Task-Specific Expert Pruning for Sparse Mixture-of-Experts

Tianyu Chen[1,2], **Shaohan Huang**[4], **Yuan Xie**[3], **Binxing Jiao**[5], **Daxin Jiang**[5], **Haoyi Zhou**[1,2], **Jianxin Li**[1,2], **Furu Wei**[4]
[1]BDBC, Beihang University, China [2]SKLSDE Lab, Beihang University, China
[3]The Institute of Acoustics of the Chinese Academy of Sciences, China
[4]Microsoft Research [5]NLP Group, Microsoft STCA

Abstract

The sparse Mixture-of-Experts (MoE) model is powerful for large-scale pre-training and has achieved promising results due to its model capacity. However, with trillions of parameters, MoE is hard to be deployed on cloud or mobile environment. The inference of MoE requires expert parallelism, which is not hardware-friendly and communication expensive. Especially for resource-limited downstream tasks, such sparse structure has to sacrifice a lot of computing efficiency for limited performance gains. In this work, we observe most experts contribute scarcely little to the MoE fine-tuning and inference. We further propose a general method to progressively drop the non-professional experts for the target downstream task, which preserves the benefits of MoE while reducing the MoE model into one single-expert dense model. Our experiments reveal that the fine-tuned single-expert model could preserve 99.3% benefits from MoE across six different types of tasks while enjoying 2x inference speed with free communication cost.

1 Introduction

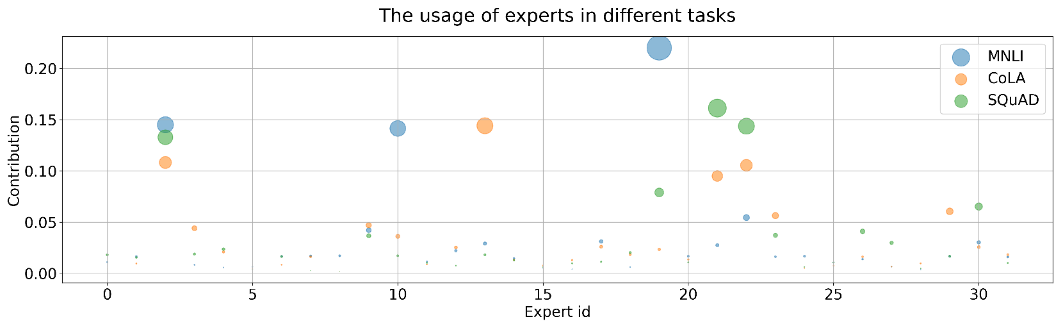

Figure 1: **The contribution of experts varies in different downstream tasks** The distribution of the contribution is quite long-tailed, indicating some experts are more "professional" on such tasks.

In recent years, scaling up neural network models has brought significant quality improvements in both language Devlin et al. [2019], Brown et al. [2020] and vision domains Dosovitskiy et al. [2020]. While large dense models have hit the boundary of model size and hardware capacity, another line of work has proposed sparsely-gated Mixture-of-Experts (MoE) layers as an efficient alternative to dense models Lepikhin et al. [2020], Fedus et al. [2021b], Riquelme et al. [2021]. In a vanilla sparsely-gated MoE model, each token of the input sequence activates a different subset of the experts, hence the computation cost per token becomes only proportional to the size of the activated sub-network. For a general pre-training task, the complexity and diversity of the optimization target require each sub-network (expert), to take a professional part of sub-task, thus increasing the model capacity and leading to generalization ability.

However, the proficiency of experts in MoE models is well under-explored in fine-tuning and serving stages. Different from a general and complex pre-training task, the downstream task for fine-tuning an MoE model is always specific and data-limited. Considering the diversity of experts in pre-trained MoE models, only a subset of experts will be well-activated, which may dominate the contribution to the input sequence. As depicted in Figure 1, even with the same pre-trained MoE model, the contribution of experts in MNLI, CoLA and SQuAD are quite different. Based on such observation, we propose an important question, *For downstream tasks, could we select the most professional expert by certain fine-tuning strategies?*.

It may have several advantages to convert large sparse MoE models into small single-expert dense model by task-specific expert pruning method. First, the selected most professional expert for the task inherits the most transferable knowledge from MoE pre-training, which may contribute to better performance than the dense-training counterpart. Second, the single-expert model avoids the parallelism of experts, which decreases the requirements of devices and cut the cost of communication between devices, thus boosting the inference efficiency. Finally, no extra distillation effort should be paid, single-expert paradigm turned the MoE model into a dense counterpart architecture, compatible to a wide range of different downstream tasks.

In this work, we explore a general paradigm about how to find the most professional expert in MoE models during the fine-tuning stage, which aims to turn the sparsely-trained MoE model into a single-expert dense counterpart. We dive into the under-explored areas, including the criterion of pruning and the timing to drop experts. In an ideal scenario, we want to efficiently train a single large model maximizing the positive transfer while expanding the capacity bottleneck; meanwhile, we would like to enjoy the benefits of sparsely activated sub-networks per task at inference time. In practice, we set a dropping threshold at every k steps of the fine-tuning process. If the proficiency score of one expert fails to meet the dropping threshold, the expert should be pruned. We repeat the dropping operation until there is only a single expert in every MoE layer or the fine-tuning process has passed half-way. Usually, the most professional expert of the task will have dominated the contribution before the end of half fine-tuning process, thus we must drop all other non-professional experts and keep the selected expert for further optimization.

We summarize our main contributions as follows:

- We explore the task-specific expert pruning paradigm, a general device-friendly fine-tuning method for Mixture-of-Experts. We carefully drop the non-professional experts for the downstream task and keep the most professional one for inference and serving. We show that the selected expert could be positively transferred and successfully preserve most benefits from MoE pre-training.
- We show that pruned MoEs outperform their dense counterparts on NLI, sentiment, similarity and QA tasks in absolute terms. Moreover, at inference time, the pruned MoEs can (i) match or even outperform the all-experts fine-tuned MoEs. and (ii) eliminate the communication cost between different devices.
- We provide visualization of the expert pruning during the fine-tuning process, revealing patterns and conclusions which helped motivate selection decisions and may further improve the final performance of fine-tuned models.

2 Related Work

Sparse Mixture-of-Experts Models Sparse Mixture-of-Experts (MoE) models that provide a highly efficient way to scale neural network training, have been studied recently in Lepikhin et al. [2020], Fedus et al. [2021b], Shazeer et al. [2017]. Compared to standard dense models that require extremely high computational cost in training, MoE models are shown to be able to deal with a much larger scale of data and converge significantly faster. Some recent studies focus on the expert assignment problem, such as formulating token-to-expert allocation as a linear assignment Lewis et al., hashing-based routing Roller et al., and distillation in routing function Dai et al. [2022]. In addition, a few recent works show different training methods for sparse MoE models. Dua et al. [2021] found that a temperature heating mechanism and dense pre-training can improve the performance of sparse translation models. Nie et al. [2021] propose a dense-to-sparse gating strategy for MoE training. Some papers have explored the performance of supervised fine-tuning in MoE models and proposed some methods to boost the quality of fine-tuning. Artetxe et al. introduced an expert regularization to improve sparse model fine-tuning. Chi et al. [2022] proposed a dimension reduction and L_2 normalization to solve the representation collapse in fine-tuning. Instead of improving fine-tuning of MoE models, this work focuses on fine-tuning spare models into single-expert dense models directly, namely task-specific expert pruning .

Efficient Inference in MoE Models MoE models have achieved promising results. However, they fail to meet requirements on inference efficiency in fine-tuning and serving stage. Recent studies apply knowledge distillation for faster inference. Rajbhandari et al. proposed a smaller MoE model (named Mixture-of-Students) with knowledge distillation. However, sparse MoE makes inference difficult for the following reasons: if the mode parameters exceed the memory capacity of a single accelerator device, inference requires multiple devices. Communication costs across devices further increase the overall serving cost. There have been some works that apply distillation of sparse MoE models into dense models Fedus et al. [2021a], Xue et al., Artetxe et al.. Fedus et al. [2021a] presents a study of distilling a fine-tuned sparse model into a dense model. The task-specific distillation method needs to fine-tune different models for different tasks respectively. This work does not require additional fine-tuning on MoE models and adapts sparse MoE models into single-expert dense models directly.

3 Preliminary

Recall the Mixture-of-Expert Transformer models proposed by Shazeer et al. Shazeer et al. [2017], where the MoE layer takes an input token representation x and then routes this to the best determined top-k experts selected from a set $\{Exp_i(x)\}_{i=1}^{E}$ of E experts. The logits $h(x) = W_r \cdot x$ produced by the router variable W_r is normalized via a softmax distribution over the available E experts at that layer. The gated-value of expert i is given by,

$$\alpha_i(x) = \frac{e^{h(x)_i}}{\sum_j^E e^{h(x)_j}} \tag{1}$$

The top-k gate values are selected for routing the token x. if $\mathcal{P}$ is the set of selected top-k indices then the output computation of the layer is the linearly weighted combination of each expert's computation on the token by the gated value,

$$y = \sum_{i \in \mathcal{P}} \alpha_i(x) Exp_i(x) \tag{2}$$

Notation In this paper, the symbol E denotes the total number of experts while e denotes the id of the expert in one Mixture-of-Experts layer. N denotes the total training steps and w denotes one specific training window. Z denotes the number of survival experts in the current training window.

Problem Formulation Given the labeled data of one downstream task and a pre-trained MoE model with several mixture-of-experts layers, where each MoE layer has E experts, we try to find the most professional expert $\hat{e}$ for the task in every MoE layer and prune the other non-professional experts, during fine-tuning the MoE model on the labeled data.

4 Method

4.1 Overview

We illustrate our method in Figure 2, where we do not break the the standard pipeline of fine-tuning an MoE model on the downstream tasks. Instead, we split the N training steps of the whole fine-tuning process into E training windows evenly. During one training window, we will accumulate the proficiency score of each expert when the representations of the input tokens pass through the

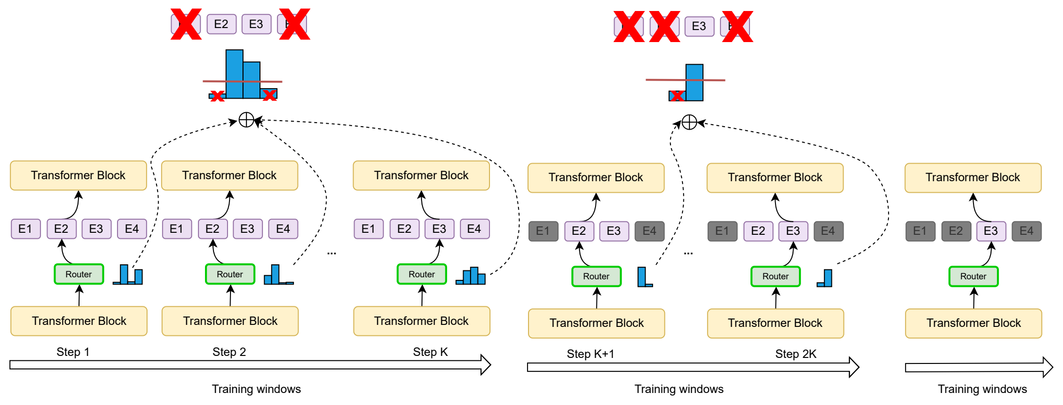

Figure 2: The pipeline of our method. We split the fine-tuning process into several training windows evenly and prune the non-professional experts at the end of each training window.

MoE layer. At the end of one training window, we decide which experts are non-professional and should be pruned. The pruned experts will never participate in the assignment of tokens at the later training windows, also will be dropped for inference. At the half point of the whole training schedule, we make the final decision and keep only the most professional expert in each MoE layer and prune all other experts. Once the most professional expert is selected, all input tokens are assigned to the expert and reduce the sparse MoE model into a dense counterpart.

4.2 The Criterion of Expert Proficiency

The first critical point of our method is the evaluation of expert proficiency. Note that when input with token sequence x_s, the MoE model will execute two operations related to the schedule of experts. The first operation is the assignment of tokens, where each token can only be assigned to a single expert. The second operation is conquering the division results, which uses the α_i to weighted sum the selected expert output and formulates the final output y_s. During fine-tuning of MoE models, both of the expert schedule-related operations are criterion of expert proficiency. We define them respectively as hit rate R and alpha score $\mathcal{A}$.

In our method, we choose $\mathcal{A}$ as the default criterion and define the proficiency score of expert i in one training window as C_i:

$$C_i = \sum_{k}^{k+N/E} \mathcal{A}_i \tag{3}$$

$$\mathcal{A}_i = \frac{\alpha_i}{\sum_i^E \alpha_i} \tag{4}$$

Here, k, N represent the start step of the training window and the total training steps in the fine-tuning schedule respectively. Both R_i and $\mathcal{A}_i$ are accumulated during a training window of K training steps. As more tokens are assigned to one expert i and the gating alpha score $\mathcal{A}_i$ gets higher, the expert i is more critical to the final output y, thus leading the prediction on the downstream task. A higher hit rate R_i will yield more input tokens to be assigned to the expert i and fewer tokens to be computed by other experts, also may give proper approximation of the expert proficiency on the downstream tasks.

4.3 The Timing to Drop Non-Professional Experts

To find the most professional expert, one intuitive method is two-pass training, which selects the expert in the first pass and fine-tunes the dense model in the second pass. However, different from post-training pruning or distillation strategy, our method is pursuing a one-pass fine-tuning paradigm, which does not incur extra computation after fine-tuning an MoE model. In our during-training strategy, we find when to drop the non-professional experts greatly affects the final performance of fine-tuned model. In one respect, if we accumulate the proficiency score $\mathcal{C}$ in a relatively long training window, the approximation of the expert proficiency will be more reliable and closer to the whole task. In another respect, the early decision to drop non-professional experts leaves plenty of training room for optimizing the chosen professional expert and avoid wasting computing on non-professional experts. Thus in our one-pass strategy, the timing to drop is a trade-off between the more appropriate selection of the professional expert and the more optimization of the final expert.

In practice, we propose combining the progressive dropping and the force dropping. We divide the whole training steps evenly into E windows and progressively make the dropping decision at the end of each window. If $E/2$ windows have passed and the MoE layer still has more than one survival experts, only the expert with the highest proficiency score will be kept.

4.4 The Progressively Dropping

During progressive dropping, we have $E/2$ chances to make the dropping decision. It is an important problem in each dropping decision that how many experts should be dropped. To select out the most non-professional experts in each training window, we propose a dynamic dropping threshold $\mathcal{T}$. Any expert with a proficiency score below the threshold $\mathcal{T}$ will be dropped.

$$\mathcal{T} = \beta * 1/Z \tag{5}$$

Here, Z denotes the number of survival experts in the current training window. β is a hyper-parameter to control the speed of the progressively dropping. A larger β will increase the threshold $\mathcal{T}$ and more experts will be dropped in each decision. In our experiments, a relatively small β maybe a good default to preserve maximum performance of MoE models.

5 Experiments

In this section, we design both sequence-level and token-level tasks to evaluate the performance and efficiency of our task-specific expert pruning paradigm. We also provide a detailed discussion about the hyper-parameters that affect our strategy.

5.1 Evaluation Tasks

GLUE Wang et al. [2019], the General Language Understanding Evaluation benchmark, is a collection of tools for evaluating the performance of models across a diverse set of existing NLU tasks, including STS-B Cer et al. [2017], CoLA Warstadt et al. [2019], MRPC Dolan and Brockett [2005], RTE Dagan et al. [2005], Haim et al. [2006], Giampiccolo et al. [2007], Bentivogli et al. [2009], SST-2 Socher et al. [2013], QQP, MNLI Williams et al. [2018] and QNLI Rajpurkar et al. [2016]. Following previous works Devlin et al. [2019], Turc et al. [2019], we exclude the WNLI task from the GLUE benchmark. Considering the small data size of RTE may lead to significant bias in performance evaluation, we exclude RTE results in Table 1, but include all GLUE tasks in Appendix Table 6.

SQuAD2.0 Rajpurkar et al. [2018], the Stanford Question Answering Datasets, is a collection of question-answer pairs derived from Wikipedia articles. In SQuAD, the correct answers can be any sequence of tokens in the given tasks. SQuAD2.0, the latest version, combines the 100,000 questions in SQuAD1.1 with over 50,000 unanswerable questions written by crowd workers in forms that are similar to the answerable ones.

5.2 Experiments Settings

We use the base Switch Transformer Fedus et al. [2021b] architecture for this set of experiments. For a fair comparison, we also employ a dense transformer model of the same architecture as Switch-base except for the mixture-of-experts layers. We pre-train both the MoE and the dense model for 125k steps on the BooksCorpus Zhu et al. [2015] and English Wikipedia corpus Foundation, with the masked language modeling objective. For the MoE model, we use a MoE layer of 32 experts in the 4th and 8th transformer block. A more detailed pre-training settings and model hyperparameters could be found in Appendix A.

Here we include several fine-tuning settings for comparison. All fine-tuning hyperparameters can be found in Appendix B:

Dense-ft: The standard dense model fine-tuning, which is for the demonstration of MoE benefits and as a strong baseline for our expert pruning strategy.

MoE-ft: The standard MoE fine-tuning method, where all 32 experts participate in the optimization for the downstream task and also for the final inference.

Two-pass-staged-drop: An intuitive two-pass pruning baseline, where we first accumulate the proficiency score of each expert in the first pass fine-tuning and keep the most professional expert to fine-tune the MoE model again.

Two-pass-eager-drop: A baseline to prove the effectiveness of our proficiency criterion, where we select the most professional expert based on our pipeline and keep the selected expert to re-fine-tune the MoE model.

MoE-staged-pruning: A task-specific expert progressively pruning strategy, where we drop one most non-professional expert at the end of each training window, yields a single-expert model for inference.

MoE-eager-pruning: The method implements all parts of our task-specific expert pruning pipeline, which will drop the experts of proficiency score below the bar at each training window and execute a force drop at the latter half training schedule.

5.3 Preserving Benefits of MoE into a Single Expert

Table 1: **The performance on the GLUE tasks for the dense models and the MoE models**, measured on the development sets. We report the average results by a set of seeds. **AVG** denotes the average score of all the tasks. Task-specific expert pruned MoE models with only single expert could achieve about 0.84 avg glue score than dense models.

Settings	CoLA	STS-B	MNLI-m	MNLI-mm	SST-2	QQP	QNLI	MRPC	AVG
Dense-ft	57.23	89.18	85.87	85.87	92.87	91.20	92.23	88.07	85.32
MoE-ft	63.75	89.37	86.93	86.93	94.03	91.37	92.90	89.13	86.80
Two-pass-staged-drop	58.32	88.64	86.30	85.90	90.00	90.50	91.40	86.90	84.75
Two-pass-eager-drop	60.89	89.02	86.10	86.20	92.60	90.60	91.50	87.72	85.58
MoE-staged-pruning	56.20	89.00	85.33	85.33	92.33	90.07	91.97	86.83	84.63
MoE-eager-pruning	**61.47**	**89.43**	**86.10**	**86.10**	**93.17**	**91.20**	**92.53**	**89.30**	**86.16**

In Table 1, the results can be divided into three groups. The first group is the baseline, where both sparsely pre-trained MoE models and their densely pre-trained counterparts are fine-tuned in standard ways. We observe the MoE enjoys full advantage at all tasks, of an 1.48 average glue score over their densely pre-trained counterparts, which may thank to the strong generalization ability of mixture-of-experts pre-training.

In the second group, we compare two intuitive expert pruning strategies based on a two-pass optimization. With more computation budget of another pass, we could select the most professional expert in the first pass while optimizing the selected expert in the second pass. We find for the two pas staged drop method, another pass could boost the final performance of the model by 0.12 average GLUE score. For two pass eager drop, another pass hurts the performance, cutting down the average GLUE score by 0.58. We assume another pass gives plenty of room for optimization of the final selected expert in the staged drop, which maybe sub-optimal yet in the first pass. However, for the two pass eager drop, the selected expert is may well optimized and tend to overfit in the second training pass.

In the last group, we are interested in whether our method could preserve most advantage of mixture-of-experts by keeping only single most professional expert. We find in the MoE staged pruning setting, the selected expert fails to preserve the advantage of the MoE and only achieves 84.63 average score, 0.69 below the densely pre-trained counterpart. However, for the MoE eager pruning, the selected expert could preserve most benefits from MoE pre-training and outperforms the densely pre-trained models by 0.84 average GLUE scores. We assume the failure of the staged strategy may be due to the lack of enough adaptation of the final selected expert, which we will further discuss in

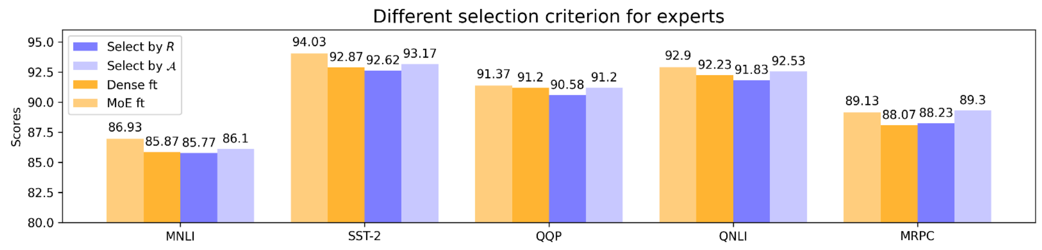

Figure 3: **Different pruning criterion for professional experts.** We compare the most professional expert chosen by hit rate R and alpha score $\mathcal{A}$ on five different tasks. The most professional expert chosen by $\mathcal{A}$ better narrow preserve the performance of standard MoE-ft.

section 5.5. In addition, we find the single expert selected by the eager method could even exceed the performance of standard fine-tuned MoE models, which have 32 experts, in STS-B and MRPC tasks. Considering the excessive model capacity of the MoE models, fine-tuning with all experts may not always be a good default for the data-limited downstream tasks.

Table 2: The performance on SQuAD2.0 task. The EM denotes the accuracy of exactly matched answers.

Settings	SQuAD2.0-F1	SQuAD2.0-EM
MoE-ft	80.59	77.85
Dense-ft	79.77	77.23
Two-pass-drop	80.11	77.31
Two-pass-eager-drop	80.33	77.40
MoE-staged-pruning	80.08	77.24
MoE-eager-pruning	**80.40**	**77.55**

For the token-level task, we present our results of SQuAD 2.0 in Table 2. We find the sparsely pre-trained MoE models exceed their densely counterparts by 0.82 F1 score and 0.62 EM score. For our task specific expert pruning method, both staged and eager pruning strategy exhibits comparable or better performance than the dense-ft baseline. The results of the SQuAD 2.0 is consistent with those in sentence-level GLUE benchmark, where the eager pruning outperforms the staged pruning by 0.32 F1 score and 0.31 EM score. In addition, the two pass optimization also outperforms the dense-ft baseline, where the two-pass-eager-drop exceeds dense-ft by 0.56 F1 score and 0.17 EM score. We assume token-level tasks are more expert-sensitive and our expert pruning method easily locates the most professional expert, leading to better performance.

5.4 Different Pruning Criterion

As mentioned in Section 4.2, both hit rate R and alpha score $\mathcal{A}$ could be the evaluation of expert proficiency. We present the results of these two different pruning criterion in Figure 3. The R based pruning only exceeds the dense-ft baseline on the MRPC task by 0.17 accuracy score while performing worse in the other five tasks. However, the $\mathcal{A}$ based pruning shows a much better performance across all five tasks and achieves better performance than the MoE-ft baseline on MRPC.

We assume more tokens assigned to one expert may not well represent its contribution because the output of the expert computation on token x will be re-weighted by gating score $\alpha_i(x)$. Thus the $\mathcal{A}$ could be viewed as "soft" R with a more precise evaluation of the expert proficiency.

5.5 A Closer Look at the Dropping Timing

One important question related to task-specific expert pruning is *when should we make the final decision to drop all other experts.* Recall that in our methods, two factors could affect the final decision. The first factor is the length of the training window and the second factor is the dropping threshold.

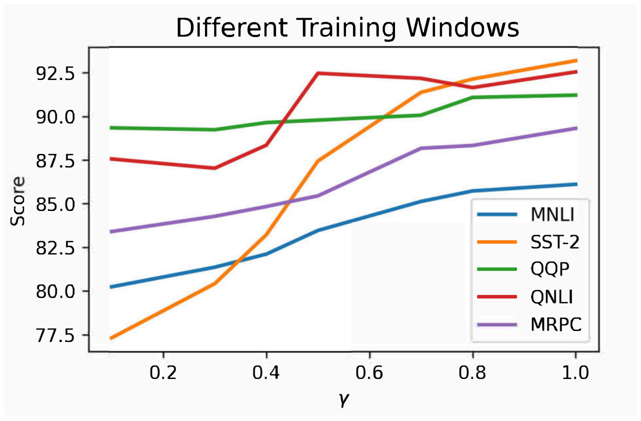

Figure 4: **The effect of training window length on GLUE tasks.** γ is a parameter to control the length of training window $\mathcal{L}$, where $\mathcal{L} = \gamma * \frac{N}{E}$. N and E denote the total training steps and the number of experts respectively.

As depicted in Figure 4, we investigate how long the training windows should be split to achieve better final performance on MNLI, SST-2, QQP, QNLI and MRPC tasks. We observe that for MNLI, SST-2, QQP and MRPC tasks, a longer training window contributes to better final performance of selected expert. While for the QNLI task, a training window length of $0.5 * \frac{N}{E}$ could also achieve comparable performance of a maximum training window. We assume for most tasks, longer training windows are beneficial for boosting the performance of the selected expert.

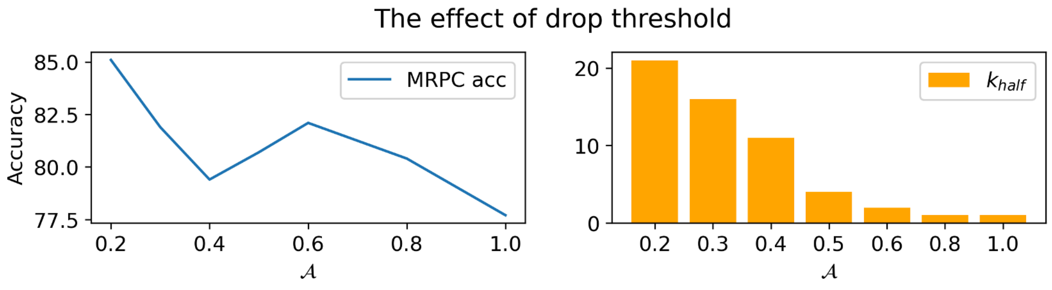

Figure 5: **Comparison of different drop threshold.** On the left figure, we illustrate the fine-tuned MoE accuracy on MRPC task with a different $\mathcal{A}$ threshold. On the right figure, we depict the number of survival experts at half of the schedule, which denoted as K_{half}.

Another way to control the progress of the pruning is the dropping threshold. We show the effect of different drop thresholds in Figure 5. A lower dropping threshold may lead to slow progress of expert selection and more survival experts to be forced to drop at the half training schedule, but provides more precise observation of experts contribution.

Finally, we compare the timing for making the final expert decision in Figure 6. For the staged pruning method, we find some experts gradually dominate the contribution to the final output during training. However, even if the final selected expert has dominate the contribution in early training window, it could not be optimized independently until the last training window, which may answer the relatively poor performance in inference stage. Based on such observation, we propose a more eager pruning method, which drops all non-professional experts at the half of training since our selected expert is dominant in the contribution. This method leaves the later half of the training schedule to optimize the selected expert, which leads to a better inference performance than the staged method.

5.6 The Efficiency of Inference

We present the inference speed of different settings in Figure 7, evaluated on MNLI and SQuAD tasks. We use the tokens per second as an metric of the model inference speed and use NVIDIA-V100-32GB for testing. For the MoE model fine-tuned with 32 experts, the extra communication cost between

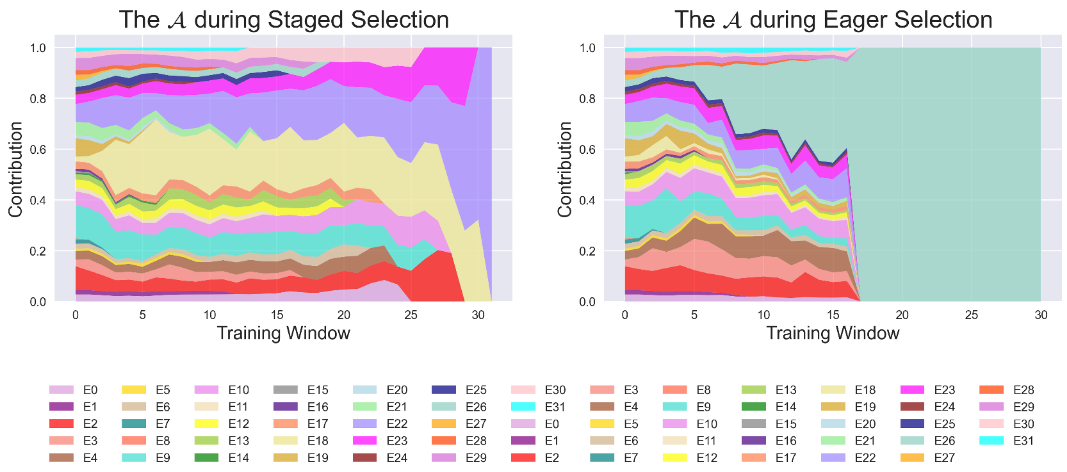

Figure 6: **Comparison of the timing for making final expert decision**. In the left sub-figure, we illustrate the staged pruning, which adopts an "lazy" pruning strategy and make the final decision at the last training window. In the right sub-figure, we illustrate a more "eager" strategy, which drops more experts in the first several training windows and leave the last half training schedule for the most professional expert.

different devices greatly reduces the inference speed, leading to the worst inference efficiency among all settings. The single expert models optimized with our task-specific expert pruning method, enjoy an inference efficiency advantage of 2x speed up on MNLI task and 1.28x speed up on SQuAD task. Because of the extra sub-MoE layers in the selected expert, our method slightly lags behind the inference efficiency of the dense model, achieving 80% inference speed of dense model on MNLI and SQuAD tasks.

6 Conclusion

In this paper, we discussed a more inference friendly fine-tuning paradigm for the pre-trained Mixture-of-experts models by fine-tuning the most professional expert and drop other experts, namely task-specific expert pruning. We empirically demonstrate that this new fine-tuning paradigm could preserve most benefits of the pre-trained MoE models and much better than the densely pre-trained counterparts, on both sentence-level tasks and token-level tasks. By carefully examining the factors of expert pruning across different settings, we demonstrate the superiority of our eager expert pruning paradigm over other possible solutions like two-pass optimization or staged expert pruning.

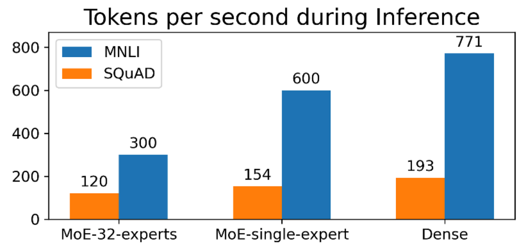

Figure 7: The inference speed of different fine-tuned models.

We conclude by highlighting the task specific expert pruning paradigms that are more inference friendly while retaining the quality gains of MoE models are promising direction of future exploration.

References

Mikel Artetxe, Shruti Bhosale, Naman Goyal, Todor Mihaylov, Myle Ott, Sam Shleifer, Xi Victoria Lin, Jingfei Du, Srinivasan Iyer, Ramakanth Pasunuru, Giri Anantharaman, Xian Li, Shuohui Chen, Halil Akin, Mandeep Baines, Louis Martin, Xing Zhou, Punit Singh Koura, Jeff Wang, Luke Zettlemoyer, Mona Diab, Zornitsa Kozareva, and Ves AI Stoyanov Meta. Efficient Large Scale Language Modeling with Mixtures of Experts. URL https://github.com/pytorch/fairseq/. arXiv: 2112.10684v1 ISBN: 1.302410245.

Luisa Bentivogli, Bernardo Magnini, Ido Dagan, Hoa Trang Dang, and Danilo Giampiccolo. The fifth PASCAL recognizing textual entailment challenge. In *TAC*. NIST, 2009.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

Daniel M. Cer, Mona T. Diab, Eneko Agirre, Iñigo Lopez-Gazpio, and Lucia Specia. Semeval-2017 task 1: Semantic textual similarity multilingual and crosslingual focused evaluation. In *SemEval@ACL*, pages 1–14. Association for Computational Linguistics, 2017.

Zewen Chi, Li Dong, Shaohan Huang, Damai Dai, Shuming Ma, Barun Patra, Saksham Singhal, Payal Bajaj, Xia Song, and Furu Wei. On the representation collapse of sparse mixture of experts. *arXiv preprint arXiv:2204.09179*, 2022.

Ido Dagan, Oren Glickman, and Bernardo Magnini. The PASCAL recognising textual entailment challenge. In *MLCW*, volume 3944 of *Lecture Notes in Computer Science*, pages 177–190. Springer, 2005.

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

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT (1)*, pages 4171–4186. Association for Computational Linguistics, 2019.

William B. Dolan and Chris Brockett. Automatically constructing a corpus of sentential paraphrases. In *Proceedings of the Third International Workshop on Paraphrasing, IWP@IJCNLP 2005, Jeju Island, Korea, October 2005, 2005*. Asian Federation of Natural Language Processing, 2005. URL https://aclanthology.org/I05-5002/.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

Dheeru Dua, Shruti Bhosale, Vedanuj Goswami, James Cross, Mike Lewis, and Angela Fan. Tricks for Training Sparse Translation Models. 2021. URL http://www.statmt.org/wmt15/translation-task.html. arXiv: 2110.08246v1 ISBN: 2110.08246v1.

William Fedus, Google Brain, Barret Zoph, and Noam Shazeer. SWITCH TRANSFORMERS: SCALING TO TRILLION PARAMETER MODELS WITH SIMPLE AND EFFICIENT SPARSITY. 2021a. arXiv: 2101.03961v1.

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

Wikimedia Foundation. Wikimedia downloads. URL https://dumps.wikimedia.org.

Danilo Giampiccolo, Bernardo Magnini, Ido Dagan, and Bill Dolan. The third PASCAL recognizing textual entailment challenge. In *ACL-PASCAL@ACL*, pages 1–9. Association for Computational Linguistics, 2007.

R Bar Haim, Ido Dagan, Bill Dolan, Lisa Ferro, Danilo Giampiccolo, Bernardo Magnini, and Idan Szpektor. The second pascal recognising textual entailment challenge. In *Proceedings of the Second PASCAL Challenges Workshop on Recognising Textual Entailment*, 2006.

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. *CoRR*, abs/2006.16668, 2020. URL https://arxiv.org/abs/2006.16668.

Mike Lewis, Shruti Bhosale, Tim Dettmers, Naman Goyal, and Luke Zettlemoyer. BASE Layers: Simplifying Training of Large, Sparse Models. URL https://github.com/pytorch/fairseq/. arXiv: 2103.16716v1.

Xiaonan Nie, Shijie Cao, Xupeng Miao, Lingxiao Ma, Jilong Xue, Youshan Miao, Zichao Yang, Zhi Yang, and Bin Cui. Dense-to-Sparse Gate for Mixture-of-Experts. Technical Report arXiv:2112.14397, arXiv, December 2021. URL http://arxiv.org/abs/2112.14397. arXiv:2112.14397 [cs] type: article.

Samyam Rajbhandari, Conglong Li, Zhewei Yao, Minjia Zhang, Reza Yazdani Aminabadi, Ahmad Awan, Jeff Rasley, and Yuxiong He Microsoft. DeepSpeed-MoE: Advancing Mixture-of-Experts Inference and Training to Power Next-Generation AI Scale. URL https://github.com/microsoft/DeepSpeed. arXiv: 2201.05596v1.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100, 000+ questions for machine comprehension of text. In *EMNLP*, pages 2383–2392. The Association for Computational Linguistics, 2016.

Pranav Rajpurkar, Robin Jia, and Percy Liang. Know what you don't know: Unanswerable questions for squad. In *ACL (2)*, pages 784–789. Association for Computational Linguistics, 2018.

Carlos Riquelme, Joan Puigcerver, Basil Mustafa, Maxim Neumann, Rodolphe Jenatton, André Susano Pinto, Daniel Keysers, and Neil Houlsby. Scaling vision with sparse mixture of experts. In *NeurIPS*, pages 8583–8595, 2021.

Stephen Roller, Sainbayar Sukhbaatar, Arthur Szlam, and Jason Weston. Hash Layers For Large Sparse Models. arXiv: 2106.04426v3.

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.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Y. Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *EMNLP*, pages 1631–1642. ACL, 2013.

Iulia Turc, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Well-read students learn better: On the importance of pre-training compact models. *arXiv preprint arXiv:1908.08962*, 2019.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019. URL https://openreview.net/forum?id=rJ4km2R5t7.

Alex Warstadt, Amanpreet Singh, and Samuel R. Bowman. Neural network acceptability judgments. *Trans. Assoc. Comput. Linguistics*, 7:625–641, 2019.

Adina Williams, Nikita Nangia, and Samuel R. Bowman. A broad-coverage challenge corpus for sentence understanding through inference. In *NAACL-HLT*, pages 1112–1122. Association for Computational Linguistics, 2018.

Fuzhao Xue, Xiaoxin He, Xiaozhe Ren, Yuxuan Lou, and Yang You. One Student Knows All Experts Know: From Sparse to Dense. arXiv: 2201.10890v2.

Yukun Zhu, Ryan Kiros, Rich Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *The IEEE International Conference on Computer Vision (ICCV)*, December 2015.

A Hyperparameters for Pre-training

Table 3 presents the hyperparameters for pre-training. Table 4 presents the hyperparameters of our MoE model.

Hyperparameters	Value
Optimizer	Adam
Training steps	125,000
Batch size	2,048
Adam ϵ	1e-6
Adam β	(0.9, 0.98)
Maximum learning rate	5e-4
Learning rate schedule	Linear decay
Warmup steps	10,000
Weight decay	0.01
Transformer dropout	0.1
MoE balance loss weight	1e-2

Table 3: Pre-training hyperparameters

Hyperparameters	Value
Transformer blocks	12
Hidden size	768
FFN inner hidden size	3,072
Attention heads	12
Number of experts	32

Table 4: Model hyperparameters

B Hyperparameters for Fine-tuning

Table 5 presents the hyperparameters for fine-tuning.

Hyperparamters	**CoLA**	**STS-B**	**RTE**	**MNLI**	**SST-2**	**QQP**	**QNLI**	**MRPC**	**SQuAD**
Batch Size	[32,16]	[32,16]	[32,16]	[32]	[32]	[32]	[32]	[32,16]	[32]
Seed	[1,2,3]	[1,2,3]	[1,2,3]	[2,3,5]	[2,3,5]	[2,3,5]	[2,3,5]	[1,2,3]	[1,2,3]
Learning rate	[2,3,4,5]e-5	[2,3,4,5]e-5	[2,3,4,5]e-5	[1,2,3,4]e-5	[1,2,3,4]e-5	[1,2,3,4]e-5	[1,2,3,4]e-5	[2,3,4,5]e-5	[2,3,4]e-5
Warm up	[16, 10]	[16, 10]	[16, 10]	[16]	[16]	[16]	[16]	[16, 10]	[10]
Epochs	[2,3,5,10]	[2,3,5,10]	[2,3,5,10]	[2,3,5]	[2,3,5]	[2,3,5]	[2,3,5]	[2,3,5,10]	[3]

Table 5: Hyperparameters for fine-tuning downstream tasks.

C More Results

Table 6: The complete GLUE results.

Settings	**CoLA**	**STS-B**	**RTE**	**MNLI-m**	**MNLI-mm**	**SST-2**	**QQP**	**QNLI**	**MRPC**	**AVG**
Dense-ft	57.23	89.18	67.27	85.87	85.87	92.87	91.20	92.23	88.07	83.31
MoE-ft	63.75	89.37	61.83	86.93	86.93	94.03	91.37	92.90	89.13	84.03
Two-pass-staged-drop	58.32	88.64	60.32	86.30	85.90	90.00	90.50	91.40	86.90	82.03
Two-pass-eager-drop	60.89	89.02	60.06	86.10	86.20	92.60	90.60	91.50	87.72	82.74
MoE-staged-pruning	56.20	89.00	**64.53**	85.33	85.33	92.33	90.07	91.97	86.83	82.40
MoE-eager-pruning	**61.47**	**89.43**	62.70	**86.10**	**86.10**	**93.17**	**91.20**	**92.53**	**89.30**	**83.56**