XMoE: Sparse Models with Fine-grained and Adaptive Expert Selection

Anonymous ACL submission

Abstract

Sparse models, including sparse Mixture-of-Experts (MoE) models, have emerged as an effective approach for scaling Transformer models. However, they often suffer from computational inefficiency since a significant number of parameters are unnecessarily involved in computations via multiplying values by zero or low activation values. To address this issue, we present XMoE, a novel MoE designed to enhance both the efficacy and efficiency of sparse MoE models. XMoE leverages small experts and a threshold-based router to enable tokens to selectively engage only essential parameters. Our extensive experiments on language modeling and machine translation tasks demonstrate that XMoE can enhance model performance while decreasing the computation load at MoE layers by over 50% without sacrificing performance. Furthermore, we present the versatility of XMoE by applying it to dense models, enabling sparse computation during inference. We provide a comprehensive analysis and make our code available at https://anonymous.4open.science/r/XMoE.

1 Introduction

Recently, remarkable advancements in large language models have been achieved through scaling up their sizes (Brown et al., 2020; Chung et al., 2022; Touvron et al., 2023). However, this progress has come with a significant increase in training costs, posing a challenge to further scaling. To address this issue, sparse models, such as sparse Mixture-of-Experts (MoE) models, have emerged as an alternative approach. These models allow for scaling the model size without a corresponding increase in computational cost (Shazeer et al., 2017; Lepikhin et al., 2021; Jaszczur et al., 2021; Fedus et al., 2022; Rajbhandari et al., 2022). The key to this ability lies in sparsely activated MoE layers. These layers comprise multiple sub-networks, or "experts", typically implemented as Feed-Forward Networks (FFNs) (Vaswani et al., 2017). Unlike traditional models that utilize all parameters for each input token, MoE models selectively activate a subset of experts. This approach effectively decouples computational costs from model size, paving the way for more efficient scaling.

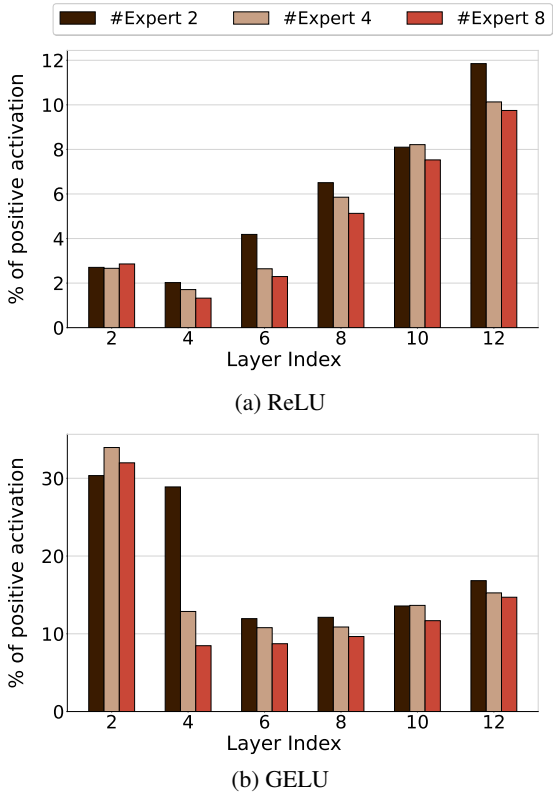


Figure 1: Average percentage of positive values in the FFN layers after the activation function. All models are decoder-only Transformers with 12 layers.

While MoE models are effective for scaling, this paper argues that MoE models exacerbate the issue of computational inefficiencies. Initially identified in the dense model T5 (Raffel et al., 2020), this issue is characterized by a significant portion of computations in the FFN layer being wasted on multiplying values by zero (Zhang et al., 2022; Li et al., 2023b). As illustrated in Figure 1, the compu-

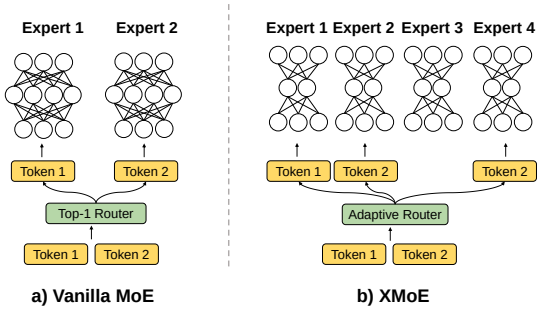


Figure 2: Overview of an MoE layer in XMoE, where tokens are routed to small experts by an adaptive router.

tational inefficiency issue is also prevalent in sparse models and even worsens as the number of experts increases. This observation suggests that only a small portion of the parameters in an expert is useful for the input, while others are unnecessarily involved in the computation. Consequently, selecting one expert for each input can already lead to a significant waste of computation. In order to alleviate this problem, a more fine-grained and adaptive strategy for parameter selection is necessary.

Figure 2 shows the overview of a novel MoE design, named XMoE, which allows tokens to select fewer parameters to improve the efficiency without hindering model performance. To achieve this, XMoE proposes to exploit small experts and a threshold-based router. First, considering that expert is the smallest unit of parameter selection in MoE models, utilizing small experts is the prerequisite for the more fine-grained selection. It allows models to choose the useful parameters precisely without activating the redundant parameters.

In order to ensure the effectiveness, a novel adaptive router is further exploited by XMoE. Different from the widely-used top-k router that dispatches each input to a fixed number of experts, this adaptive router allows tokens to self-determine the number of required experts based on a pre-defined threshold. Intuitively, an easily processed token can be routed to a single small expert, while a critical token may require multiple experts (Zhou et al., 2022). An adaptive router allows models to leverage the difference of the input complexity to dynamically allocate computational resources. This not only enhances model efficiency but also yields potential quality improvements when computational resources are constrained.

In conjunction with the aforementioned design, XMoE aims to enhance the efficiency of MoE models, with a focus on improving quality with a fixed computational budget or reducing computational costs without compromising performance. Extensive experiments in language modeling and machine translation demonstrate performance gains over existing MoE methods. XMoE also enables a reduction in Floating Point Operations (FLOPs) by over 50% with minimal impact on performance. Additionally, our investigation extends to training dense models using XMoE to facilitate sparse computation during inference. This approach not only matches the performance of dense counterparts but also facilitates a substantial reduction in FLOPs.

Our contributions can be summarized as: (1) We identify a computational inefficiency issue in current sparse MoE models. (2) We propose XMoE to improve the efficiency of MoE models with small experts and a novel routing strategy. (3) Extensive experiments in language modeling and machine translation demonstrate XMoE as a promising alternative to existing sparse and dense models.

2 Background

2.1 Mixture-of-Experts (MoE)

MoE is a family of neural network architecture that enables conditional computation by sparsely activating small sub-networks, so-called experts, based on a pre-defined router. The core component of MoE models is the MoE layer, which consists of a set of experts $\{E_1, \cdots, E_N\}$ and a routing function (Zoph et al., 2022). Each expert E is a parameterized function. The routing function is utilized to route individual tokens to their assigned experts. In this work, we consider MoE for Transformer, where FFN layers within Transformer are substituted with large MoE layers, in which each expert is an independent FFN.

Trainable Router. A commonly used router is based on a gating network consisting of a trainable weight matrix followed by a softmax function. For the input token x with its intermediate representation denoted as $h \in \mathbb{R}^d$, the router computes the probability distribution over experts as:

$$p = \mathrm{Softmax}(W \cdot h), \tag{1}$$

where $W \in \mathbb{R}^{N \times d}$, N denotes the total number of experts and d denotes the hidden size of the model. The set of top-k experts $\mathcal{E}$ is decided based on p, where $|\mathcal{E}| = k$. Each expert processes tokens independently. Then, the final output of the token

is the weighted sum of the output of its k experts:

$$y = \sum_{i \in \mathcal{E}} p_i E_i(h). \tag{2}$$

Load Imbalance. Learnable routers can easily cause the load imbalance issue, as shown in Shazeer et al. (2017). If there is no constraint employed during training, most tokens would be dispatched to a small number of experts, leading to a large portion of experts being insufficiently trained (Lepikhin et al., 2021). In addition, if experts are distributed across different nodes, most nodes must wait for others to finish the computation, thus hindering the training efficiency.

Capacity. Expert capacity is introduced to MoE models to avoid the severe load imbalance issue by limiting the maximum number of tokens routed to each expert (Lepikhin et al., 2021; Fedus et al., 2022; Rajbhandari et al., 2022; Puigcerver et al., 2024). Suppose a given batch's token number is T, and the expert number is N. The expert capacity C is:

$$C = \frac{T}{N} \cdot \gamma \tag{3}$$

where γ refers to the preset capacity factor. If an expert is underutilized, the unused capacity buffers are filled with padding tokens. Once an expert is at capacity, additional tokens are dropped, which means being passed to the subsequent Transformer block directly.

MoE for Dense Models. Although MoE models are proposed for training large-scale sparse models, recent studies bring the concept of MoE for either pruning or training dense models. Zhang et al. (2022) propose to divide the feed-forward network layers in a pretrained dense Transformer into several small experts based on heuristic strategies. Each input token selectively activates top-k experts instead of utilizing the whole FFNs. Chen et al. (2023) propose to view MoE as a regularization method for training dense Transformers. Although our work proposes XMoE for sparse model training, it can also be utilized for training dense models to enable sparse computation during inference.

2.2 FFN as Memory

Recent studies suggest that an FFN layer can be conceptualized as a memory layer composed of numerous key-value memory pairs (Lample et al., 2019; Geva et al., 2021). In this view, each column in the first matrix of an FFN layer is a key vector, and the corresponding row in the second matrix is the value vector. It is observed that only some memory values benefit an input token (Dai et al., 2022), leading to most memory pairs contributing to redundant computations. We hypothesize that in MoE models, which incorporate multiple FFNs, the beneficial memories tend to be distributed across different FFNs. This dispersion of useful memories diminish overall efficiency.

3 Method

3.1 FFN Decomposition

Considering that widely-used activation functions such as ReLU and GELU operate on an element-wise basis, it is reasonable to conceptualize an FFN layer as a composition of several smaller FFN layers

$$y = W_2\, \sigma(W_1 \cdot x) = \sum_i W_2^i\, \sigma(W_1^i \cdot x), \tag{4}$$

where W_j^i is the parameters of i-th small FFN layer and σ denotes the activation function. Instead of maintaining a single large FFN, XMoE trains multiple small FFNs and compose them to produce an output. In subsequent discussions, we assume that the dimensions are identical for each experts unless stated otherwise. Our preliminary experiments showed that the GELU activation function performed slightly better. Therefore, we will use GELU by default.

3.2 Threshold-based Router

XMoE's router consists of a trainable weight matrix $W \in \mathbb{R}^{d \times n}$, where each column of W serves as a centroid representing an expert. Given an input token x and its intermediate representation $h \in \mathbb{R}^d$, the probability of choosing the i-th experts is p_i, according to the Eq (1).

Considering the distribution p varies with layers, tokens and model configurations, manual selection of an optimal value for the top-k parameter k can be challenging (Yang et al., 2021; Li et al., 2023a). A higher value of k can ensure effectiveness but at the cost of increased computational overhead per token, potentially impacting efficiency. Conversely, a lower value of k may reduce computational load but restrict the model's capacity to handle complex tokens. To navigate this trade-off, XMoE employs a threshold-based selection mechanism, enabling

tokens to self-determine the number of experts they should be routed to based on a predefined threshold parameter t, which ranges from 0 to 1.

The overview of the selection procedure is illustrated in Figure 3. Initially, the probabilities p are sorted in descending order: $p = [p_{i_1}, p_{i_2}, ..., p_{i_n}]$, where $p_{i_1} \geq p_{i_2} \geq ... \geq p_{i_n}$ and i_j indicates the index of top-j expert. The router then identifies the smallest index m such that the cumulative sum of probabilities up to index m is greater than or equal to the threshold t:

$$\arg\min_{m} \sum_{j=1}^{m} p_{i_j} \geq t. \tag{5}$$

The input tokens are dispatched to their first m experts. According to the Eq (2), an expert's contribution to the output is directly proportional to its assigned probability p. Thus a high cumulative sum of probabilities indicates that tokens are routed to the most relevant experts, while other experts with lower probabilities have minimal influence and can be ignored.

Priority. The threshold-based router in MoE models permits a token to select multiple experts, potentially overwhelming certain experts and leading to dropped tokens due to limited expert capacities. To mitigate this, XMoE assigns a priority r to a token dispatched to a specific expert. The experts process tokens with higher priorities first. Specifically, the prioritization is based on the likelihood of a token considering the expert as its preferred choice, determined by a heuristic rule. If a token views an expert as its top-i choice with probability p, the priority r is set heuristically as $(p - i)$. Appendix A.1 presents the algorithmic implementation.

3.3 Auxiliary Loss

Following Fedus et al. (2022), we utilize an auxiliary loss as a part of the training objective. It encourages input tokens to be uniformly dispatched to the experts. We only impose this constraint on the top-1 assignment. The details are provided in Appendix A.2.

3.4 Complexity

Within an MoE layer, a total of $C \cdot N$ tokens are processed by the experts, where C denotes the capacity and N is the number of experts. Let $O(E)$ represents the computational complexity associated with an expert handling a single token. The computational complexity of an MoE layer, as indicated

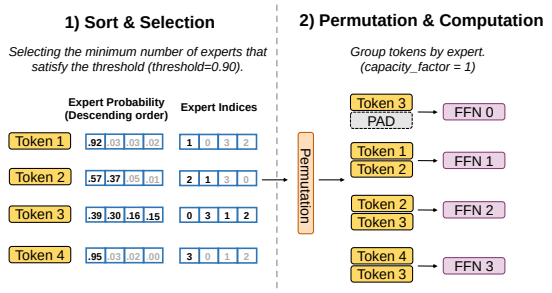


Figure 3: Overview of the threshold-based router. The number of tokens processed per expert is determined according to both the total token number and the capacity factor.

by Eq. 3, can be expressed as:

$$O(\text{MoE}) \propto \gamma \cdot T \cdot O(E). \tag{6}$$

A small γ can reduce the computational cost but aggravate token dropping since experts may not have enough capacity to process received tokens. On the contrary, increasing γ could lead to additional computational overhead and tend to waste computing resources on padding tokens. XMoE allows to trade off the efficacy and efficiency by concurrently increasing γ and diminishing $O(E)$.

3.5 Comparison with Top-k Router

Top-1 Router. The top-1 router exclusively assigns each token to a single expert. It potentially leads to inefficiencies when experts receive fewer tokens than their capacity, resulting in wasted computations processing padding tokens. In contrast, the threshold-based router dispatch each token to at least one expert. Consequently, the token assignment of the top-1 router is a subset of that produced by the threshold-based router. This enables models with the threshold-based router to implicitly reduce computational inefficiencies.

Top-k Router. Top-k router allows each token to choose multiple experts, thus unlikely leading to computation waste, especially when the capacity is limited. However, this router processes tokens with the same amount of computation, regardless of the difference in complexity between tokens. In contrast, an adaptive router allows the model to dynamically allocate computational resources to important tokens while reducing unnecessary computations, leading to better utilization of training computation.

The threshold-based router and the top-k router share parameters of the same shape, with their only

distinction lying in their respective selection strategies. This structural similarity indicates that the threshold-based routing strategy and the top-k strategy can be seamlessly interchanged without necessitating any modifications.

3.6 Dense Model Applications

A dense model can be treated as an MoE model with only one expert per layer. XMoE can be applied by decomposing FFN layers in the dense model into multiple smaller ones. In contrast to training a sparse model, we *densely* train this model by setting the threshold $t = 1.0$ and γ as N. This setup ensures that each token is processed by all experts without token dropping. It is expected that the router can learn to measure the importance of different experts. During inference, both t and γ can be adjusted to enable sparse computation.

4 Experiments

4.1 Tasks and Datasets

Language Modeling. We pretrain models on two English-language datasets, OpenWebText (Gokaslan and Cohen, 2019) and Wikitext-103 (Merity et al., 2017). The former is a public reproduction of WebText used for GPT-2 training (Radford et al., 2019). The latter is a smaller language modeling corpus containing Wikipedia articles.

Machine Translation. We have collected 5 language pairs from WMT23 datasets. Each language pair is trained independently. The detailed statistics for these language pairs can be found in Table 4.

4.2 Model Configuration

Language Modeling. We adopt the Transformer decoder with 12 layers. Our implementation is based on Megatron-LM (Shoeybi et al., 2019). We use the tokenizer of GPT-2 of which the vocabulary size is $50,257$. The model is trained for 60K steps in total on OpenWebText and 4k steps on Wikitext-103. The learning rate is 4.5×10^{-4} and the cosine learning rate decay is exploited. During training, we use float16 for acceleration. The threshold t is set to 0.90 based on the results of pilot experiments.

Machine Translation. We adopt the Transformer-based architecture with 12 encoder layers and 6 decoder layers. Our implementation is based on fairseq (Ott et al., 2019). The vocabulary is learned from the training data for each language pair using byte-pair encoding. The threshold t is set to 0.90. Automatic mixed precision is enabled to accelerate training. We report the detokenized BLEU and chrF2^{++} using sacreBLEU[1].

For sparse models, an MoE layer is utilized to replace the dense FFN layer in every alternate Transformer block, following previous practice (Zhou et al., 2022) . The capacity factor γ is set to 1.0 for models with top-1 routers. For models with smaller experts, we increase the capacity factor according to Eq (6) to ensure the consumed training compute identical. We run our experiments on one node with 4 NVIDIA A100 GPUs.

4.3 Baselines

Top-1 routing, which assigns each token to one expert, is widely used in MoE models, such as Switch Transformers (Fedus et al., 2022), BASE Layers (Lewis et al., 2021) and Hash Layers (Roller et al., 2021). Switch Transformer utilizes a learnable top-1 router with an auxiliary loss to alleviate the load imbalance issue. BASE Layer proposes to view the token assignment as a linear assignment problem. They force each expert to process an equal number of tokens during training while using greedy assignments at test time. Hash Layers replace the learnable router with simple hash functions. We implement Hash Layers with a random hash function, which is suggested to have strong performance (Roller et al., 2021). We extend the top-1 router of Switch Transformer to the top-k ($k > 1$) router.

5 Results

5.1 Language Modeling

Table 1 reports the perplexity results of models with equivalent computational complexity in MoE layers. A consistent performance gain is observed across both the OpenWebText and WikiText-103 datasets when reducing the expert size. Specifically, when the expert size decreases to 384, XMoE with 323M parameters achieves the best performance. It surpasses top-k routing and Switch Transformer by 0.33 and 0.65 perplexity points on OpenWebText, respectively. The difference between XMoE and top-k routing lies in the routing strategy, while the difference between top-k routing and Switch Transformer lies in the expert size. This suggests that a threshold-based router can assist models in

[1] BLEU:*nrefs:1 | case:mixed | eff:no | tok:13a | smooth:exp | version:2.3.2*. chrF2:*nrefs:1 | case:mixed | eff:yes | nc:6 | nw:2 | space:no | version:2.3.2*

Params	Method	Top-k	# Experts	Expert Size	OpenWebText	WikiText-103
124M	Dense Transformer	1	1	3072	22.61	22.24
323M	Switch Transformer	1	8	3072	20.11	20.60
	Hash Layer	1	8	3072	21.27	21.63
	BASE Layer	1	8	3072	20.49	21.13
	Top-k Gating	2	16	1536	19.81	20.24
		4	32	768	19.75	20.14
		8	64	384	19.79	20.10
	XMoE	-	16	1536	19.57	20.16
			32	768	19.49	20.00
			64	384	**19.46**	**19.97**
556M	Switch Transformer	1	8	4096	19.17	19.72
	Hash Layer	1	8	4096	20.25	22.01
	BASE Layer	1	8	4096	19.72	21.03
	Top-k Gating	2	16	2048	18.94	19.47
		4	32	1024	18.94	19.36
		8	64	512	18.99	19.15
	XMoE	-	16	2048	18.72	19.41
			32	1024	**18.68**	19.26
			64	512	18.72	**19.10**

Table 1: Test perplexity$^{\downarrow}$ on OpenWebText and WikiText-103. Models consume approximately the same training and inference FLOPs through the adjustment of γ according to Eq. 6. The "Top-k" column denotes the number of selected experts per token, and "-" denotes not applicable.

leveraging expert capacities, and smaller experts can enhance model quality. There are intriguing outcomes when we increases the parameter count to 556M by expanding the intermediate dimension of experts. It is shown that top-k routing with a size of 1024 outperforms that with a size of 512 on OpenWebText, as does XMoE. We attribute this to the optimization of sparse models and leave this for future investigation.

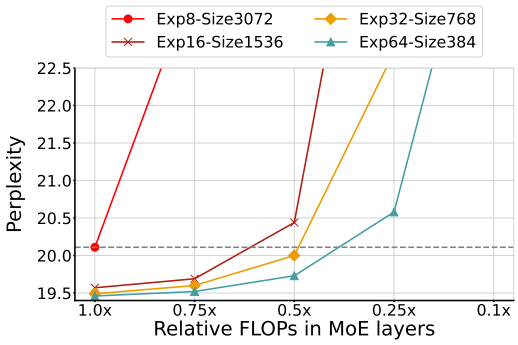
Efficiency. Figure 4 illustrates the impact of the number of floating point operations (FLOPs) within MoE layers on perplexity score. We see that models exhibit poorer performance as FLOPs decrease. This trend is expected, as limited expert capacity can lead to more dropped tokens, thereby increasing the number of inadequately processed tokens.

It is observed that XMoE with small expert size is robust to the reductions in FLOPs. Notably, XMoE with an expert size of 384 outperforms Switch Transformer (referred to as Exp8-Size3072) on WikiText-103, while consuming only 25% of the FLOPs of the latter. This finding suggests that a significant portion of parameters in models with large expert sizes are unnecessarily engaged in computations. By replacing these large experts with smaller ones, XMoE not only improves performance but also saves computational resources.

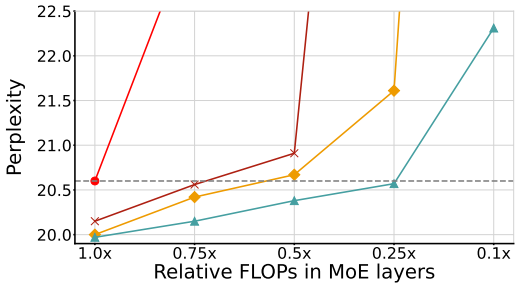
Method	# Experts	Size	FLOPs	OpenWeb	Wiki
Dense	1	3072	1.00x	22.61	22.24
XMoE	8	384	1.00x	22.89	21.93
			0.75x	22.89	21.93
			0.50x	22.90	21.93
			0.25x	23.07	21.97
	16	192	1.00x	22.94	21.92
			0.75x	22.97	21.92
			0.50x	23.02	21.92
			0.25x	23.27	21.93

Table 2: Test perplexity on OpenWebText and WikiText-103.

Dense Model. Dense models can be trained as XMoE by partitioning their FFN layers into smaller ones. As shown in Table 2, these modified models exhibit a remarkable reduction of over 50% in computational complexity with only a marginal decrease in performance across both datasets. On the WikiText-103 dataset, XMoE can outperform the dense model while achieving a 75% reduction in



(a) OpenWebText



(b) WikiText-103

Figure 4: Test perplexity (PPL) with regard to the the normalized FLOPs in the MoE layer during inference. We adjust FLOPs by modifying the capacity factor γ.

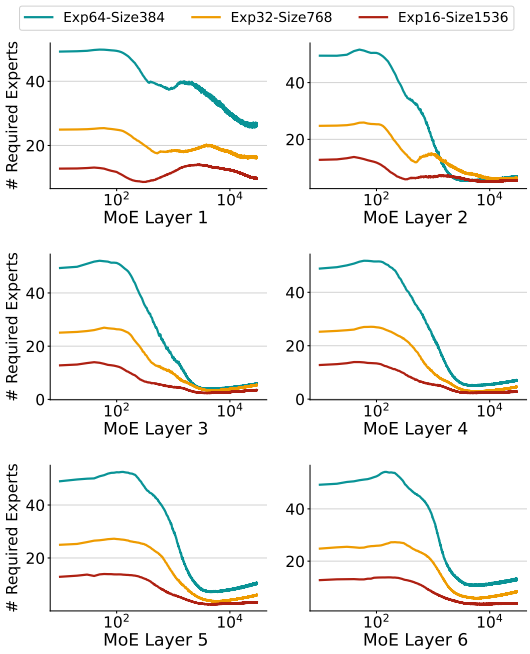


Figure 5: Average number of required experts with regards to training steps across different layers.

Floating Point Operations (FLOPs). These findings underscore the potential of sparse models as promising alternatives to their dense counterparts.

5.2 Machine Translation

We compare XMoE with Switch Transformer and Top-k routing. Models are trained to translate other languages to English. All models have the same number of shared parameters and have the same FLOPs. The results are detailed in Table 3, revealing that XMoE outperforms its counterparts. The observations generally align with the finding on language modeling that a reduction in expert size can always lead to better performance. However, the extent of the improvement depends upon factors such as the specific language being translated, and the evaluation metrics employed.

5.3 Analysis

Efficiency. Figure 5 shows the number of experts that tokens require to satisfy the threshold vs. training step. Initially, a substantial portion of experts is required to satisfy the threshold, but as training progresses, a significant reduction in the required number of experts is observed. This phenomenon is consistent with existing research indicating that dense models exhibit sparse activation once trained (Li et al., 2023b). Our proposed XMoE leverages this emergent sparsity to to enhance computational efficiency by employing small experts and a threshold-based router. The use of small-scale experts facilitates fine-grained expert selection, thereby mitigating the activation of redundant parameters. Concurrently, the threshold-based router encourages tokens to select the fewest experts. While the threshold-based routing may initially lead to an excessive selection of experts, the computations is supposed to exhibit sparsity after training.

Effectiveness. The visualization in Figure 6 illustrates the average percentage of positive values after the activation function across different configurations. It can be seen that a reduction in expert size consistently correlates with an increase in this percentage. This trend suggests that models with small experts can more effectively leverage the parameters within selected experts, thereby yielding performance improvements.

5.4 Hyperparameters

Threshold. We investigate the effect of threshold t on the perplexity score during both training and inference stages. From Figure 7, we see that increasing the threshold generally improves performance. However, XMoE with a threshold of 0.9 slightly outperforms the model with a threshold

Method	#Experts	Size	Uk-En		De-En		Ru-En		He-En		Zh-En		Avg	
			BLEU	*ChrF*	*BLEU*	*ChrF*	*BLEU*	*ChrF*	*BLEU*	*ChrF*	*BLEU*	*ChrF*	*BLEU*	*ChrF*
Dense	1	2048	29.2	53.0	28.2	52.2	25.3	50.9	32.8	58.0	16.7	43.4	26.4	51.5
Switch	32	2048	29.9	53.9	29.2	53.1	26.7	52.0	34.9	59.7	17.4	44.8	27.6	52.7
Top-k	64	1024	30.7	54.6	29.2	53.1	27.4	53.5	35.9	**60.9**	17.6	44.9	28.2	53.4
	128	512	30.9	54.9	29.3	53.1	27.8	53.5	**36.3**	**60.9**	17.7	45.1	28.4	53.5
XMoE	64	1024	31.7	**55.7**	**29.4**	**53.3**	28.4	**54.0**	36.1	60.5	**18.1**	**45.6**	**28.7**	**53.8**
	128	512	**32.0**	**55.7**	29.2	53.2	**28.5**	53.9	35.5	60.1	18.0	45.4	**28.7**	53.7

Table 3: Machine translation on WMT23 datasets. *ChrF* is the abbreviation of ChrF2^{++}

.

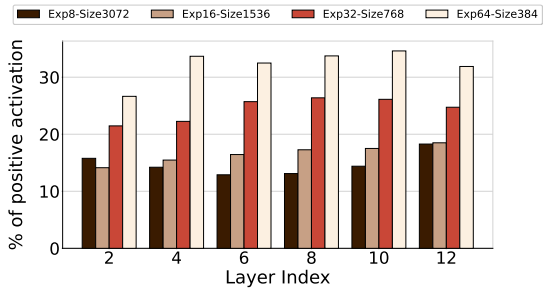


Figure 6: Average percentage of positive values in the FFN layers after the activation function.

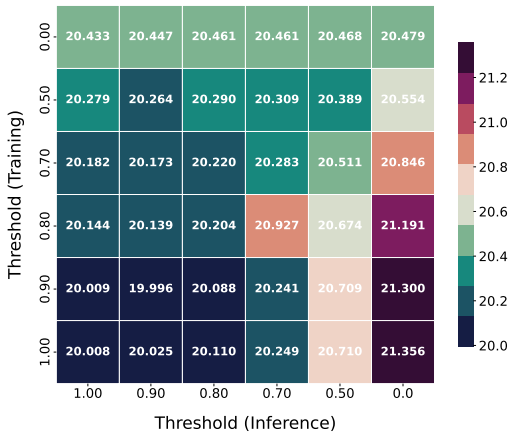


Figure 7: Effect of threshold on perplexity$^{\downarrow}$ during the training and inference stages. The models are trained on the WikiText-103 dataset utilizing 32 experts, each with a size of 768.

of 1.0. It is noteworthy that at a threshold of 1.0, each token is sent to all experts by the threshold-based router. In contrast, at a threshold of 0.0, the router behaves like a top-1 router. Hence, replacing the top-1 router with the top-N router, where N represents the number of experts, is also a simple approach to enhance performance.

Wall Time. Figure 8 illustrates the per-batch wall time during inference. We observe that a reduction in FLOPs at the Mixture of Experts (MoE) layers correlates with a decrease in overall wall

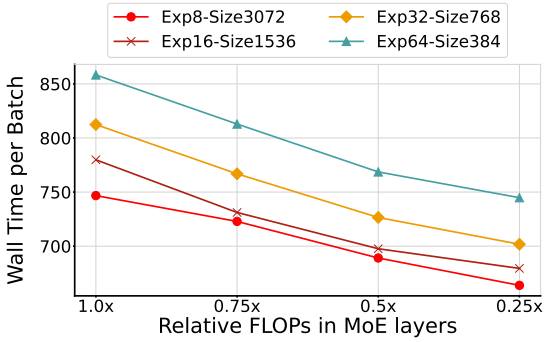


Figure 8: Per-batch wall time vs. FLOPs at MoE layers.

time. However, XMoE with smaller experts exhibit significantly higher latency compared to those with larger experts. The reason is that smaller experts require more sparse computation, which is not well supported on computation hardware such as TPUs and GPUs (Li et al., 2023b). In addition, the increased number of experts introduces additional computational overhead to the routing process due to operations like sorting and ranking across the expert pool. These observations highlight the limitations associated with further reducing expert dimensions. The size of experts should be properly chosen in order to fully utilize the advantages offered by XMoE.

6 Conclusion

This paper proposes a novel MoE design, XMoE, with the primary objective of improving the efficacy and efficiency of sparse MoE models. By employing small experts and a threshold-based router, XMoE demonstrates performance enhancements while significantly reducing FLOPs, leveraging the inherent sparsity of the model. Our research sheds light on the utilization of sparsity to improve model quality. As for future directions, we aim to further harness the advantages of sparse computation, focusing on enhancements from both hardware and algorithmic perspectives.

7 Limitations

Our experiments were conducted on language modeling and machine translation tasks. To ascertain the effectiveness of XMoE across a broader spectrum of NLP tasks, additional experiments are necessary. Additionally, due to the limited computational resources in our experiments, the largest model explored in this paper comprises 556 million parameters, notably smaller than the parameter counts in prevalent large-scale language models, which often exceed billions. To substantiate the claims made in this paper, further investigations in larger-scale configurations are needed. The expert size is also an important factor to XMoE, and setting it to 1 could yield valuable insights. Regrettably, our current implementation makes this model unfeasible. We will leave these further investigation as our future work.

References

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, T. J. Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeff Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. *ArXiv*.

Tianlong Chen, Zhenyu Zhang, Ajay Kumar Jaiswal, Shiwei Liu, and Zhangyang Wang. 2023. Sparse moe as the new dropout: Scaling dense and self-slimmable transformers. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*.

Hyung Won Chung, Le Hou, S. Longpre, Barret Zoph, Yi Tay, William Fedus, Eric Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, Albert Webson, Shixiang Shane Gu, Zhuyun Dai, Mirac Suzgun, Xinyun Chen, Aakanksha Chowdhery, Dasha Valter, Sharan Narang, Gaurav Mishra, Adams Wei Yu, Vincent Zhao, Yanping Huang, Andrew M. Dai, Hongkun Yu, Slav Petrov, Ed Huai hsin Chi, Jeff Dean, Jacob Devlin, Adam Roberts, Denny Zhou, Quoc V. Le, and Jason Wei. 2022. Scaling instruction-finetuned language models. *ArXiv*.

Damai Dai, Li Dong, Yaru Hao, Zhifang Sui, Baobao Chang, and Furu Wei. 2022. Knowledge neurons in pretrained transformers. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8493–8502, Dublin, Ireland. Association for Computational Linguistics.

William Fedus, Barret Zoph, and Noam Shazeer. 2022. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *J. Mach. Learn. Res.*, 23:120:1–120:39.

Mor Geva, Roei Schuster, Jonathan Berant, and Omer Levy. 2021. Transformer feed-forward layers are key-value memories. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 5484–5495, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Aaron Gokaslan and Vanya Cohen. 2019. Openwebtext corpus.

Sebastian Jaszczur, Aakanksha Chowdhery, Afroz Mohiuddin, Lukasz Kaiser, Wojciech Gajewski, Henryk Michalewski, and Jonni Kanerva. 2021. Sparse is enough in scaling transformers. *Advances in Neural Information Processing Systems*, 34:9895–9907.

Guillaume Lample, Alexandre Sablayrolles, Marc'Aurelio Ranzato, Ludovic Denoyer, and Hervé Jégou. 2019. Large memory layers with product keys. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 8546–8557.

Dmitry Lepikhin, HyoukJoong Lee, Yuanzhong Xu, Dehao Chen, Orhan Firat, Yanping Huang, Maxim Krikun, Noam Shazeer, and Zhifeng Chen. 2021. Gshard: Scaling giant models with conditional computation and automatic sharding. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net.

Mike Lewis, Shruti Bhosale, Tim Dettmers, Naman Goyal, and Luke Zettlemoyer. 2021. BASE layers: Simplifying training of large, sparse models. In *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pages 6265–6274. PMLR.

Jiamin Li, Qiang Su, Yitao Yang, Yimin Jiang, Cong Wang, and Hong Xu. 2023a. Adaptive gating in mixture-of-experts based language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 3577–3587, Singapore. Association for Computational Linguistics.

Zonglin Li, Chong You, Srinadh Bhojanapalli, Daliang Li, Ankit Singh Rawat, Sashank J. Reddi, Ke Ye, Felix Chern, Felix X. Yu, Ruiqi Guo, and Sanjiv Kumar. 2023b. The lazy neuron phenomenon: On emergence of activation sparsity in transformers. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2017. Pointer sentinel mixture models. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. 2019. fairseq: A fast, extensible toolkit for sequence modeling. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, pages 48–53, Minneapolis, Minnesota. Association for Computational Linguistics.

Joan Puigcerver, Carlos Riquelme Ruiz, Basil Mustafa, and Neil Houlsby. 2024. From sparse to soft mixtures of experts. In *The Twelfth International Conference on Learning Representations*.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21:140:1–140:67.

Samyam Rajbhandari, Conglong Li, Zhewei Yao, Minjia Zhang, Reza Yazdani Aminabadi, Ammar Ahmad Awan, Jeff Rasley, and Yuxiong He. 2022. Deepspeed-moe: Advancing mixture-of-experts inference and training to power next-generation AI scale. In *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*, volume 162, pages 18332–18346.

Stephen Roller, Sainbayar Sukhbaatar, Arthur Szlam, and Jason Weston. 2021. Hash layers for large sparse models. In *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pages 17555–17566.

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc V. Le, Geoffrey E. Hinton, and Jeff Dean. 2017. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*.

Mohammad Shoeybi, Mostofa Patwary, Raul Puri, Patrick LeGresley, Jared Casper, and Bryan Catanzaro. 2019. Megatron-lm: Training multi-billion parameter language models using model parallelism. *CoRR*, abs/1909.08053.

Hugo Touvron, Louis Martin, Kevin R. Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Daniel M. Bikel, Lukas Blecher, Cristian Cantón Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony S. Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel M. Kloumann, A. V. Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, R. Subramanian, Xia Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zhengxu Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. 2023. Llama 2: Open foundation and fine-tuned chat models. *ArXiv*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5998–6008.

An Yang, Junyang Lin, Rui Men, Chang Zhou, Le Jiang, Xianyan Jia, Ang Wang, Jie Zhang, Jiamang Wang, Yong Li, Di Zhang, Wei Lin, Lin Qu, Jingren Zhou, and Hongxia Yang. 2021. Exploring sparse expert models and beyond. *CoRR*, abs/2105.15082.

Zhengyan Zhang, Yankai Lin, Zhiyuan Liu, Peng Li, Maosong Sun, and Jie Zhou. 2022. MoEfication: Transformer feed-forward layers are mixtures of experts. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 877–890, Dublin, Ireland. Association for Computational Linguistics.

Yanqi Zhou, Tao Lei, Hanxiao Liu, Nan Du, Yanping Huang, Vincent Zhao, Andrew M. Dai, Zhifeng Chen, Quoc V. Le, and James Laudon. 2022. Mixture-of-experts with expert choice routing. In *NeurIPS*.

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 Appendix

A.1 Expert Selection

The process for expert selection concerning a given token is presented in Algorithm 1. Tokens are dispatched to experts according to I. Subsequent to the routing phase, each expert independently handles the tokens assigned to them. Given that the capacity of each expert is restricted to C, only the top C tokens, as per the priority R, are processed

Code	Language	#Bitext	Test
Uk	Ukrainian	10M	flores200
De	German	30M	WMT22
Ru	Russian	10M	WMT22
He	Hebrew	10M	flores200
Zh	Chinese	30M	WMT22

Table 4: Statistics of the training resources X→En from WMT23.

by each expert. Any tokens beyond capacity are disregarded.

Algorithm 1 Expert Selection Procedure

Require: Probability distribution $p = [p_1, p_2, ..., p_n]$, Threshold t

Ensure: Indices of selected experts I, corresponding priorities R

```
Sort p in descending order
Initialize I = [] and sum = 0
for i = 1 to n do
    sum = sum + p_i
    Append i to I
    Append p_i − i to R
    if sum ≥ t then
        Break
    end if
end for
return I, R
```

A.2 Load Balancing Loss

Let N represent the total number of experts involved in the evaluation process. The auxiliary loss function is formulated as follows:

$$\text{loss} = N \cdot \sum_{i=1}^{N} f_i \cdot p_i, \tag{7}$$

where f_i denotes the fraction of tokens that rank the i-th expert as their top choice, and p_i represents the sum of probabilities assigned to the top-ranked selection by these tokens.