 logits knowledge distillation (KD) loss for computing sensitivity scores to slim all experts. To justify this design choice, we conduct comparative experiments

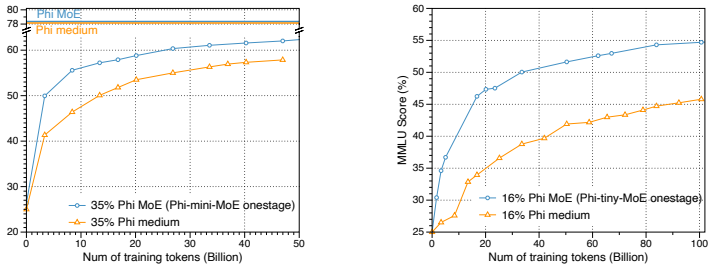


Figure 3: Performance of pruned Phi-3.5-MoE versus pruned Phi-3-medium. Left: pruned ratio 35%; Right: pruned ratio 16%.

with various pruning approaches for MoE layers. We evaluate five methods: (1) Expert slimming with top-8 KD loss (2) Expert slimming with causal language modeling loss (CLM) (3) Expert pruning based on sensitivity scores computed from KD loss (4) Expert pruning based on activation frequency (Muzio et al., 2024) (5) M-SMoE: Merging experts based on routing logits (Li et al., 2024).

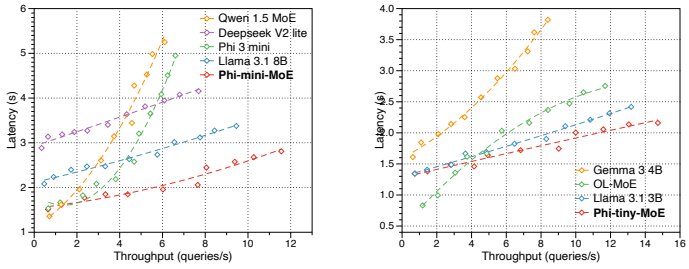
Table 3: Ablation studies on pruning architecture and criterion.

Prune Ratio	Expert Slimming (KD)	Expert Slimming (CLM)	Prune Expert (KD)	Prune Expert (freq) (Muzio et al., 2024)	M-SMoE (Li et al., 2024)
50%	**63.76**	59.30	53.41	48.59	38.54
25%	**43.25**	37.52	30.38	31.87	25.50

As shown in Table 3, using KD loss consistently outperforms CLM loss across different pruning ratios. This improvement likely stems from the teacher model's ability to mitigate noise in the training data. Our results also show that expert slimming outperforms expert pruning, suggesting that all experts contain specialized knowledge, and slimming them better preserves this knowledge compared to removing entire experts. We include studies on expert diversity in Appendix A.8. Merging experts similarly proves ineffective, potentially due to the increased size and heterogeneous weight distributions within each expert.

4.6 Inference cost

We conduct inference speed profiling for our compressed models and compare them with models of similar performance or activated parameter counts in Figure 4. We collect the inference metrics using the inference benchmarking scripts implemented by DeepSpeed[3]. More details of the profiling can be found in Appendix A.7. The results demonstrate that our compressed models achieve lower latency and comparable or higher throughput across different client loads. Notably, Phi-mini-MoE and Phi-tiny-MoE maintain these computational efficiency advantages while yielding competitive performance.



(a) Phi-mini-MoE (b) Phi-tiny-MoE

Figure 4: Latency versus throughput comparison across models under varying client loads.

[3]https://github.com/deepspeedai/DeepSpeedExamples/tree/master/benchmarks/inference/mii

5 Discussion

MoE Compression. Previous MoE compression techniques can be broadly categorized into three approaches: expert-level pruning, expert merging, and unstructured expert slimming. Expert-level pruning methods (Muzio et al., 2024; Lu et al., 2024) identify and remove redundant experts. Muzio et al. (2024) uses router logits and activation count to identify redundant experts, leading to large performance drop with high compression ratio (Section 4.5). Lu et al. (2024) enumerates all possible expert combinations for each layer to find the one with lowest reconstruction loss, which can becomes computationally prohibitive for large models with many experts. Expert merging techniques (Liu et al., 2024a; Li et al., 2024) attempt to preserve knowledge by combining experts rather than removing them. Li et al. (2024) aligns neurons across experts and then merges them based on router information, but does not work well in our settings. Liu et al. (2024a) focuses on task-specific settings and employs evolutionary search to identify merging weights. This approach requires hundreds of iterations to converge, making it substantially more expensive than our pruning method. Additionally, determining an appropriate metric for task-agnostic setting is nontrivial. He et al. (2025) and Xie et al. (2024) study unstructured expert slimming, which cannot leads to efficiency without specified hardware. Some other works consider techniques besides pruning for compressing MoE, operating on different dimensions and are complementary to SlimMoE. Yang et al. (2024b) first employ Layer-wise expert-level non-uniform pruning, then apply Singular Value Decomposition to further compress the remaining experts. Kim et al. (2023) utilize quantization to MoE models, reducing memory through lower numerical precision rather than architectural changes.

Dense Model Compression. Dense model compression has been extensively studied in recent literature (Xia et al., 2024; Muralidharan et al., 2024; Men et al., 2024). For instance, ShortGPT (Men et al., 2024) and LaCo (Yang et al., 2024c) propose to remove entire layers. Other works (Xia et al., 2024; Ashkboos et al., 2024; Ma et al., 2023) introduce various strategies for compressing width dimensions. Minitron (Muralidharan et al., 2024) represents a notable work that shares similarities with our approach. They focus on dense models and employ one-shot pruning and distillation method on Nemotron 4 15B (Parmar et al., 2024) to produce Minitron 8B, followed by further pruning to 4B, which is a two-stage compression pipeline. However, their comparison with single-stage approaches does not account for the tokens used in the initial stage. In contrast, SlimMoE introduces a principled multi-stage approach for MoE models. We systematically studies the multi-stage approach by experimenting with different stage lengths and numbers, demonstrating that multi-stage outperforms one-stage approaches under the same token budget.

Computational Cost for the Multi-stage Approach. While our experiments on Phi-3.5-MoE requires 400B tokens for performance recovery, the computational overhead remains manageable due to our strategic token allocation. Since most tokens are trained in the final stage when the model is heavily pruned, the overall computational cost is substantially reduced compared to training target-size models from scratch, which would require the full 4T tokens. SlimMoE also demonstrates robustness to varying token budgets, enabling users to adjust computational resources based on their constraints. To illustrate this flexibility, we provide additional results under a reduced training budget of 90B tokens in Appendix A.4.

6 Conclusion

In this paper, we presented SlimMoE, a multi-stage framework for compressing large MoE models by high ratios. Using SlimMoE, we compressed Phi 3.5-MoE to create Phi-mini-MoE and Phi-tiny-MoE using only 10% of the original pretraining data. These models significantly outperform both MoE and dense models with similar activated parameter counts. Our experiments demonstrate the effectiveness of the multi-stage approach and highlight the importance of preserving knowledge across experts through expert slimming. Notably, while our evaluation focuses on Phi-series models, SlimMoE is architecturally agnostic, making it broadly applicable to other MoE architectures. To the best of our knowledge, this is the first work to prune large MoE models at such high ratios (<20% of original parameters) while achieving state-of-the-art performance.

Acknowledgment

We would like to thank Shuohang Wang for the helpful discussion on distillation.

References

Marah Abdin, Jyoti Aneja, Hany Awadalla, Ahmed Awadallah, Ammar Ahmad Awan, Nguyen Bach, Amit Bahree, Arash Bakhtiari, Jianmin Bao, Harkirat S. Behl, Alon Benhaim, Misha Bilenko, Johan Bjorck, Sébastien Bubeck, Martin Cai, Qin Cai, Vishrav Chaudhary, Dong Chen, Dongdong Chen, Weizhu Chen, Yen-Chun Chen, Yi-Ling Chen, Hao Cheng, Parul Chopra, Xiyang Dai, Matthew Dixon, Ronen Eldan, Victor Fragoso, Jianfeng Gao, Mei Gao, Min Gao, Amit Garg, Allie Del Giorno, Abhishek Goswami, Suriya Gunasekar, Emman Haider, Junheng Hao, Russell J. Hewett, Wenxiang Hu, Jamie Huynh, Dan Iter, Sam Ade Jacobs, Mojan Javaheripi, Xin Jin, Nikos Karampatziakis, Piero Kauffmann, Mahoud Khademi, Dongwoo Kim, Young Jin Kim, Lev Kurilenko, James R. Lee, Yin Tat Lee, Yuanzhi Li, Yunsheng Li, Chen Liang, Lars Liden, Xihui Lin, Zeqi Lin, Ce Liu, Liyuan Liu, Mengchen Liu, Weishung Liu, Xiaodong Liu, Chong Luo, Piyush Madan, Ali Mahmoudzadeh, David Majercak, Matt Mazzola, Caio César Teodoro Mendes, Arindam Mitra, Hardik Modi, Anh Nguyen, Brandon Norick, Barun Patra, Daniel Perez-Becker, Thomas Portet, Reid Pryzant, Heyang Qin, Marko Radmilac, Liliang Ren, Gustavo de Rosa, Corby Rosset, Sambudha Roy, Olatunji Ruwase, Olli Saarikivi, Amin Saied, Adil Salim, Michael Santacroce, Shital Shah, Ning Shang, Hiteshi Sharma, Yelong Shen, Swadheen Shukla, Xia Song, Masahiro Tanaka, Andrea Tupini, Praneetha Vaddamanu, Chunyu Wang, Guanhua Wang, Lijuan Wang, Shuohang Wang, Xin Wang, Yu Wang, Rachel Ward, Wen Wen, Philipp Witte, Haiping Wu, Xiaoxia Wu, Michael Wyatt, Bin Xiao, Can Xu, Jiahang Xu, Weijian Xu, Jilong Xue, Sonali Yadav, Fan Yang, Jianwei Yang, Yifan Yang, Ziyi Yang, Donghan Yu, Lu Yuan, Chengruidong Zhang, Cyril Zhang, Jianwen Zhang, Li Lyna Zhang, Yi Zhang, Yue Zhang, Yunan Zhang, and Xiren Zhou. Phi-3 technical report: A highly capable language model locally on your phone. *CoRR*, abs/2404.14219, 2024.

Joshua Ainslie, James Lee-Thorp, Michiel de Jong, Yury Zemlyanskiy, Federico Lebrón, and Sumit Sanghai. GQA: training generalized multi-query transformer models from multi-head checkpoints. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023, Singapore, December 6-10, 2023*, pp. 4895–4901. Association for Computational Linguistics, 2023. doi: 10.18653/V1/2023.EMNLP-MAIN.298. URL https://doi.org/10.18653/v1/2023.emnlp-main.298.

Saleh Ashkboos, Maximilian L. Croci, Marcelo Gennari Do Nascimento, Torsten Hoefler, and James Hensman. Slicegpt: Compress large language models by deleting rows and columns. In *ICLR*. OpenReview.net, 2024.

Jacob Austin, Augustus Odena, Maxwell I. Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie J. Cai, Michael Terry, Quoc V. Le, and Charles Sutton. Program synthesis with large language models. *CoRR*, abs/2108.07732, 2021.

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

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Pondé de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Joshua Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish,

Ilya Sutskever, and Wojciech Zaremba. Evaluating large language models trained on code. *CoRR*, abs/2107.03374, 2021.

Tianyu Chen, Shaohan Huang, Yuan Xie, Binxing Jiao, Daxin Jiang, Haoyi Zhou, Jianxin Li, and Furu Wei. Task-specific expert pruning for sparse mixture-of-experts. *CoRR*, abs/2206.00277, 2022.

Jang Hyun Cho and Bharath Hariharan. On the efficacy of knowledge distillation. In *ICCV*, pp. 4793–4801. IEEE, 2019.

Mohammed Nowaz Rabbani Chowdhury, Meng Wang, Kaoutar El Maghraoui, Naigang Wang, Pin-Yu Chen, and Christopher Carothers. A provably effective method for pruning experts in fine-tuned sparse mixture-of-experts. *arXiv preprint arXiv:2405.16646*, 2024.

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 (1)*, pp. 2924–2936. Association for Computational Linguistics, 2019.

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. *CoRR*, abs/1803.05457, 2018.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems. *CoRR*, abs/2110.14168, 2021.

DeepSeek-AI, Aixin Liu, Bei Feng, Bin Wang, Bingxuan Wang, Bo Liu, Chenggang Zhao, Chengqi Deng, Chong Ruan, Damai Dai, Daya Guo, Dejian Yang, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, Hao Zhang, Hanwei Xu, Hao Yang, Haowei Zhang, Honghui Ding, Huajian Xin, Huazuo Gao, Hui Li, Hui Qu, J. L. Cai, Jian Liang, Jianzhong Guo, Jiaqi Ni, Jiashi Li, Jin Chen, Jingyang Yuan, Junjie Qiu, Junxiao Song, Kai Dong, Kaige Gao, Kang Guan, Lean Wang, Lecong Zhang, Lei Xu, Leyi Xia, Liang Zhao, Liyue Zhang, Meng Li, Miaojun Wang, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Mingming Li, Ning Tian, Panpan Huang, Peiyi Wang, Peng Zhang, Qihao Zhu, Qinyu Chen, Qiushi Du, R. J. Chen, R. L. Jin, Ruiqi Ge, Ruizhe Pan, Runxin Xu, Ruyi Chen, S. S. Li, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shaoqing Wu, Shengfeng Ye, Shirong Ma, Shiyu Wang, Shuang Zhou, Shuiping Yu, Shunfeng Zhou, Size Zheng, Tao Wang, Tian Pei, Tian Yuan, Tianyu Sun, W. L. Xiao, Wangding Zeng, Wei An, Wen Liu, Wenfeng Liang, Wenjun Gao, Wentao Zhang, X. Q. Li, Xiangyue Jin, Xianzu Wang, Xiao Bi, Xiaodong Liu, Xiaohan Wang, Xiaojin Shen, Xiaokang Chen, Xiaosha Chen, Xiaotao Nie, and Xiaowen Sun. Deepseek-v2: A strong, economical, and efficient mixture-of-experts language model. *CoRR*, abs/2405.04434, 2024a.

DeepSeek-AI, Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Daya Guo, Dejian Yang, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Han Bao, Hanwei Xu, Haocheng Wang, Haowei Zhang, Honghui Ding, Huajian Xin, Huazuo Gao, Hui Li, Hui Qu, J. L. Cai, Jian Liang, Jianzhong Guo, Jiaqi Ni, Jiashi Li, Jiawei Wang, Jin Chen, Jingchang Chen, Jingyang Yuan, Junjie Qiu, Junlong Li, Junxiao Song, Kai Dong, Kai Hu, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Lei Xu, Leyi Xia, Liang Zhao, Litong Wang, Liyue Zhang, Meng Li, Miaojun Wang, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Mingming Li, Ning Tian, Panpan Huang, Peiyi Wang, Peng Zhang, Qiancheng Wang, Qihao Zhu, Qinyu Chen, Qiushi Du, R. J. Chen, R. L. Jin, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, Runxin Xu, Ruoyu Zhang, Ruyi Chen, S. S. Li, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shaoqing Wu, Shengfeng Ye, Shengfeng Ye, Shirong Ma, Shiyu Wang, Shuang Zhou, Shuiping Yu, Shunfeng Zhou, Shuting Pan, T. Wang, Tao Yun, Tian Pei, Tianyu Sun, W. L. Xiao, and Wangding Zeng. Deepseek-v3 technical report. *CoRR*, abs/2412.19437, 2024b.

Tim Dettmers, Mike Lewis, Sam Shleifer, and Luke Zettlemoyer. 8-bit optimizers via block-wise quantization. *arXiv preprint arXiv:2110.02861*, 2021.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, Arun Rao, Aston Zhang, Aurélien Rodriguez, Austen Gregerson, Ava Spataru, Baptiste Rozière, Bethany Biron, Binh Tang, Bobbie Chern, Charlotte Caucheteux, Chaya Nayak, Chloe Bi, Chris Marra, Chris McConnell, Christian Keller, Christophe Touret, Chunyang Wu, Corinne Wong, Cristian Canton Ferrer, Cyrus Nikolaidis, Damien Allonsius, Daniel Song, Danielle Pintz, Danny Livshits, David Esiobu, Dhruv Choudhary, Dhruv Mahajan, Diego Garcia-Olano, Diego Perino, Dieuwke Hupkes, Egor Lakomkin, Ehab AlBadawy, Elina Lobanova, Emily Dinan, Eric Michael Smith, Filip Radenovic, Frank Zhang, Gabriel Synnaeve, Gabrielle Lee, Georgia Lewis Anderson, Graeme Nail, Grégoire Mialon, Guan Pang, Guillem Cucurell, Hailey Nguyen, Hannah Korevaar, Hu Xu, Hugo Touvron, Iliyan Zarov, Imanol Arrieta Ibarra, Isabel M. Kloumann, Ishan Misra, Ivan Evtimov, Jade Copet, Jaewon Lee, Jan Geffert, Jana Vranes, Jason Park, Jay Mahadeokar, Jeet Shah, Jelmer van der Linde, Jennifer Billock, Jenny Hong, Jenya Lee, Jeremy Fu, Jianfeng Chi, Jianyu Huang, Jiawen Liu, Jie Wang, Jiecao Yu, Joanna Bitton, Joe Spisak, Jongsoo Park, Joseph Rocca, Joshua Johnstun, Joshua Saxe, Junteng Jia, Kalyan Vasuden Alwala, Kartikeya Upasani, Kate Plawiak, Ke Li, Kenneth Heafield, Kevin Stone, and et al. The llama 3 herd of models. *CoRR*, abs/2407.21783, 2024.

Gongfan Fang, Hongxu Yin, Saurav Muralidharan, Greg Heinrich, Jeff Pool, Jan Kautz, Pavlo Molchanov, and Xinchao Wang. Maskllm: Learnable semi-structured sparsity for large language models. In Amir Globersons, Lester Mackey, Danielle Belgrave, Angela Fan, Ulrich Paquet, Jakub M. Tomczak, and Cheng Zhang (eds.), *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, 2024. URL http://papers.nips.cc/paper_files/paper/2024/hash/0e9a05f5ce62284c91e4a33498899124-Abstract-Conference.html.

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

Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. The language model evaluation harness, 07 2024. URL https://zenodo.org/records/12608602.

IBM Granite Team. Granite 3.0 language models, 2024.

Song Han, Huizi Mao, and William J Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149*, 2015a.

Song Han, Jeff Pool, John Tran, and William J Dally. Learning both weights and connections for efficient neural networks. *arXiv preprint arXiv:1506.02626*, 2015b.

Shwai He, Daize Dong, Liang Ding, and Ang Li. Towards efficient mixture of experts: A holistic study of compression techniques, 2025. URL https://arxiv.org/abs/2406.02500.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. In *ICLR*. OpenReview.net, 2021.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

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, Gianna Lengyel, Guillaume Bour, Guillaume Lample, Lélio Renard Lavaud, Lucile Saulnier, Marie-Anne Lachaux, Pierre Stock, Sandeep Subramanian, Sophia Yang, Szymon Antoniak, Teven Le Scao, Théophile Gervet, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. Mixtral of experts. *CoRR*, abs/2401.04088, 2024.

Keller Jordan, Yuchen Jin, Vlado Boza, Jiacheng You, Franz Cesista, Laker Newhouse, and Jeremy Bernstein. Muon: An optimizer for hidden layers in neural networks, 2024. URL https://kellerjordan.github.io/posts/muon/.

Young Jin Kim, Raffy Fahim, and Hany Hassan Awadalla. Mixture of quantized experts (moqe): Complementary effect of low-bit quantization and robustness. *arXiv preprint arXiv:2310.02410*, 2023.

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.

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*. OpenReview.net, 2024.

Chen Liang, Haoming Jiang, Zheng Li, Xianfeng Tang, Bing Yin, and Tuo Zhao. Homodistil: Homotopic task-agnostic distillation of pre-trained transformers. In *ICLR*. OpenReview.net, 2023.

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. *CoRR*, abs/2407.00945, 2024a.

Jiawei Liu, Chunqiu Steven Xia, Yuyao Wang, and Lingming Zhang. Is your code generated by chatGPT really correct? rigorous evaluation of large language models for code generation. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023a. URL https://openreview.net/forum?id=1qvx610Cu7.

Liyuan Liu, Chengyu Dong, Xiaodong Liu, Bin Yu, and Jianfeng Gao. Bridging discrete and backpropagation: Straight-through and beyond. *Advances in Neural Information Processing Systems*, 36:12291–12311, 2023b.

Liyuan Liu, Jianfeng Gao, and Weizhu Chen. Sparse backpropagation for moe training. *arXiv preprint arXiv:2310.00811*, 2023c.

Liyuan Liu, Young Jin Kim, Shuohang Wang, Chen Liang, Yelong Shen, Hao Cheng, Xiaodong Liu, Masahiro Tanaka, Xiaoxia Wu, Wenxiang Hu, Vishrav Chaudhary, Zeqi Lin, Chengruidong Zhang, Jilong Xue, Hany Awadalla, Jianfeng Gao, and Weizhu Chen. GRIN: gradient-informed moe. *CoRR*, abs/2409.12136, 2024b.

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

Christos Louizos, Max Welling, and Diederik P Kingma. Learning sparse neural networks through l_0 regularization. *arXiv preprint arXiv:1712.01312*, 2017.

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 (1)*, pp. 6159–6172. Association for Computational Linguistics, 2024.

Xinyin Ma, Gongfan Fang, and Xinchao Wang. Llm-pruner: On the structural pruning of large language models. In *NeurIPS*, 2023.

Xin Men, Mingyu Xu, Qingyu Zhang, Bingning Wang, Hongyu Lin, Yaojie Lu, Xianpei Han, and Weipeng Chen. Shortgpt: Layers in large language models are more redundant than you expect. *CoRR*, abs/2403.03853, 2024.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. Can a suit of armor conduct electricity? A new dataset for open book question answering. In *EMNLP*, pp. 2381–2391. Association for Computational Linguistics, 2018.

Seyed-Iman Mirzadeh, Mehrdad Farajtabar, Ang Li, Nir Levine, Akihiro Matsukawa, and Hassan Ghasemzadeh. Improved knowledge distillation via teacher assistant. In *AAAI*, pp. 5191–5198. AAAI Press, 2020.

Pavlo Molchanov, Stephen Tyree, Tero Karras, Timo Aila, and Jan Kautz. Pruning convolutional neural networks for resource efficient inference. *arXiv preprint arXiv:1611.06440*, 2016.

Pavlo Molchanov, Arun Mallya, Stephen Tyree, Iuri Frosio, and Jan Kautz. Importance estimation for neural network pruning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 11264–11272, 2019.

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. *arXiv preprint arXiv:2409.02060*, 2024.

Saurav Muralidharan, Sharath Turuvekere Sreenivas, Raviraj Joshi, Marcin Chochowski, Mostofa Patwary, Mohammad Shoeybi, Bryan Catanzaro, Jan Kautz, and Pavlo Molchanov. Compact language models via pruning and knowledge distillation. In *NeurIPS*, 2024.

Alexandre Muzio, Alex Sun, and Churan He. Seer-moe: Sparse expert efficiency through regularization for mixture-of-experts. *CoRR*, abs/2404.05089, 2024.

Jupinder Parmar, Shrimai Prabhumoye, Joseph Jennings, Mostofa Patwary, Sandeep Subramanian, Dan Su, Chen Zhu, Deepak Narayanan, Aastha Jhunjhunwala, Ayush Dattagupta, Vibhu Jawa, Jiwei Liu, Ameya Mahabaleshwarkar, Osvald Nitski, Annika Brundyn, James Maki, Miguel Martinez, Jiaxuan You, John Kamalu, Patrick LeGresley, Denys Fridman, Jared Casper, Ashwath Aithal, Oleksii Kuchaiev, Mohammad Shoeybi, Jonathan M. Cohen, and Bryan Catanzaro. Nemotron-4 15b technical report. *CoRR*, abs/2402.16819, 2024.

Hao Peng, Xin Lv, Yushi Bai, Zijun Yao, Jiajie Zhang, Lei Hou, and Juanzi Li. Pre-training distillation for large language models: A design space exploration. *CoRR*, abs/2410.16215, 2024. doi: 10.48550/ARXIV.2410.16215. URL https://doi.org/10.48550/arXiv.2410.16215.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. *Advances in Neural Information Processing Systems*, 36:53728–53741, 2023.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. GPQA: A graduate-level google-proof q&a benchmark. *CoRR*, abs/2311.12022, 2023.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. Winogrande: An adversarial winograd schema challenge at scale. In *AAAI*, pp. 8732–8740. AAAI Press, 2020.

Victor Sanh, Thomas Wolf, and Alexander Rush. Movement pruning: Adaptive sparsity by fine-tuning. *Advances in neural information processing systems*, 33:20378–20389, 2020.

Noam Shazeer. Glu variants improve transformer. *arXiv preprint arXiv:2002.05202*, 2020.

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *arXiv preprint arXiv:1701.06538*, 2017.

Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc V. Le, Ed H. Chi, Denny Zhou, and Jason Wei. Challenging big-bench tasks and whether chain-of-thought can solve them. In *ACL (Findings)*, pp. 13003–13051. Association for Computational Linguistics, 2023.

Gemma Team, Aishwarya Kamath, Johan Ferret, Shreya Pathak, Nino Vieillard, Ramona Merhej, Sarah Perrin, Tatiana Matejovicova, Alexandre Ramé, Morgane Rivière, et al. Gemma 3 technical report. *arXiv preprint arXiv:2503.19786*, 2025.

Qwen Team. Qwen1.5-moe: Matching 7b model performance with 1/3 activated parameters", February 2024. URL https://qwenlm.github.io/blog/qwen-moe/.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, Tianle Li, Max Ku, Kai Wang, Alex Zhuang, Rongqi Fan, Xiang Yue, and Wenhu Chen. Mmlu-pro: A more robust and challenging multi-task language understanding benchmark. In *NeurIPS*, 2024.

Mengzhou Xia, Zexuan Zhong, and Danqi Chen. Structured pruning learns compact and accurate models. *arXiv preprint arXiv:2204.00408*, 2022.

Mengzhou Xia, Tianyu Gao, Zhiyuan Zeng, and Danqi Chen. Sheared llama: Accelerating language model pre-training via structured pruning. In *ICLR*. OpenReview.net, 2024.

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. *CoRR*, abs/2410.12013, 2024.

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. *CoRR*, abs/2412.15115, 2024a.

Cheng Yang, Yang Sui, Jinqi Xiao, Lingyi Huang, Yu Gong, Yuanlin Duan, Wenqi Jia, Miao Yin, Yu Cheng, and Bo Yuan. Moe-i^2: Compressing mixture of experts models through inter-expert pruning and intra-expert low-rank decomposition. *CoRR*, abs/2411.01016, 2024b.

Yifei Yang, Zouying Cao, and Hai Zhao. Laco: Large language model pruning via layer collapse. In *EMNLP (Findings)*, pp. 6401–6417. Association for Computational Linguistics, 2024c.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. Hellaswag: Can a machine really finish your sentence? In *ACL (1)*, pp. 4791–4800. Association for Computational Linguistics, 2019.

Qingru Zhang, Simiao Zuo, Chen Liang, Alexander Bukharin, Pengcheng He, Weizhu Chen, and Tuo Zhao. Platon: Pruning large transformer models with upper confidence bound of weight importance. In *International conference on machine learning*, pp. 26809–26823. PMLR, 2022.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. Judging llm-as-a-judge with mt-bench and chatbot arena. In *NeurIPS*, 2023.

A Appendix

A.1 Results on extended tasks.

Table 4: Extended evaluation on commonsense reasoning, QA, and code generation tasks.

Model	# Total param	# Act. param	Winograde	Hellaswag	GPQA	PIQA	OpenbookQA	BoolQ	MBPP
Mixture-of-Experts (MoE) Models									
Phi 3.5-MoE	42B	6.6B	78.61	82.31	39.90	81.72	63.20	88.93	73.30
Qwen 1.5 MoE	14B	2.7B	70.40	79.40	27.27	80.25	49.40	81.77	42.90
DeepSeek V2 Lite	16B	2.4B	70.40	74.81	25.76	79.60	46.20	83.30	59.50
OL-MoE	6.9B	1.3B	66.54	75.44	24.75	80.79	48.00	78.78	36.50
Granite 3.0 MoE	3.4B	0.8B	67.17	70.64	27.78	78.45	44.40	78.99	52.90
Dense Models									
LLaMA 3.1 8B	8B	8B	75.61	78.27	28.79	81.28	52.40	85.20	67.70
Qwen 2.5 7B	7.6B	7.6B	75.90	72.25	34.34	72.20	50.00	86.85	79.40
Phi 3 small	7.4B	7.4B	73.38	81.97	30.81	81.72	57.20	87.13	72.80
Gemma 3 4B	4.3B	4.3B	65.67	55.80	27.27	73.45	49.60	80.03	78.00
Phi 3 mini	3.8B	3.8B	76.95	76.48	26.77	81.63	53.80	85.87	72.50
LLaMA 3.2 3B	3.2B	3.2B	67.72	70.40	22.22	78.45	43.20	77.98	63.00
Qwen 2.5 3B	3.1B	3.1B	71.10	70.37	23.74	74.54	46.60	83.59	72.50
Gemma 3 1B	1B	1B	55.80	47.66	22.73	71.98	40.20	70.86	57.70
LLaMA 3.2 1B	1.2B	1.2B	62.04	61.0	21.21	75.35	36.0	57.37	33.10
Our Models									
Phi-mini-MoE	7.6B	2.4B	75.45	74.76	27.78	81.77	51.20	85.02	69.60
Phi-tiny-MoE	3.8B	1.1B	70.09	67.44	29.29	79.16	48.00	81.07	53.70

A.2 Results of pretrained models

Results of pretrained models are presented in Table 5.

Table 5: Performance comparison of pretrained models on various benchmark tasks. * We report the performance of the instruct model for Phi 3.5-MoE.

Model (pretrained)	# Total param	# Act. param	MMLU	Winograde	Arc-C	Hellaswag	BoolQ
Mixture-of-Experts (MoE) Models							
Phi 3.5-MoE*	42B	6.6B	78.36	78.61	65.52	82.31	88.93
Qwen 1.5 MoE	14B	2.7B	61.24	71.82	56.06	79.17	80.79
DeepSeek V2 Lite	16B	2.4B	58.09	75.69	56.31	79.79	79.34
OL-MoE	6.9B	1.3B	54.96	72.45	58.53	79.12	77.86
Granite 3.0 MoE	3.4B	0.8B	48.44	67.8	49.57	73.15	75.38
Dense Models							
LLaMA 3.1 8B	8B	8B	65.22	77.03	57.51	80.89	82.69
Qwen 2.5 7B	7.6B	7.6B	74.20	75.90	63.31	79.46	87.77
Gemma 3 4B	4.3B	4.3B	59.51	72.53	58.36	77.23	80.03
LLaMA 3.2 3B	3.2B	3.2B	56.07	72.14	48.38	75.46	73.30
Qwen 2.5 3B	3.1B	3.1B	65.55	71.35	57.41	74.49	83.94
Gemma 3 1B	1B	1B	26.26	61.01	39.68	62.38	64.77
LLaMA 3.2 1B	1.2B	1.2B	31.00	62.03	38.30	65.70	65.50
Our Models							
Phi-mini-MoE	7.6B	2.4B	69.87	75.85	62.29	76.03	85.14
Phi-tiny-MoE	3.8B	1.1B	60.08	71.90	57.68	67.33	80.97

A.3 Architecture of intermediate sizes and dense model

Architecture configurations for Phi-mini-MoE and Phi-tiny-MoE are shown in Table 6. Architecture configurations of intermediate sizes for Phi-mini-MoE and Phi-tiny-MoE are shown in Table 7. Architecture configurations of pruned Phi 3 medium are shown in Table 8.

Table 6: Model configuration details and parameter counts of Phi-mini-MoE and Phi-tiny-MoE.

Model	d_{model}	n_{head} (q/kv)	d_{expert}	n_{layer}	n_{expert}	top-k	# Total Param	# Act. Param
Phi-3.5-MoE	4096	32/8	6400	32	16	2	41.9B	6.6B
Phi-mini-MoE	4096	32/8	960	32	16	2	7.6B	2.4B
Phi-tiny-MoE	4096	16/4	448	32	16	2	3.8B	1.1B

Table 7: Model configuration details and parameters counts of intermediate sizes.

Model	d_{model}	n_{head}	d_{expert}	n_{layer}	n_{expert}	top-k	# Total Param	# Act. Param
Phi 3.5-MoE / GRIN-MoE	4096	32/8	6400	32	16	2	41.9B	6.6B
Phi-mini-MoE-stage1	4096	32/8	2240	32	16	2	15.7B	3.4B
Phi-mini-MoE (stage2)	4096	32/8	960	32	16	2	7.6B	2.4B
Phi-tiny-MoE-stage1	4096	24/6	2624	32	16	2	17.8B	3.3B
Phi-tiny-MoE-stage2	4096	20/5	1024	32	16	2	7.6B	1.9B
Phi-tiny-MoE (stage3)	4096	16/4	448	32	16	2	3.8B	1.1B

A.4 Results with lower training budgets.

We evaluated SlimMoE's effectiveness under reduced training budgets. We conducted experiments on Phi-mini-MoE's size using only 90B total tokens, with 25B tokens allocated to the first stage. Table 9 presents the performance comparison between our multi-stage approach and the one-stage baseline. The results demonstrate that even with a 4.4× reduction in training tokens, our multi-stage framework consistently outperforms the one-stage approach, confirming that SlimMoE's advantages persist even under resource constraints.

A.5 Training Details

Training Hyperparameters. We use different hyperparameter settings during the compression process and subsequent SFT distillation. During compression, we use the AdamW optimizer with a learning rate of 1e-4 and weight decay of 0.01. We apply cosine learning rate decay with 100 warmup steps. For SlimMoE, we apply these warmup steps at the beginning of each stage. We use a batch size of 4096 and a maximum sequence length of 4096. For SFT distillation, we maintain the same optimizer but reduce the learning rate to 2e-6 and weight decay to 1e-4. We use a batch size of 1536. All training was conducted on 64 A100 GPUs.

A.6 Evaluation Details

We use lm-evaluation-harness (Gao et al., 2024) for evaluations on all tasks except MT-bench, Humaneval and MBPP, where we follows original implementation of MT-bench and use evalplus base mode (Liu et al., 2023a) for coding tasks. We use different evaluation settings for pretrained and instruct model. For pretrained models, we use 5 shots for all tasks. For instruct models, we use 5 shots for MMLU, MMLU pro, Winograde, Hellaswag and PIQA, 0 shot for Arc-challenge with chat prompt, 0 shot cot for GPQA, 2 shots for BoolQ, 10 shots for openbookQA, 3 shots for BBH and 8 shots for GSM8K. We apply chat template when evaluating instruct models.

A.7 Computational cost

For inference speed profiling, we use vLLM as the serving backend for all evaluated models. The number of concurrent clients is set as default (1, 2, 4, 6, 8, 12, 16, 20, 24, 28, 32). The prompt of request has a mean length of 2048 tokens and the generation length is 256 tokens.

Table 8: Model Configuration Details for Phi 3 Dense Models

Model	d_{model}	n_{head}	d_{ffn}	n_{layer}	# Total Param
Phi 3 Dense	5120	40/10	17920	40	14.0B
35% Phi 3 Dense	5120	40/10	3297	40	5.0B
16% Phi 3 Dense	5120	20/5	1098	40	1.9B

Table 9: Performance comparison under reduced training budget (90B tokens total)

Method	MMLU	Winograde	Arc-C	HellaSwag
One-stage	64.11	72.34	59.13	73.23
Multi-stage	65.59	72.74	62.54	73.38

For each client load, we record both throughput total latency. We also conduct memory profiling during finetuning Phi-mini-MoE and Phi-tiny-MoE in Table 10.

Table 10: Memory Profiling Results

Model	**Optimizer**	**Peak Allocated Memory (MB)**	**Peak Reserved (MB)**
Phi-mini-MoE	8-bit Adam	59204.31	68756.00
	Muon	59759.51	61690.00
Phi-tiny-MoE	8-bit Adam	40573.62	47046.00
	Muon	40556.37	42862.00

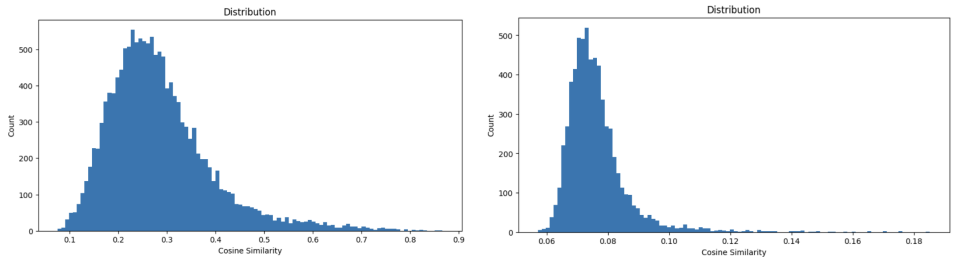
A.8 Expert similarity

We notice that in some existing works (Lu et al., 2024; Liu et al., 2024a), the performance drop observed with expert pruning is smaller than what we have observed on Phi-3.5-MoE. We attribute this discrepancy to the differences between model families. Previous studies mainly used Mixtral as their target model, where experts exhibit high similarity, possibly because of tailored initialization. This similarity could reduces the performance impact of expert pruning. In contrast, Phi-MoE-3.5 appears to distribute knowledge more heterogeneously across experts. We present our detailed study below:

We calculated the expert similarity in Mixtral 7×8B and Phi-3.5-MoE. Figure 5 presents the distribution of maximum cosine similarities between neurons across different experts within each model.

We randomly sample a pair of experts from each model. For each neuron in the first expert, we identified the neuron in the second expert with the highest cosine similarity, then plotted the distribution of these maximum similarity values across all neurons.

The results show that Mixtral exhibits substantially higher inter-expert similarity. We also observe that in Mixtral, the neurons with highest cosine similarity tend to be positioned at corresponding indices across experts. In contrast, Phi 3.5-MoE shows lower similarity and lacks this positional correspondence pattern.



(a) Mixtral 7x8B (b) Phi 3.5-MoE

Figure 5: Cosine similarity distribution of neurons in randomly sampled two experts