EFFICIENT EXPERT PRUNING FOR PRE-TRAINING OF MIXTURE-OF-EXPERTS LARGE LANGUAGE MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

While Mixture-of-Experts (MoE) Large Language Models (LLMs) achieve higher accuracy with fewer active parameters, their pre-training remains challenging due to the enormous parameter sizes and low training efficiency caused by imbalanced expert routing. Unlike previous expert pruning methods that focus on the post-training phase, this paper proposes an efficient Expert Pruning Algorithm (EPA) for the pre-training of MoE LLMs. This algorithm enhances training efficiency while preserving model accuracy by pruning underutilized experts and rearranging experts within expert parallel groups based on token distribution. Extensive experimental results demonstrate that EPA can significantly reduce model size and improve training efficiency while maintaining nearly unchanged accuracy. Specifically, a 1010B parameter MoE LLM trained from scratch using EPA exhibits substantial improvements in training efficiency and delivers excellent performance across tasks in various domains. Code and 1010B model will be made available.

1 INTRODUCTION

In recent years, significant progress has been made in the field of MoE large models, with work such as Mixtral(Jiang et al., 2024) and DeepSeek-V3 (Liu et al., 2024a) achieving important breakthroughs in model scale, accuracy, and training efficiency. The Mixture of Experts (MoE) architecture enables a significant increase in model capacity (number of parameters) without linearly inflating computation (FLOPs) by activating only a small subset of total parameters for each token, though the static parameters used to construct MoE models still take up considerable memory. During the training process, the MoE model often experiences expert imbalance with certain experts frequently activated and others rarely engaged, which can cause some experts to struggle to learn useful representations (impacting the model's overall performance) and also waste computational resources.

The addition of auxiliary loss is currently widely used to alleviate the phenomenon of imbalanced workload among experts, thereby enhancing computational performance(Fedus et al., 2022; Jiang et al., 2024). However, MoE architectures rely on assistance loss, making the final language modeling optimization goal is weakened, resulting in expert struggle to learn true expertise. Excessive pursuit of expert loading balancing will suppress the expressive ability of the model, but completely ignoring the balance will cause waste of computing resources for training and inference (Wang et al., 2024).

Expert pruning is a technique that structured pruning the model by identifying and removing experts that have minimal impact on performance during training(He & Xiao, 2023), thereby reducing model complexity and memory usage. Currently, there is relatively little work on expert pruning during the pre-training phase of MoE models, with more focus on pruning conducted post-training (Liu et al., 2024b; Lu et al., 2024; Xie et al., 2024). Given the substantial computational costs inherent in pre-training, the implementation of expert pruning during the pre-training phase is anticipated to generate benefits in multiple aspects: improving model performance, accelerating the pre-training process, lowering training costs, reduce the number of parameters in to decrease the GPU memory requirements during deployment, and ultimately mitigating carbon emissions.

In this study, We have conducted a comprehensive study on the phenomenon of expert loading imbalance in pre-training. Our contribution is summarized as follows:

- We propose an efficient expert pruning algorithm (EPA) for the pre-training phase. This algorithm reduces the number of model parameters by pruning experts with low load and enhances training efficiency through rearranging the load distribution across expert parallel groups. Distinct from conventional expert pruning methods, the proposed approach directly lowers computational costs and improve the model performance during pre-training.
- We constructed a sparse model with 1010B parameters by leveraging EPA. The training efficiency improved by 48.3%. Concurrently, the model parameter reduced by 33%. Furthermore, the 1010B pre-trained model attained accuracy comparable to that of state-of-the-art models on major tasks across different domains.

2 Related Work

2.1 Expert Pruning for MoE models

Currently, the research on expert pruning in MoE primarily focuses on the post-training stage rather than the pre-training stage. This involves pruning an already trained model for specific domain tasks, emphasizing the retention of experts that contribute the most to the target task. Lu et al. (2024)use post-training expert pruning (general + task-specific) to minimize output Frobenius norm via heuristic search, and boost inference speed with dynamic expert skipping. EEP (Liu et al., 2024b) is a gradient-free pruning for SMoE, keeping key experts and integrating pruned ones' knowledge for fine-tuning. Cluster-Driven Expert Pruning (Guo et al., 2025) does hierarchical expert clustering in MoE layers to cut redundancy, then global pruning by removing redundant clusters via unified importance scoring. It uses dynamic pruning with spectral analysis for adaptive expert merging, and updates routing to keep performance stable. The method is universal but relies on pruning criteria and data quality. MoE Pruner (Xie et al., 2024) uses weight absolute values and activation-routing weight products for one-shot pruning (no retraining, lower costs), and uses expert knowledge distillation to recover performance. Su et al. (2025) identifies super experts (via LLM massive activations) to strengthen ordinary experts, cutting parameters while keeping core performance. Both methods displays low overhead,and suit large LLM trimming and deployment. The methods mentioned above require pruning for specific datasets during the model fine-tuning or inference stages, and cannot solve the significant computational and memory burden associated with LLMs during the pre-training process.

2.2 expert load balance method in MoE models

With the advancement of Mixture-of-Experts (MoE) models, load balancing has attracted growing attention from researchers and developers. DeepSpeed-MoE optimizes the load balancing on token-level distribution across different experts and GPUs, with a flexible multi-expert and multi-data design (Rajbhandari et al., 2022). ST-MoE utilizes z-loss to enhance training stability (Zoph et al., 2022). Mixtral employs a Dynamic Token Redistribution strategy to balance the token load between experts, and alleviates the token over load with specialize sparse Megablocks (Jiang et al., 2024). OpenMoE has revealed the issue of context-independent specialization and the Drop-Towards-the-End problem of terminal tokens (Xue et al., 2024). JetMoE attempts Dropless MoE by meticulously controlling the gating to ensure that no expert exceeds the capacity limit (Shen et al., 2024). DeepSeekMoE employs expert-level balance loss and device-level balance loss to ensure the computation balance for both experts and GPUs (Dai et al., 2024). DeepSeek-V3 designs a complementary sequence-wise auxiliary loss to prevent the extreme imbalance in sequence (Liu et al., 2024a). While auxiliary loss contributes to the enhancement of load balancing, an excessively large auxiliary loss may introduce substantial perturbation gradients in the training process. Specifically, when the coefficient of the auxiliary loss is assigned an overly high value, the model will eventually demonstrate a higher perplexity, which in turn leads to suboptimal performance outcomes.In contrast to prior methods, the proposed approach refrains from the employment of supplementary auxiliary loss functions. Instead, its core focus lies in the pre-training process of LLMs leveraging an expert parallelism strategy. Within this framework, the distribution of tokens across different expert parallel groups is reorganized to attain a relatively balanced token allocation. This optimization of token distribution, in turn contributes to the enhancement of efficiency throughout the LLM pre-training workflow.

3 METHOD

3.1 EXPERT PRUNING

To investigate the load characteristics of expert within MoE LLM during the pre-training phase, two models with parameter scales of 10B and 20B are constructed, followed by the pre-training from scratch. Comprehensive details pertaining to the model architecture and pre-training dataset are provided in the Appendix. We monitor the variation in the number of tokens processed by each expert throughout the training process. It is observed that in the initial phase of training (0–100 iterations), token processing is concentrated among a small subset of experts. As training progresses (100–400 iterations), tokens become more evenly distributed across a larger pool of experts, resulting in substantial fluctuations in the workload of individual experts during this stage. Subsequently, a relatively stable period is attained, wherein the proportion of tokens allocated to each expert became relatively fixed, and the rate of change in token counts slows considerably. In this period, a substantial discrepancy in token allocation across experts is noted, with the ratio between the maximum and minimum token counts reaching approximately xx times—indicating a severe workload imbalance (Figure1).Therefore, it is reasonable to decide on the pruning of experts during the stable period.

This paper proposes a layer-adaptive effective expert pruning algorithm (EPA), the detailed implementation of which is provided in Algorithm 1. This algorithm formulates pruning strategies at the layer level, with such strategies being determined based on the distribution of tokens across experts within each individual layer. Experts targeted for pruning must satisfy the following two criteria:

- If the token load of a specific expert is lower than α of the average load of all experts, that expert shall be pruned;
- When all experts are sorted in ascending order of their respective loads, if the cumulative load of the experts with the smallest loads accounts for less than β of the total number of tokens, all such experts will be pruned.

Algorithm 1 Expert Pruning Algorithm

Require: Sequence length S, number of experts E, the number of experts selected per token topK, number of training iterations ITER, number of model layers L
α : If the token load of a particular expert is less than alpha of the average load of all experts, that expert will be pruned
β : If the cumulative load of the experts that would be pruned is less than beta of the total number of tokens, all such experts will be pruned
Step1: Recording tokens per layer per experts per train iters, ETN [ITER, L, E]
Step2: Getting the total number of pruned experts per layer, all_discard_expert[ITER, L], and the number of markers per layer per expert, discard_expert [ITER, L, E]
for train_iter in ITER:
  for layer in L:
    for expert in E
      num_token = ETN[train_iter, layer, expert]
      all_num_token += num_token
      if all_num_token $<$ S*top_K* β and num_token $<$ S*top_K * α / E:
        all_discard_expert[train_iter, layer] += 1
        discard_expert[train_iter, layer, num_token] += 1
Step3: Pruning experts after token distribution of experts getting stable
  for layer in L:
    the number of pruned experts
    discard_expert_num = all_discard_expert[iter_cut, layer]
    the index of pruned experts
    discard_expert_index[layer] =discard_expert[iter_cut, layer].
    topk(discard_expert_num, max)
Step4: Pruning experts based on the index, discard_expert_index

Table 1 illustrates the influence of α and β values on training performance, where the evaluation is conducted based on the model's loss on the test set. When α is configured to 10% and 20%, the

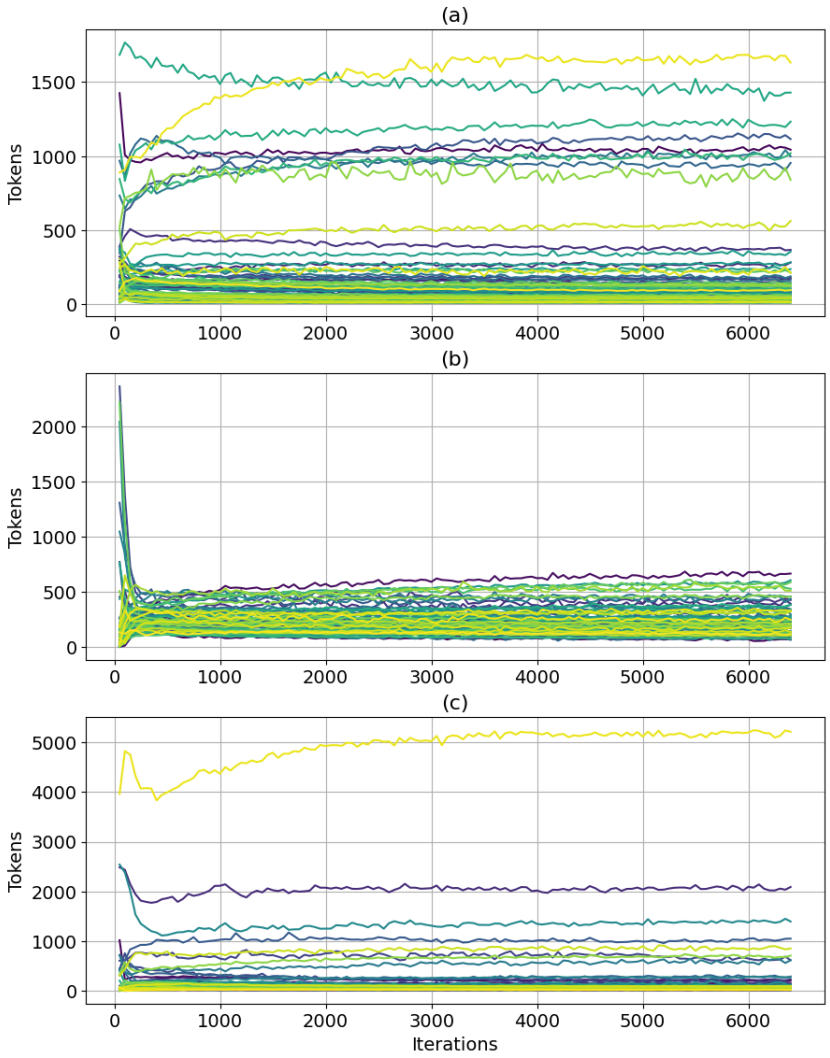

Figure 1: The variation of pruned experts (20B) number during training: (a) Layer 2; (b) Layer 24; (c) Layer 47

number of parameters decreases by approximately 1/5 and 1/3, respectively. Notably, in comparison with the unpruned model, the loss even exhibits a marginal reduction. This phenomenon arises because tokens can be redistributed among unpruned experts, which enables these experts to undergo more thorough training. As α increases to 40%, the remaining number of parameters accounts for merely 47.37% of the initial quantity. However, the accuracy on the test set only undergoes a slight decline. To balance the relationship between model accuracy and pruning ratio, α <20% is applied to the first 1/6 layers and the last 1/6 layers of the model, whereas α <40% is adopted for the middle 2/3 layers. This pruning strategy lead to the removal of approximately half of the model parameters, accompanied by only a slight decrease in the model's training accuracy (1.900). The selection of these specific alpha ratios is justified by the fact that the token distribution across

Table 1: Criteria for pruning

Pruning ratio	Params(%)	Loss (test)
No pruning	100	1.889
α <10% of average and β <10%	78.14	1.878
α <20% of average and β <10%	63.06	1.888
α <40% of average and β <10%	47.36	1.904
α <60% of average and β<10%	44.47	1.907
α <20% or 40% of average and β<10%	50.63	1.900
α <20% of average and β<5%	63.32	1.898
α <20% of average and β<15%	63.06	1.888
No pruning with auxiliary loss (Mixtral)	100	1.9149
No pruning with auxiliary loss (DeepSeek-V3)	100	1.915

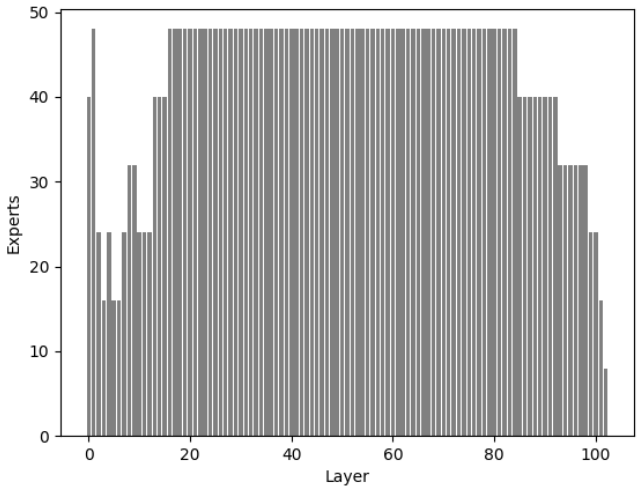

Figure 2: The number of experts in each layer of the 1010B model.

experts exhibits substantial variations in the initial and final layers of the model, while showing a more balanced pattern in the middle layers. In contrast to α, the effect of β is relatively weaker. When α remains constant and beta increases from 5% to 15%, both the number of model parameters and the validation set loss display only negligible changes. Additionally, this study compares the effectiveness of introducing auxiliary losses , specifically focusing on the load-balancing auxiliary loss function utilized by Mixtral 7B, as well as the expert-level and device-level losses adopted by DeepSeek-V3. The results demonstrate that regardless of the auxiliary loss scheme implemented, a negative impact on model accuracy is observed. Furthermore, the accuracy with auxiliary loss even drops below that of the model with its parameter count reduced to 44% of the original (i.e., α=60%, β=10%).

3.2 Robustness across tasks

To validate the influence of the EPA on the accuracy of downstream tasks, we construct a model with 20 billion parameters and conduct pre-training on a corpus of 100 billion tokens. The model's detailed architectural specifications and the training protocol (including hyperparameter settings, optimization strategies, and data) are provided in Appendix for reproducibility.

Table 2 reports the 20B model's performance across downstream tasks under different experimental conditions, specifically varying pruning percentages and auxiliary loss configurations.We have evaluated the accuracy of the 20B model across five types of tasks (Mathematics: Cmath, GSM8k; Question Answering: TriviaQA, NatureQuestions; Coding: CRUXEval-I). The unpruned base model exhibited the lowest average accuracy among all tested configurations. After integrating the auxiliary loss from DeepSeek-V3, the model's average accuracy was marginally lower than that of the base model. This trend is consistent with the accuracy pattern observed for the 10B model on the same test set. When the EPA is applied with α of 20%, the model's average accuracy showed an improvement compared to the base model. This result also aligns with the accuracy performance of the 10B model on the test set. Notably, a more optimized pruning strategy yielded further gains: when pruning the first 1/6 layers and the last 1/6 layers of the 20B model with α of 20%, while applying α of 40% to the middle 2/3 layers, the model's average accuracy slightly exceed that of the configuration where a uniform α of 20% is apply across all layers. Relative to the unpruned base model, the model optimized with the aforementioned layered pruning strategy achieved an 18.9% improvement in accuracy, accompanied by a reduction of 13% parameters. These results demonstrate the effectiveness of the proposed pruning method. Our key findings are summarized as follows:

(1) The model's accuracy exhibits no statistically significant variation before and after pruning, which directly corroborates the feasibility of the EPA framework;

(2) Even with an increased pruning ratio, the degradation in accuracy remains within an acceptable range, indicating the robustness of the proposed pruning strategy;

(3) The auxiliary loss scheme under comparison fails to outperform our approach in terms of accuracy, suggesting the superiority of our design in preserving task performance;

(4) The model demonstrates consistent performance across diverse task domains (including natural language processing, knowledge-based question answering (QA), code generation, and mathematical reasoning) with no notable performance deficiencies observed pre- and post-pruning. This result confirms that our model pruning process does not introduce unintended biases during dataset construction.

3.3 Expert rearranging

Pruning a MoE LLM by removing lightly loaded experts contributes to improve load balancing across the remaining experts. Nevertheless, in practical pre-training scenarios, loading imbalance issues between computing nodes still persist unresolved, which arises from the adoption of parallelization strategies. With the continuous growth in the model scale, MoE LLMs typically leverage expert parallelism for pre-training to accommodate computational demands. In the framework of expert parallelism, the experts within a model are partitioned into distinct groups. Specifically, experts belonging to the same group are usually deployed within a high-speed interconnection domain, and this configuration is intended to optimize communication efficiency, which is a critical factor for the performance of expert parallelism. In contrast, experts from different groups are generally distributed across separate computing nodes. If significant load imbalance occurs between these expert groups, nodes with low computational loads may complete their assigned tasks prematurely and enter an idle state, waiting for nodes burdened with heavy loads to finish. This asynchronous completion of tasks ultimately compromises the overall computational efficiency of the pre-training process.

To mitigate this node-level load imbalance issue, we propose an expert rearranging algorithm (see Figure 3). This algorithm rearranges the allocation of experts across parallel groups based on the token distribution patterns among the expert parallel groups. With this strategy, it ensures that tokens are distributed relatively evenly across different expert parallel groups, thereby alleviating inter-node load imbalance and optimizing pre-training efficiency.

4 Construct a 1010B model with EPA

The architectural infrastructure of the 1010B model bears significant similarities to that of Yuan 2.0-102B (Wu et al., 2023). A key modification lies in the comprehensive replacement of the Multilayer Perceptron (MLP) module with an expert system, while adopting a routing algorithm configured

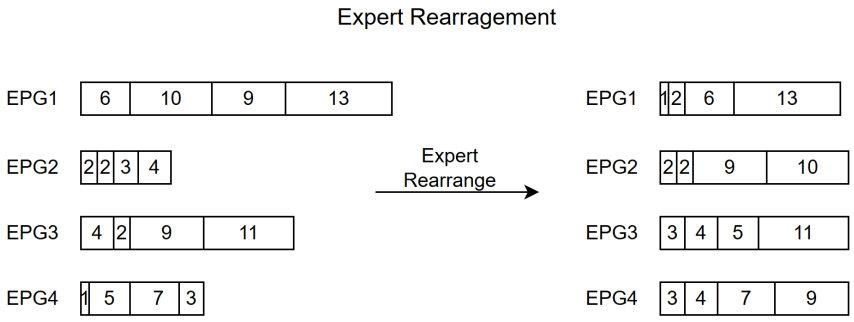

Figure 3: Expert Rearrangement Diagram. The left panel illustrates the original distribution of experts across the Expert Parallel Groups (EPGs), revealing a significant load imbalance. The right panel depicts the distribution of experts among the EPGs following redistribution.

Table 2: A Comparative Analysis of Expert Load Balancing Solutions Across Different Tasks (20B)

	Base Model	**With auxiliary loss (DeepSeek-V3)**	**EPA** α<20%	**EPA** α<20% **or 40%**
Activated Parameters	0.9B	0.9B	0.9B	0.9B
Total Parameters	20B	20B	18.9B	17.4B
Cmath	22.5865	15.4827	24.1348	24.0437
GSM8K	7.8089	8.1122	8.5671	10.1592
TriviaQA	12.0181	13.1519	14.5125	16.0998
NaturalQuestions	8.0679	8.3106	11.1049	11.2031
CRUXEval-I	6.8836	6.8836	8.1352	6.6333
Average	11.47	10.39	13.29	13.63

with a top-K value of 2. For detailed parameters pertaining to the model structure, reference may be made to Appendix.The 1010B model is derived from the pruning of a base MoE model with 103 layers and 64 experts. After pruning, the number of model layers remains unaltered, whereas the total parameter count is reduced from 1515B to 1010B, representing an approximate one-third reduction in parameters compared to the pre-pruned baseline. This model is trained on a corpus of 1.4 Trillion tokens; additional specifics regarding the training data are available in the Appendix.

Figure 2 depicts the distribution of experts in the 1010B model following the pruning process. As observed, a spindle-shaped structure emerges along the pathway from input to output: the input-side layers and middle layers exhibit a higher number of retained experts. A prominent expert pruning region is identified between the input layers and middle layers. The proportion of pruned experts gradually increases as moving from the middle layers toward the output layers, ultimately resulting in the lowest number of retained experts in the output layer. The pruning rate is assigned in accordance with the distinct characteristics of each layer. Specifically, layers with a more uniform token distribution are retained with a larger number of experts, whereas layers with a more skewed token distribution undergo maximal pruning of redundant experts. This non-uniform pruning strategy is designed to maximize the preservation of model performance. Notably, the specific criteria for pruning were determined through preliminary experiments conducted on the 10B model.

As illustrated by the performance data presented in Table 4, the model's training performance exhibits a substantial improvement following the application of expert pruning and rearranging. Specifically, its computational performance rises from 62.14 TFlops/GPU to 92.6 TFlops/GPU, corresponding to a remarkable performance enhancement of 49%. The integration of auxiliary loss also contributes to the improvement of training performance, with the model achieving 80.36 TFlops/GPU and 80.82 TFlops/GPU respectively under this configuration. Nevertheless, a comparative analysis reveals that the performance of the DeepSeek-V3 load balancing method is 12.7%

Table 3: Performance comparison of the base model 1515B (activated 68.5B) under different load balancing methods and expert pruning and rearranging methods.The tests are conducted on 824 A800 chips, with a stream parallel dimension of 103 and a data parallel dimension of 4, while the numerical precision during the training process is set to BF16.

Total Params	Expert load Balancing	layers	Experts	EPG	TFLOPS
1515B	-	103	64	4	62.14
1515B	Mixtral Auxiliary loss	103	64	4	80.36
1515B	DeepSeek-V3 Sequence-Wise Auxiliary Loss	103	64	4	80.82
1142B	DeepSeek-V3 Sequence-Wise Auxiliary Loss	103	48	2	84.74
1010B	EPA ($\alpha = 0.2$,$\beta = 0.1$)	103	48	2	82.25
	+rearrangement	103	48	2	92.60

Table 4: Comparison on tasks between the 1010B model and representative dense and MoE base models.

	Benchmark	#Shots	LLaMA-3.1 405B Base	DeepSeek-V3 Base	1010B
	Architecture		Dense	MoE	MoE
	# activated params		405B	37B	68.5B
	# total params		405B	671B	1010B
language	Pile-test	-	0.542	0.548	0.594
	MMLU	1-shot	84.4 (5-shot)	87.1(5-shot)	77.6
	ARC-Challenge	0-shot	95.3(25-shot)	95.3	93.8
	TriviaQA	1-shot	82.7(5-shot)	82.9(5-shot)	75.46
	NaturalQuestions	1-shot	41.5(5-shot)	40(5-shot)	43.3
Code	HumanEval	0-shot	54.9	65.2	70.7
	MBPP	3-shot	68.4	75.4	75.9
Math	GSM8K	8-shot	83.5	89.3	86.1
	MATH	4-shot	49	61.6	66.1

lower than that of the EPA method. With respect to the parameter count data in Table 4, the adoption of the EPA method leads to a significant reduction in the model's parameter scale.The number of parameters decreases from 1515B to 1010B, reflecting a 33% reduction. This parameter optimization not only effectively alleviates the memory consumption during the pre-training phase but also reduces the memory requirements for subsequent deployment processes, thereby enhancing the model's practical applicability. For the purpose of conducting a more comprehensive comparative study, additional experiments are designed: a 1515B model is tested under the condition where the original number of experts is reduced from 64 to 48 (while keeping the maximum number of experts consistent with that after EPA pruning), and the load balancing method of DeepSeek-V3 is adopted in this experimental setup. The results of this control experiment demonstrate that the EPA method still enables a 9.3% increase in training performance and an 11.6% reduction in the number of parameters, further verifying the effectiveness and superiority of the EPA method in optimizing model performance and reducing parameter complexity.

We compared the accuracy of our pre-trained 1010B model with that of the DeepSeek-V3 Base model and the Llama3.1-405B Base model. It can be observed that our model achieves leading accuracy on the HumanEval and MBPP code evaluation tasks, as well as the Math mathematics task. On the ARC-Challenge reasoning task, the accuracy of our model is slightly higher than that of DeepSeek-V3 but lower than that of LLaMA-3.1-405B. Additionally, on the Pile-test, MMLU, TriviaQA, and NaturalQuestion tasks, the accuracy of our model is slightly lower than that of both DeepSeek-V3 and LLaMA-3.1-405B.

5 CONCLUSION

In this paper, we propose an efficient Expert Pruning Algorithm (EPA) for the pre-training of MoE LLMs. This method effectively enhances the computational performance of the model, improves the accuracy on downstream tasks, and significantly reduces the model's parameter count. We have successfully applied the EPA method to the pre-training of a 1010B model. Compared with the state before pruning, the proposed method achieves a 49% improvement in computational performance and a 33% reduction in model parameters. Meanwhile, the accuracy of the resulting model is comparable to that of cutting-edge models in the industry, such as DeepSeek-V3.1 Base and LLaMA3.1-405B Base.

REFERENCES

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

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.

Hongcheng Guo, Juntao Yao, Boyang Wang, Junjia Du, Shaosheng Cao, Donglin Di, Shun Zhang, and Zhoujun Li. Cluster-driven expert pruning for mixture-of-experts large language models. *arXiv preprint arXiv:2504.07807*, 2025.

Yang He and Lingao Xiao. Structured pruning for deep convolutional neural networks: A survey. *IEEE transactions on pattern analysis and machine intelligence*, 46(5):2900–2919, 2023.

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

Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, et al. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*, 2024a.

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

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. *arXiv preprint arXiv:2402.14800*, 2024.

Samyam Rajbhandari, Conglong Li, Zhewei Yao, Minjia Zhang, Reza Yazdani Aminabadi, Ammar Ahmad Awan, Jeff Rasley, and Yuxiong He. Deepspeed-moe: Advancing mixture-of-experts inference and training to power next-generation ai scale. In *International conference on machine learning*, pp. 18332–18346. PMLR, 2022.

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

Zunhai Su, Qingyuan Li, Hao Zhang, YuLei Qian, Yuchen Xie, and Kehong Yuan. Unveiling super experts in mixture-of-experts large language models. *arXiv preprint arXiv:2507.23279*, 2025.

Lean Wang, Huazuo Gao, Chenggang Zhao, Xu Sun, and Damai Dai. Auxiliary-loss-free load balancing strategy for mixture-of-experts. *arXiv preprint arXiv:2408.15664*, 2024.

Shaohua Wu, Xudong Zhao, Shenling Wang, Jiangang Luo, Lingjun Li, Xi Chen, Bing Zhao, Wei Wang, Tong Yu, Rongguo Zhang, et al. Yuan 2.0: A large language model with localized filtering-based attention. *arXiv preprint arXiv:2311.15786*, 2023.

Table 5: Structural parameters of the models

Model	10B	20B	1010B
Num of activated parameters	0.52B	1.18B	69.7B
Num of layers	12	48	103
Num of experts before pruning	64	64	64
Hidden size	1024	1024	1024
FFN hidden size	4096	2048	16384
Attention hidden size	4	4	36
Num of attention heads	256	256	256

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

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

Barret Zoph, Irwan Bello, Sameer Kumar, Nan Du, Yanping Huang, Jeff Dean, Noam Shazeer, and William Fedus. St-moe: Designing stable and transferable sparse expert models. *arXiv preprint arXiv:2202.08906*, 2022.

A Appendix

A.1 Text Corpus for Pre-training

We mainly use a large-scale bilingual (EN-CN) language corpus in the pre-training stage. Approximately 60% of the corpus are in English, and 40% are Chinese. The mathematical data focuses on adding data related to numerical calculations, formula derivation, and mathematical applications. In addition, we expand the code data on a large scale. The text corpus is selected and combined based on classification, which reduces the proportion of news and commentary data while retaining the proportion of professional data (including advertising, economy, healthcare, law, business, literature, culture, history, politics, art, entertainment, knowledge, viewpoints, education, science, technology et al..). Our work shows that a larger proportion of data are necessary to support the model in pre-training to obtain satisfactory code capabilities, and that appropriately reducing text data, especially the reduction of low-quality text data (e.g. common crawl), will not have much impact on the model's language capabilities.

A.2 Hyper-parameters of model structure