SEER-MoE: Sparse Expert Efficiency through Regularization for Mixture-of-Experts

Stanford CS224N Custom Project

Alex Muzio **Alex Sun** **Churan He**
Department of Computer Science
Stanford University
{alexfm, xs15, churanhe}@stanford.edu

Abstract

The advancement of deep learning has led to the emergence of Mixture-of-Experts (MoEs) models, known for their dynamic allocation of computational resources based on input. Despite their promise, MoEs face challenges, particularly in terms of memory requirements. To address this, our work introduces SEER-MoE, a novel two-stage framework for reducing both the memory footprint and compute requirements of pre-trained MoE models. The first stage involves pruning the total number of experts using a heavy-hitters counting guidance, while the second stage employs a regularization-based fine-tuning strategy to recover accuracy loss and reduce the number of activated experts during inference. Our empirical studies demonstrate the effectiveness of our method, resulting in a sparse MoEs model optimized for inference efficiency with minimal accuracy trade-offs.

1 Introduction

Recent advances in deep learning have propelled the field towards increasingly large and complex models to achieve state-of-the-art performance across a myriad of tasks. Among these, Mixture-of-Experts (MoEs) models have emerged as a promising architecture, distinguished by their ability to dynamically allocate computational resources based on the input (Fedus et al., 2021; Zoph et al., 2022; Jiang et al., 2024; et al, 2024). This paradigm, characterized by a sparse gating mechanism that routes inputs to a subset of specialized expert networks, allows for the scalable expansion of model parameters while maintaining a relatively constant computational footprint per input token.

However, the path forward for MoEs involves addressing significant challenges, particularly regarding their substantial memory requirements. The advent of very large MoE models such as Grok-1 xAI (2024), with 314B parameters distributed across 8 experts, underscores the urgency of addressing this issue. While the sparse nature of MoEs promises enhanced efficiency and scalability, the sheer size of the models introduces a new set of complexities, particularly in terms of memory footprint.

In light of these challenges, our work aims to investigate and develop pruning and fine-tuning strategies that dynamically adjust the quantity and allocation of experts in an MoE-based language model.

The main contributions of our paper is an in-depth study of Parameter Count / FLOPs for MoE models and the SEER-MoE method, a novel 2-stage approach that takes a step in the direction of reducing the memory footprint / compute requirements for pretrained MoE models. Our first stage proposes to prune the total number of experts in the model with a novel *heavy-hitters counting* guidance to reduce the memory footprint for loading the entire MoE model. Our second stage proposes an effecitve regularization-based finetuning strategy to recover the accuracy loss from previous pruning while simultanously reducing the number of activated experts during inference. The combination of both stages yields a sparse MoEs model with cheaper memory requirements while being optimized for inference efficiency, at the compensation of minimized accuracy drops.

We perform extensive empirical studies with the popular Mixtral 8x7b Jiang et al. (2024) MoEs model on both SST5 Socher et al. (2013) and MMLU Hendrycks et al. (2020) to validate the effectiveness of our method, including an in-depth ablation study to understand different design choices for each of our stage.

2 Related Work

Mixture-of-Experts (MoE) dates back at least since work from Jacobs et al. (1991), which introduce a new model architecture composed of many separate networks and each one handles a subset of the complete set of training cases. Each expert specializes in a different region of the input space. Eigen et al. (2014) extended the Mixture-of-Experts to a stacked model with multiple layers of gating and experts, and exponentially increases the number of effective experts through layers of combination. As the rapid advancement of LLM, MoE (Jiang et al., 2024; et al, 2024; xAI, 2024) gained increased popularity for it's scalability, efficiency and STOA evaluation result from various benchmarks. Most MoE architectures Fedus et al. (2021); Zoph et al. (2022); Jiang et al. (2024) includes a specific gating network that learns the optimal routing from input tokens to experts. However, the weights of the gating network stay fixed, regardless of what task is being solved. To the best of our knowledge, there is no such method that explores Top-K routing adaptation.

Sparsification aims to remove certain parts of the network. In Mixture-of-Experts models, only a few experts (top-K) chosen by the router will be activated in each layer to generate output, therefore removing unused experts can linearly reduce the model size without loss in performance. (Lu et al., 2024) introduced a heuristic search method to prune the number of experts in post-training. The method is based on the enumeration of expert combinations and choosing the target eliminating experts based on the lowest reconstruction loss. They verified their approach's effectiveness on Mixtral 8x7b (Jiang et al., 2024). However, the method has high time complexity and isn't applicable to models with large expert counts nor cross-layer pruning (Zhang et al., 2023) proposed an eviction algorithm targeting on KV cache based on "Heavy Hitters" which holds a significant role for model performance. We hypothesize that a similar strategy in expert activation counting can be applied to MoE pruning.

3 Methodology

The computation and resources used by large MoEs models (FLOPs) mainly come from two factors:

The total number of available experts at each MoE layer, denoted as M_l for layer l, and determines how much VRAM we need to completely load the model on the GPUs/TPUs.

The top-K number of experts activated, denoted as k_l for layer l, and controls how much computation is used per token.

In terms of notation, suppose we have a fine-tuning dataset $\mathcal{D}$ which consists of sample (x_i, y_i) pairs, the gating network at layer l is g_l, and the jth expert at the lth layer is denoted as e_l^j. Moreover, suppose the features collected at layer l is $f_l(x_i)$.

3.1 Parameter and Compute Scaling of MoE Transformers

Following the work of Kaplan et al. (2020), we extend the parameterization of the Transformer architecture to that of the sparse MoE architecture (more details in Appendix A). We utilize the Transformer hyperparameters together with $n_{experts}$ (number of experts per layer) and n_{topk} (number of experts activated per token). Using N to denote the number of non-embedding parameters and considering $d_{attn} = d_{ff}/4 = d_{model}$

$$N \approx 4d_{model}^2 n_{layer}(1 + 2n_{experts}) \tag{1}$$

And the FLOPs for the forward pass:

$$C_{fwd} \approx 8d_{model}^2 n_{layer}(1 + 2n_{topk}) + 2n_{layer} n_{ctx} d_{model} \tag{2}$$

It is notable that for the MoE blocks of the network, the number of parameters from the MoE Feedforward layers increase in proportion to $n_{experts}$ while the FLOPs per Token to n_{topk}. This

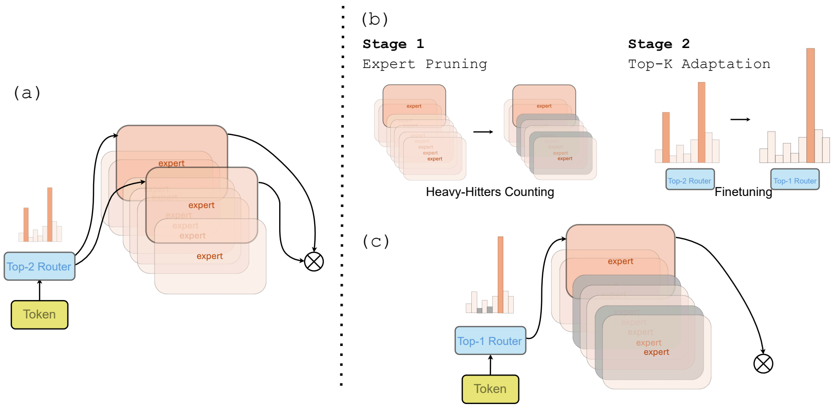

Figure 1: SEER-MoE visualized in a two-stage process. (a) The initial model with all experts and top-2 router. (b) Stage 1 involves expert pruning based on heavy-hitters counting to identify and retain the most critical experts; Stage 2 includes top-K adaptation through fine-tuning to optimize the number of active experts, culminating in a model that balances efficiency and performance. (c) SEER-MoE with expert pruning and top-K adaptation.

motivates us to think whether we can target reducing both the compute FLOPs of the model and the memory separately.

Specifically for the Mixtral 8x7B Jiang et al. (2024) model, which utilizes 2 experts per token, Expert blocks computations account for about 55% of the total FLOPS. Additionally, for a model with the same architecture but with only a single expert being activated, FLOPs reduces by 27%. This motivates us to explore whether we are able to adapt existing models to use less compute.

3.2 Expert Sparsification with Heavy-hitters Counting

Considering the large memory requirement of MoE models described and considering that the bulk of the parameters belong to the Expert layers, we propose reducing the total number of experts M_l at each layer l. We do so, by reducing the number of expert models. We propose to carry out this investigation with a data-driven strategy.

For a MoEs model, an expert e_l^j is activated for a token if its corresponding router logit $g_l(f_l(x_i))^j$ is ranked in the top-K after softmax. Specifically, we denote this selection process with a function $\gamma(.)$:

$$\gamma(e_l^j, x_i) = 1 \text{ if } j \in \text{ArgTopK}(\text{softmax}(g_l(f_l(x_i))), K), \tag{3}$$

$$\gamma(e_l^j, x_i) = 0 \text{ else} \tag{4}$$

For each expert e_l^j, we define the *activation counts* a_l^j as the total number of times it gets activated. Formally, this could be expressed as follows:

$$a_l^j = \sum_{(x_i, y_i) \in \mathcal{D}} \mathbb{1}[\gamma(e_l^j, x_i) = 1] \tag{5}$$

Equivalently, Eqn.6 is performing ***a Monte-Carlo estimate of the marginal probability*** $P(e_l^j$ ***gets activated***$)$ using the dataset $\mathcal{D}$, which provides theoretical motivation for our adopted technique here.

Soft Counting We also propose another variant with a softer and more relaxed version of heavy-hitters counting. Intuitively, for certain tokens, if expert e_l^j is activated, we don't know whether it wins over other experts by a slight margin or gets activated with high confidence. Since the binary activation count can not capture this exact magnitude of confidence in activating certain experts, we propose to directly leverage the softmax probabilities as soft counts. Formally, this can be defined as:

$$a_l^j = \sum_{(x_i, y_i) \in \mathcal{D}} \text{softmax}(g_l(f_l(x_i)))^j \tag{6}$$

Layer Expert Pruning Now with the statistics of the heavy-hitters counts, we could leverage them as powerful guidance to remove experts that are unlikely to be activated for data from $P_{\mathcal{D}}$. Suppose we want to keep a total of $\hat{M}_l$ experts per layer, the kept experts at layer l are denoted as:

$$\text{ArgTopK}(\bigcup_{j\in[1,M_l]} \{a_l^j\}, \hat{M}_l) \tag{7}$$

We repeat this for every layer in the MoEs model to get an entire mask.

Global Expert Pruning Since the counts have uniform magnitude range across all layers, we can also carry out a global sorting and pruning to remove the experts with the least probability of getting activated. Suppose we only want to keep a total of $\hat{M}$ experts in the network. The kept ones are denoted as:

$$\text{ArgTopK}(\bigcup_{l\in[1,L]} \bigcup_{j\in[1,M_l]} \{a_l^j\}, \hat{M}) \tag{8}$$

Compared with the above Layer Expert Pruning option, this will provide a nonuniform expert sparsity pattern across layers but could potentially be of higher-quality.

To recap, for pruning experts from the MoEs model to reduce storage burden and memory footprints, we propose to perform pruning based on heavy-hitters counting. This counting could be either ***actual activation counts*** or a ***soft counting with softmax probabilities***. The actual removal of experts could be conducted either ***layer-wise*** or ***globally***. This gives us a total of four configurations with combinations of pairs of counting and removal strategies, and we are going to provide detailed ablation results in the Experiment section.

3.3 Enhancing Expert Efficiency: Advanced Finetuning Strategies

With the goal of reducing the number of experts activated for each token during inference, while still maintaining competitive performance, we propose different fine-tuning procedures for MoE model.

3.3.1 Top-K adaptation

Starting from a pretrained model trained with top-k > 1, we posit that by fine-tuning the model on a downstream task while reducing k during training is a feasible and simple strategy to adapt the model to utilize. We focus on fine-tuning since we are interested in utilizing existing pretrained more efficiently.

Given that we are trying to target the best open-source available MoE model, which is the Mixtral 8x7b [1] standard fine-tuning was not feasible given the amount of memory required (even using 8xA100 80GB), therefore we opted for explore QLoRA Dettmers et al. (2023) fine-tuning on the self-attention blocks to reduce the memory footprint of the optimization.

In this work, we propose Top-K reduction procedures with simplicity in mind: Static top-k with $k < K$ and Annealing top-k from $K \rightarrow k$ with $k < K$. We also explore additional methods such as QLoRA fine-tuning targeting only the gating network.

3.3.2 Entropy-based gating regularization

Entropy, in the context of information theory, is a measure of the unpredictability or randomness of a distribution. For a categorical probability distribution (which is the case of the expert gating network), entropy is defined as $H(X) = -\sum_{i=1}^{n} p(x_i)\log p(x_i)$. We posit that a gating network with a more peaky distribution, meaning lower entropy, relies more heavily on a single expert. Therefore, by minimizing the entropy of the gating network's distribution, we encourage the model to make more decisive selection of experts while reducing the computational overhead associated with activating multiple experts.

The final loss we propose is $loss = \mathcal{L}_{\text{cross entropy}} + \lambda\mathcal{L}_{\text{entropy}}$, and explicitly:

[1] On March 17th, 2024, on the final deadline of this report, Grok-1 model came out which is the largest open-source MoE model with 314B parameters. Interestingly, this model has a very similar architecture to Mixtral 8x7b model, with 8 experts and top-2 routing which makes this work also applicable for that model.

$$\text{loss} = -\sum_{t=1}^{T}\sum_{i} y_{t,i}\log(p_{t,i}) + \lambda\left(-\sum_{t=1}^{T}\sum_{j} p_{t,j}\log(p_{t,j})\right) \tag{9}$$

Here λ is a hyperparameter, and the first term denotes the standard cross-entropy loss, and the second term denotes the entropy loss. The top-2 gating mechanism inherently provides a form of redundancy, which can be beneficial for robustness and handling uncertainty. Moving to a more peaky, top-1 distribution could reduce this redundancy, potentially making the model more sensitive to errors in the expert selection.

3.4 SEER-MoE: Sparse Expert Efficiency through Regularization for Mixture-of-Experts

We introduce SEER-MoE (Sparse Expert Efficiency through Regularization for Mixture-of-Experts), a two-stage approach designed to enhance the computational efficiency of pslightlyed MoE models, specifically targeting the Mixtral 8x7b (but is also applicable to other MoE models) such as Grok-1:

Stage 1: MoE-specific sparsification decreases the total number of available experts (M_l) in each MoE layer via effectively pruning the less significant ones.

Stage 2: Top-K adaptation i regularization techniques during fine-tuning, encouraging the model to rely on fewer experts without compromising the quality of the learned representations while better utilizing the information from the experts that have been pruned in Stage 1.

Finally, considering the total Model Parameter/FLOPs analysis from Section 3.1, by reducing the number of experts by 25% and using a single activated expert, we reduce the model parameters by $\approx 25\%$ and FLOPs by $\approx 27\%$ while only slightly degrading model performance.

4 Experiments

This section outlines the experimental setup, including model configurations, training details, and evaluation metrics used to assess the performance of our proposed SEER-MoE method.

4.1 Experimental details

We utilized the Mixtral 8x7b model for all our experiments, chosen for its superior performance over OpenMoE in our initial tests (for more quality results, check Appendix C). Our fine-tuning employed QLoRA on 8 NVIDIA A100 GPUs (80GB), running for 1000 steps (unless specified otherwise) using Adam optimizer with a weight decay of 0.01.

Distributed training (Technical challenges with sharding). Despite facing challenges with distributed training and sharding, we managed to optimize our setup for efficient training. We utilized QLoRA fine-tuning to minimize memory usage, crucial for our hardware constraints. We also explored PyTorch's Fully Sharded Data Parallelism (FSDP) Zhao et al. (2023) as a viable option but given that QLoRa was showing good results with focused on QLoRA. We implemented the full distributed training loop using the Gugger et al. (2022) library.

4.2 Data

We chose MMLU (Hendrycks et al., 2020) for multitask language understanding with 57 subjects and Stanford Sentiment Treebank-5 (SST5) dataset (Socher et al., 2013) for sentiment classification.

MMLU dataset contains questions, choices, and correct labels. The question and choice pairs are formulated to prompt the following format in Appendix B. The answer will be extracted from model output and compared against the correct label to determine output accuracy, which is used to evaluate various expert sparsification strategies (counting, pruning, subject-specific masking) quantitatively.

SST5 dataset contains texts and sentiment labels (very negative, negative, neutral, positive, very positive). The text is applied to the instruction template from Appendix B to form the prompt to the model. Accuracy will be computed by comparing the extracted label from the model's text output with the ground-truth label. We use the the SST5 training set ($\approx$ 8.5k examples) for fine-tuning model on different configurations (expert sparsity, top-K, fine-tuning precedure) and evaluate on the validation set (≈ 1.1k examples).

4.3 Evaluation Method

The evaluation metrics we used are result accuracy and reduced computation (FLOPs) and memory footprint.

Accuracy is measured by the percentage of answers extracted from model output matches with the validation dataset's labels. For MMLU task, we calculate the accuracy from number of correct answer among all questions and total number of 1531 questions in validation set. For SST task, we calculate the accuracy from number of correctly assigned sentiment with the total number of 1101 texts in validation set.

FLOPs and Memory Reduction is calculated based on model sparsification and ablation configuration. The FLOPs per token of the pruned model are calcuated based on Table 4. With expert removal, the model would contain less experts thus less parameters. The memory reduction can be calculated by number of expert reduction at each layer.

5 Results and Analysis

5.1 Expert Sparsification with Heavy-hitters Counting

We start by evaluating the effectiveness of our proposed pruning strategy to reduce the storage and memory footprint of MoE models with Heavy-hitters Counting.

Baselines We have 3 baselines to compare against: Dense baseline without expert, randomly pruning the experts and the state-of-the-art expert pruning stategy proposed by Lu et al. (2024).

For clarity, we report the results with the dense model numbers serving as the reference to indicate how much pruning affects the model. In terms of the baseline statistics of the dense model, from our evaluation on MMLU, we get a Mean Accuracy of 60.55% over all subjects and a Memory Usage of 86GB with $bf16$. Notice that our dense accuracy of Mixtral on MMLU does not exactly match the numbers reported in Lu et al. (2024) due to potential hardsware mismatch and prompt difference. However, we are comparing with the accuracy drop from the respective dense baselines which ensures a fair comparison on the same scale.

All reported results are zero-shot without fine-tuning.

Comparison We demonstrate the comparison of our top-confirming configuration with the baselines in Table 1. As observed, at either 25% or 50% expert sparsity, we significantly surpass all the baselines by a clear margin, achieving much smaller accuracy drop. For example, compared with the latest SOTA approach Lu et al. (2024), at the same 25% expert sparsity, we only lose $\mathbf{3.85}$ accuracy points, almost halve the accuracy drop of 6.40 of Lu et al. (2024).

Moreover, we could observe that with the expert sparsity introduced into the model, we successfully reduce the memory footprint for loading Mixtral by 24% with 25% expert sparsity and 45% with 50% expert sparsity. While it is not possible to fully load the entire dense Mixtral $8x7b$ model on a single $80GB$ $A100$ GPU(Dense: $86GB$), our proposed apporach provides a viable solution at the compensation of minimized accuracy loss by pruning unimportant experts from the model guided by Heavy Hitters Counting. We are also going to show later that this accuracy drop could be further minimized with Task-Specific Finetuning.

Ablation Here, we will compare different heavy hitters counting option and expert removal strategies. Recall that we could use either **actual activation counts** or **soft counting** to gather count data and conduct either **layer pruning** or **global pruning** to actually remove the experts. Moreover, for MMLU containing 57 total subjects, we also study whether using a **Subject-Specific Pruning** strategy benefits the performance. Concretely, with Subject-Specific Pruning, we are going to gather counting statistics and perform pruning for each subject independently, which creates a unique expert mask for solving each subject. Without this option, we gather counting numbers from samples of all subjects and adopt the same expert mask to solve all subjects. We demonstrate comprehensive ablation results in the following Table 2.

Method	Total Expert Sparsity ↑	Accuracy Drop from Dense ↓	Memory Usage↓	Speedup↑
Dense	0	0	×1	×1
Random	25%	6.17	×**0.76**	×**1.20**
Lu et al. (2024)	25%	6.40	×**0.76**	×**1.20**
Ours	25%	**3.85**	×**0.76**	×**1.20**
Random	50%	15.19	×**0.55**	×**1.27**
Lu et al. (2024)	50%	16.12	×**0.55**	×**1.27**
Ours	50%	**13.78**	×**0.55**	×**1.27**

Table 1: Comparison with baseline approaches. Ours achieves the minimized accuracy drop from the dense baseline at all expert sparsity levels. Notably, we beat the state-of-the-art Lu et al. (2024) with a clear margin.

Method	Counting Strategy	Pruning Strategy	Subject-Specific Pruning	Accuracy Drop from Dense ↓
Dense	n/a	n/a	n/a	0
Ours	Activation	Layer	Yes	16.59
Ours	Activation	Global	Yes	14.63
Ours	Soft	Layer	Yes	15.54
Ours	Soft	Global	Yes	7.90
Ours	Soft	Layer	No	12.80
Ours	Soft	Global	No	**3.85**

Table 2: Ablation results of ours. Using soft counting with global pruning and no subject-specific mask yields the best result.

From the table, we could make the following three observations:

Observation #1: Global pruning works better than layer pruning. Given our count statistics are uniform in magnitude across all layers, global pruning offers more flexible solutions without the layer constraint. Expectedly, we see the results are better with it.

Observation #2: Using soft counting works better than actual activation counts. Recall that soft counting, by directly accumulating the softmax probability for each expert, gives us a sense of the confidence in selecting each expert to better cope with the scenarios when certain expert barely wins over others. This is validated by the superiorty in results shown in the table.

Observation #3: Subject-specific pruning does not help to gain better performance on MMLU. Although we expect the subject-specific pruning could offer improvements by varying the expert mask adaptively based on the subject, the results suggest otherwise. There could be a theoretical support for this phenomenon. Recall that as discussed in Sec. 3.2, heavy-hitters counting is equivalently performing a maginal probability Monte-Carlo estimation. Without subject-specific pruning, more samples are used to build this estimation, which could make it more solid.

Visualization and Analysis Here, we provide a heatmap visualization of the collected heavy-hitter statistics of Mixtral on MMLU in Fig. 2. As observed in the figure, we could observe that some experts are heavily activated and leveraged during inference for example Expert #2 from Layer 26 and 30; whereas some experts are barely activated for example Expert #7 from Layer 22 and 23. From this discrepancy of acitvation patterns, we could see why heavy-hitters counting could serve as an effective guidance for pruning experts.

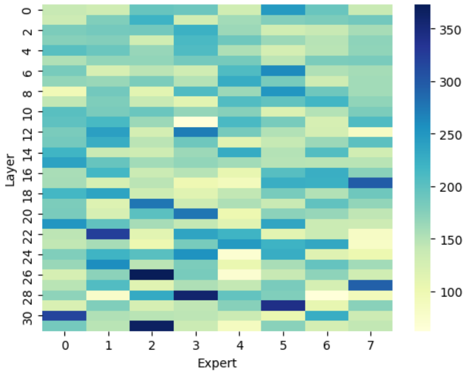

Figure 2: Heavy Hitters Counting Heatmap with Mixtral 8x7b on MMLU.

5.2 Finetuning

We assessed the efficacy of various strategies for reducing the number of activated experts within the Mixtral 8x7b model by fine-tuning on a sentiment classification task, SST5. The model was finetuned using the training set and evaluated on the evaluation set via text generation employing greedy decoding. For details regarding the prompts we used, refer to Appendix B.

We established a baseline by fine-tuning with top-2 routing to gauge the performance delta attributed to the reduced number of activated experts. This was further compared with a baseline in-context learning approach. We also evaluate the different procedures proposed in Section 3.3. Table 3 summarizes our findings.

Our results reveal a performance gap when using a single expert as opposed to two, with zero-shot accuracy dropping by 8.2 percentage points. However, through QLoRA fine-tuning (FT), this gap is mostly bridged, yielding a similar accuracy of 50.8%. Notably, employing two experts still holds a marginal advantage, as indicated by the 53.6% accuracy rate post fine-tuning. Nevertheless, when allowed two experts, the QLoRA approach demonstrated a slight improvement over the zero-shot baseline. These findings suggest that fine-tuning can recover some of the losses attributed to reducing the number of activated experts. Finally, the approach that works the best combines Annealing with Entropy minimization approach, performing better than naive Top-1 finetuning and only 1.8% less than Top-2 finetuning.

Method	Top-K	SST5 acc ↑
Zero-shot	1	42.6%
Zero-shot	2	50.8%
QLoRA FT	1	**50.8%**
QLoRA FT	2	**53.6%**
QLoRA on router	1	diverges
QLoRA on router	2	51.4%
FT + Entropy loss, $\lambda = 1$	1	45.5
FT + Entropy loss, $\lambda = 0.1$	1	50.7%
+ Annealing Top-K	1	48.6%
+ Annealing Top-K (more steps)	1	51.4%
+ Annealing Top-K + Entropy Loss	1	**51.8%**

Table 3: Comparison of fine-tuning strategies and their impact on SST5 accuracy.

5.3 SEER-MoE: putting everything together

We now explore whether combining both approaches yield additional gains. We first sparsify the experts via the heavy-hitters counting and then finetune. Our goal with this is to understand how we can reduce FLOPs utilization without losing performance.

Notably, our SEER-MoE approach, achieved a competitive accuracy of **48.0%** while only activating a single expert. This highlights the potential of SEER-MoE to maintain high model performance even under significantly reduced computational overhead. Remarkably, the accuracy attained mirrors that of the two-expert configuration following only QLoRA fine-tuning, underscoring the effectiveness of our comprehensive strategy in reducing FLOPs without detriment to accuracy.

Counting / Pruning Strategy	Top-K FT Method	Top-K	SST5 acc ↑
Activation / Global (25%)	QLoRA FT	2	**49.0%**
Activation / Global (25%)	QLoRA FT	1	47.5%
Soft / Global (25%)	QLoRA FT	1	46.7%
Activation / Global (25%)	QLoRA FT + Annealing Top-K + Entropy Loss	1	**48.0%**

Figure 3: Full stage approach results on SST5.

Furthermore, it is evident from the results that the activation-based pruning combined with a single-expert QLoRA fine-tuning confers a substantial accuracy gain over the soft pruning approach. This suggests that the more targeted activation-based pruning method combines effectively with our fine-tuning paradigm.

6 Conclusion

Our SEER-MoE framework effectively mitigates some of the computational inefficiencies of Mixture-of-Experts (MoE) models with minor compromise to performance. Through a two-stage process that includes expert sparsification and Top-K adaptation via fine-tuning we've significantly cut down both FLOPs and memory usage for Mixtral 8x7b. Testing on SST5 and MMLU benchmarks shows that SEER-MoE achieves strong performance while reducing the number of active experts and parameters, making MoE models more viable across various applications and hardware constraints.

References

Machel Reid et al. 2024. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. 2023. Qlora: Efficient finetuning of quantized llms.

David Eigen, Marc'Aurelio Ranzato, and Ilya Sutskever. 2014. Learning factored representations in a deep mixture of experts.

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

Sylvain Gugger, Lysandre Debut, Thomas Wolf, Philipp Schmid, Zachary Mueller, Sourab Mangrulkar, Marc Sun, and Benjamin Bossan. 2022. Accelerate: Training and inference at scale made simple, efficient and adaptable. `https://github.com/huggingface/accelerate`.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2020. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*.

Robert A. Jacobs, Michael I. Jordan, Steven J. Nowlan, and Geoffrey E. Hinton. 1991. Adaptive mixtures of local experts. *IEEE Neural Computation*.

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. 2024. Mixtral of experts.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. 2020. Scaling laws for neural language models. *CoRR*, abs/2001.08361.

Xudong Lu, Qi Liu, Yuhui Xu, Aojun Zhou, Siyuan Huang, Bo Zhang, Junchi Yan, and Hongsheng Li. 2024. Not all experts are equal: Efficient expert pruning and skipping for mixture-of-experts large language models.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, Seattle, Washington, USA. Association for Computational Linguistics.

xAI. 2024. Open release of grok-1. `https://x.ai/blog/grok-os`. Accessed: March 17th, 2024.

Fuzhao Xue, Zian Zheng, Yao Fu, Jinjie Ni, Zangwei Zheng, Wangchunshu Zhou, and Yang You. 2024. Openmoe: An early effort on open mixture-of-experts language models. *arXiv preprint arXiv:2402.01739*.

Zhenyu Zhang, Ying Sheng, Tianyi Zhou, Tianlong Chen, Lianmin Zheng, Ruisi Cai, Zhao Song, Yuandong Tian, Christopher Ré, Clark Barrett, Zhangyang Wang, and Beidi Chen. 2023. $\mathrm{H}_2\mathrm{o}$: Heavy-hitter oracle for efficient generative inference of large language models.

Yanli Zhao, Andrew Gu, Rohan Varma, Liang Luo, Chien-Chin Huang, Min Xu, Less Wright, Hamid Shojanazeri, Myle Ott, Sam Shleifer, Alban Desmaison, Can Balioglu, Pritam Damania, Bernard Nguyen, Geeta Chauhan, Yuchen Hao, Ajit Mathews, and Shen Li. 2023. Pytorch fsdp: Experiences on scaling fully sharded data parallel.

Barret Zoph, Irwan Bello, Sameer Kumar, Nan Du, Yanping Huang, Jeff Dean, Noam Shazeer, and William Fedus. 2022. St-moe: Designing stable and transferable sparse expert models.

A Parameters/FLOPs per Token for MoE Transformers

Table 4 describes the amount of parameters and FLOPs per token for a Transformer MoE model.

Operation	Parameters	FLOPs per Token
Embed	$(n_{vocab} + n_{ctx})d_{model}$	$4d_{model}$
Attention: QKV	$n_{layer}d_{model}3d_{attn}$	$2n_{layer}d_{model}3d_{attn}$
Attention: Mask	—	$2n_{layer}n_{ctx}d_{attn}$
Attention: Project	$n_{layer}d_{attn}d_{model}$	$2n_{layer}d_{attn}d_{embd}$
MoE Feedforward	$n_{experts}n_{layer}2d_{model}d_{ff}$	$2n_{topk}n_{layer}2d_{model}d_{ff}$
MoE Gating	$n_{experts}n_{layer}d_{model}$	$2n_{experts}n_{layer}d_{model}$
De-embed	—	$2d_{model}n_{vocab}$

Table 4: Parameter counts and compute (forward pass) estimates for a MoE Transformer model. Nonlinearities, biases, and layer normalization are omitted.

B Prompts

In this section we describe the different prompts that we use for different tasks to query the desired answer. We have experimented with different prompts and the ones presented below are chosen due to better performance or that match previously reported benchmarks.

SST5 For Mixtral:

```
[INST] You are a helpful assistant. Your task is of sentiment classification.
Categorize the following text as either "very negative", "negative",
 "neutral", "positive" or "very positive":
\{TEXT\}
Only generate the label, without explanations:[/INST]
```

SST5 For OpenMoE:

```
<<SYS>> You are a helpful assistant. Your task is of sentiment classification. <</SYS>>
<s>[INST] Categorize the following text in one of the following sentiments
’very negative’, ’negative’, ’neutral’, ’positive’ or ’very positive’:
\{TEXT\} [/INST]
```

MMLU For Mixtral 8x7b:

```
[INST] The following are multiple choice questions (with answers) about \{SUBJECT\}.

\{QUESTION\}

\{CHOICES\}

Only respond with the letter of the correct answer and no explanation.
Answer:
[/INST]
```

MMLU For OpenMoE:

```
<<SYS>> You are a helpful assistant. Your task is of multiple choice question answering
based on your knowledge.
For subject \{SUBJECT\}
Choose the best answer (A), (B), (C), or (D)
 to the following question without explanation:<</SYS>>
```

```
<s>[INST] \{QUESTION\} from the following choices:

\{CHOICES\}

[/INST]
```

C OpenMoE results

We also experimented with OpenMoE Xue et al. (2024) models but we were not able to get reasonable results. We evaluate the OpenMoE-8B-Chat model on MMLU and SST5 and the results can be seen in Table 5.

Model	**Setup**	**SST5 Acc.**	**MMLU Acc.**
OpenMoE-8B-Chat (1.1T+SFT)	expert-topk=1	35.0%	25.7%
OpenMoE-8B-Chat (1.1T+SFT)	expert-topk=2	35.6%	26.5%

Table 5: Evaluation on MMLU and SST5 for different for OpenMoE models with top-1 and top-2.

Considering the random-guessing baseline for MMLU to be 25% and for SST5 to be 20%, we did not further pursue utilizing these models for additional experiments.