TC-MoE: Augmenting Mixture of Experts with Ternary Expert Choice

Anonymous authors
Paper under double-blind review

Abstract

The Mixture of Experts (MoE) architecture has emerged as a promising solution for reducing computational overhead by selectively activating subsets of model parameters. The effectiveness of MoE models is primarily dependent on their routing mechanisms, with the widely adopted Top-K routing scheme used to activate experts. However, the Top-K scheme has notable limitations, including unnecessary activations and underutilization of existing experts. In this work, rather than modifying the routing mechanism as in previous studies, we propose Ternary Choice MoE (TC-MoE), a novel approach that expands the expert space by multiplying each expert with the ternary set $\{-1, 0, 1\}$. This expansion allows for more efficient and effective expert activations without incurring significant computational cost. Additionally, given the unique characteristics of the expanded expert space, we introduce a new load balancing loss and reward loss to ensure workload balance and achieve a flexible trade-off between effectiveness and efficiency. Extensive experiments demonstrate that TC-MoE achieves an average improvement of more than 1.1% over the traditional approaches, while reducing the average number of activated experts by up to 9%. These results confirm that TC-MoE effectively address the inefficiencies of classical routing schemes, offering a more efficient and scalable solution for MoE-based large language models.

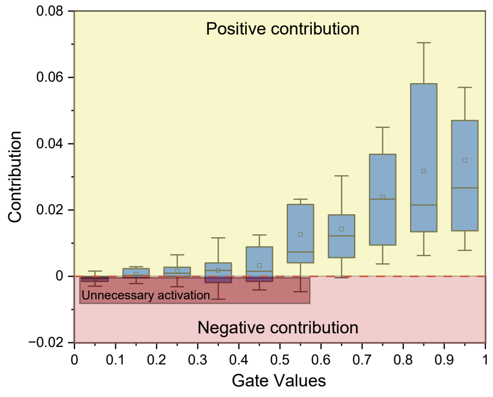
1 Introduction

In recent years, Large Language Models (LLMs) (Brown et al., 2020; Touvron et al., 2023; Achiam et al., 2023) have demonstrated impressive performance across a wide range of domains. However, modern LLMs still face inefficiencies, as they typically utilize all their parameters for every input token during both training and inference. This leads to a substantial increase in computing resource requirements as the models scale. To address these challenges, researchers have introduced the Mixture of Experts (MoE) architecture (Shazeer et al., 2017). The MoE architecture facilitates parameter scaling while maintaining reasonable computational expenses. Unlike traditional dense models, MoE models incorporate a routing mechanism that selectively activates specific subsets of parameters for particular input tokens. Recent advancements in MoE models (Jiang et al., 2024; Dai et al., 2024; Wu et al., 2024) have paved the way for scaling language models to unprecedented sizes and achieving remarkable performance enhancements.

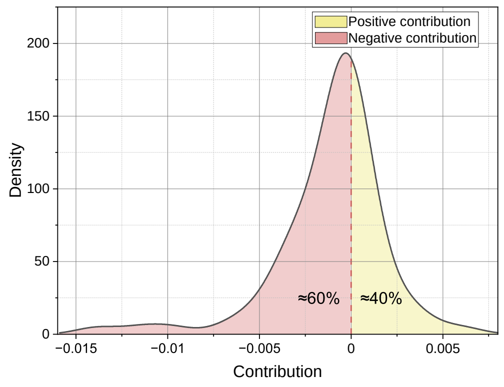
Within the MoE architecture, the routing mechanism plays a critical role as it significantly influences both the efficiency and effectiveness of model training. Traditional MoE frameworks, such as GShard (Lepikhin et al., 2021), Switch Transformers (Fedus et al., 2022), and ST-MoE (Zoph et al., 2022) employ the Top-K routing scheme. This method calculates the routing probability for each expert with respect to every input token. The Top-K experts, selected based on the highest routing probabilities, are then activated for each input token. The output is the weighted sum of the outputs from the activated experts.

However, recent work (Zhou et al., 2022; Huang et al., 2024), along with our experiments, demonstrates that the Top-K routing scheme is suboptimal. We identify the following limitations:

- **Unnecessary Activations**: The Top-K scheme activates a fixed number of experts for each token, neglecting the possibility of adaptively choosing the number of activated experts. As



(a) Distribution of contributions from activated experts. The experts are categorized based on their gate values. This shows that some activations contribute negatively to the performance, indicating unnecessary activations.



(b) Distribution of contributions from experts with low gate values after flipping the sign of their outputs. The results demonstrate that some experts can positively impact performance when their output signs are flipped.

Figure 1: Analysis of the limitations in the conventional Top-K routing scheme in a model with 2 activated experts out of 8. We compute the contribution of each activated expert by measuring the difference in the model output quality when the activation is masked. More details of this figure are provided in Appendix A.1.

shown in Figure 1a, not all activated experts in the Top-K scheme contribute positively to the model's performance, indicating the presence of unnecessary activations.

- **Underutilization of Existing Experts**: The Top-K scheme restricts experts to having only non-negative weights, disregarding the potential benefits of employing negative weights. As demonstrated in Figure 1b, approximately 40% of the experts with low gate values have positive contributions when their output signs are flipped.

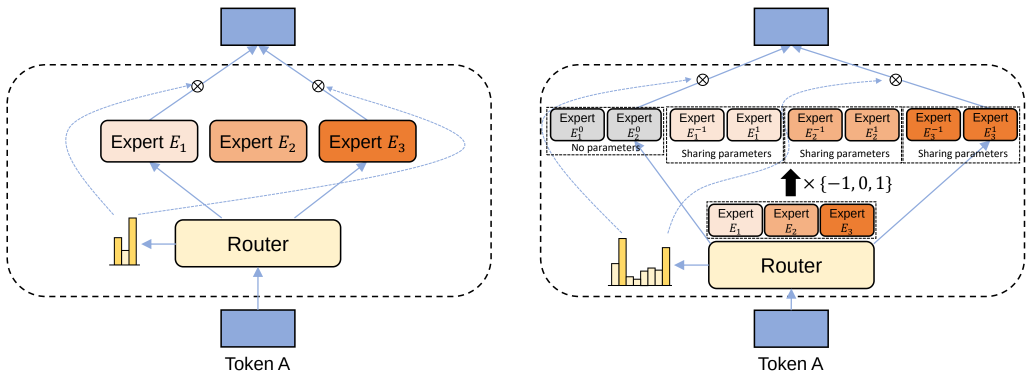
In this work, we introduce **Ternary Choice Mixture of Experts (TC-MoE)** to address these limitations. Unlike previous studies (Zhou et al., 2022; Yang et al., 2024; Huang et al., 2024) that focus on modifying the routing scheme, we explore an alternative direction by expanding the expert space. Inspired by the concept of ternary quantization, where weights are projected to $\{-1, 0, 1\}$, we propose creating an expanded expert space by multiplying each expert in the original space with $\{-1, 0, 1\}$. As demonstrated in Figure 2b, by applying the ternary choice to each expert, we obtain an expanded expert space without increasing the parameter count of the experts. This expanded expert space enables the router to learn more complex routing strategies during the training process, thereby addressing the aforementioned limitations without modifying the routing scheme.

However, the expanded expert space exhibits unique characteristics that differ from the original expert space. As illustrated in Figure 2b, certain pairs of experts share the same parameters, while some experts have no parameter and incur no computational cost. Due to these differences, the traditional load balancing loss becomes unsuitable. To address this, we propose a new load balancing loss to ensure an equitable distribution of workload. Furthermore, based on our analysis, we introduce a novel reward loss to facilitate a flexible trade-off between efficiency and effectiveness.

We thoroughly evaluate our method using multiple common benchmarks. The results demonstrate that our TC-MoE outperforms all competitors. Compared to the baseline model, our method achieves an average improvement of 1.1%, while reducing the average number of activated experts by up to 9%. When compared with other dynamic routing methods, our approach consistently achieves superior performance under different activation budgets. These results strongly confirm that TC-MoE effectively addresses the limitations of the classical routing scheme.

Our contributions can be summarized as follows:

1. We propose TC-MoE, a novel method to address the limitations of the classical routing scheme in MoE models. Unlike previous studies that focus on enhancing the routing



(a) Conventional MoE. (b) Our proposed TC-MoE.

Figure 2: Comparison of conventional MoE and our TC-MoE. In this example, the original expert space comprises 3 experts, with a router activating 2 of them. By multiplying the original expert space with $\{-1, 0, 1\}$, TC-MoE obtains an expanded expert space with a total of 8 experts. Two experts have no parameter and incur no computational cost. The remaining experts are divided into three groups, with each group of experts sharing parameters.

scheme, TC-MoE leverages the concept of ternary quantization to expand the expert space without incurring significant costs. This expansion promotes more effective activations by the router.

2. Given the unique load balance requirements in our expanded expert space, we redesign the load balancing loss. Additionally, we introduce a novel reward loss to achieve a flexible trade-off between effectiveness and efficiency.

3. Our experimental results demonstrate that TC-MoE achieves superior performance with fewer activated parameters, confirming significant improvements in both effectiveness and efficiency over the baseline.

2 Related Work

Mixture of Experts Models. MoE models (Jacobs et al., 1991; Jordan & Jacobs, 1994) have been extensively studied in artificial intelligence. The concept of using a trainable gating network to determine a sparse combination of experts is pioneered by the Sparsely-Gated MoE (Shazeer et al., 2017). Since then, numerous studies (Lepikhin et al., 2021; Fedus et al., 2022; Zoph et al., 2022; Jiang et al., 2024; Dai et al., 2024; Wei et al., 2024) have built upon this framework, demonstrating compelling empirical results by scaling MoE models to unprecedented sizes.

Routing Schemes. The MoE architecture relies on a routing module to determine the activation of experts, making the routing scheme a critical factor for MoE model performance. Early works (Shazeer et al., 2017; Fedus et al., 2022) employ the Top-K routing scheme, which calculates the routing probabilities for each experts and activates the Top-K experts with the highest probabilities. Recent studies have focused on improving routing schemes. Zhou et al. (2022) introduce the expert choice routing mechanism, which assigns equal capacity to every expert and allows tokens to compete for expert selection. Yang et al. (2024) propose a threshold-based router that uses a manually set threshold to control the number of activated experts for each token. Huang et al. (2024) also propose a threshold-based router but take it a step further by incorporating a dynamic loss to prevent from activating too many experts.

Heterogeneous Experts Design. Unlike classical MoE frameworks that utilize feed-forward networks with the same configuration for all experts, recent works have explored the design of heterogeneous experts. Ainslie et al. (2023b) propose a heavy branch alongside a light branch, using a router to select important tokens for processing through the heavy branch. Raposo et al. (2024) further refine this concept in the decoder-only setting by defining the light branch as a skip connection.

Additionally, Zeng et al. (2024) introduce a set of null experts alongside ordinary experts, while Wang et al. (2024) explore more diverse strategies for integrating heterogeneous experts.

In this paper, we propose TC-MoE, a novel method that complements existing research on routing mechanisms in MoE models. Our framework provides a comprehensive design for expert spaces by leveraging heterogeneous experts, thereby improving the overall performance and scalability of MoE architectures.

3 Method

In this section, we begin with an overview of the widely-used Top-K routing mechanism in MoE models. We then introduce our Ternary Choice MoE, a method that expands the expert space with minimal computational overhead. Following this, we propose a new load balance loss function to ensure effective load balance across the expanded expert space. Finally, we present a reward loss technique that achieves a flexible trade-off between efficiency and performance in our approach.

3.1 Top-K Routing Mechanism

In a typical MoE architecture for transformer language models, Feed-Forward Network (FFN) layers are replaced with MoE layers. Each MoE layers consists of N independent FFNs, referred to as experts, $\{E_1, E_2, \cdots, E_N\}$, along with a trainable router. Given a hidden representation $\mathbf{h} \in \mathbb{R}^d$ of the input token, the router computes the probability distribution over the experts as follows:

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

where $\mathbf{W}_g \in \mathbb{R}^{N\times d}$ is a trainable weight matrix, and $\mathbf{b}_g \in \mathbb{R}^N$ is the bias term. Then the Top-K router selects the top K experts with the highest probabilities for each input token. The gate values for the selected experts are set to the normalized probabilities, while those for the other experts are set to 0:

$$g_i(\mathbf{h}) = \begin{cases} p_i(\mathbf{h}) / \sum_{j\in\mathcal{E}} p_j(\mathbf{h}), & i \in \mathcal{E} \\ 0, & i \notin \mathcal{E} \end{cases} \tag{2}$$

where $\mathcal{E}$ denotes the set of the top K experts with the highest probabilities. The final output O of the MoE layer is computed as the weighted sum of the outputs from the activated experts:

$$\mathbf{O} = \sum_{i\in\mathcal{E}} g_i(\mathbf{h}) \cdot E_i(\mathbf{h}). \tag{3}$$

3.2 Ternary Choice MoE

Although the Top-K routing scheme is widely used in MoE models, we argue that this method still has notable limitations. As illustrated in Figure 1, the Top-K scheme suffers from unnecessary activations, where some activated experts negatively impact model performance. It also underutilizes existing experts, overlooking the potential benefits of contrasting expert outputs. While most previous studies focus on modifying the routing scheme to address these issues, we propose an alternative approach, Ternary Choice MoE, which expands the expert space and provides the router a more diverse set of activation options. Specifically, as illustrated in Figure 2b, we augment the original expert space by multiplying it with the ternary set $\{-1, 0, 1\}$. This allows us to project each expert E_i into three distinct experts $\{E_i^{-1}, E_i^0, E_i^1\}$, formulated as follows:

$$E_i^1(\mathbf{h}) \coloneqq E_i(\mathbf{h}), \quad E_i^0(\mathbf{h}) \coloneqq 0, \quad E_i^{-1}(\mathbf{h}) \coloneqq -E_i(\mathbf{h}), \quad \forall \mathbf{h} \in \mathbb{R}^d. \tag{4}$$

In this design, both E_i^1 and E_i^{-1} share the same parameters as E_i. While E_i^0 has no parameter and is the same across all experts. We further simplify by maintaining only $E_1^0, \cdots, E_K^0$, as this is sufficient for the Top-K router to activate any number from 0 to K of these experts. Therefore, TC-MoE has totally $2N + K$ experts.

Our method only requires maintaining the parameters of E_i, consistent with the original model. The only additional parameters and costs arise in the router. With the number of experts increased to

$2N + K$, we introduce $(N + K)d + N + K$ extra parameters and $O((N + K)d)$ additional computational costs in the router. However, this cost is negligible compared to the overall computational cost of the MoE block.

For simplicity, we define the sets of each type of expert as follows:

$$E^{-1} := \{E_i^{-1} | i \in [N]\}, \quad E^0 := \{E_i^0 | i \in [K]\}, \quad E^1 := \{E_i^1 | i \in [N]\} \tag{5}$$

Our method provides an alternative perspective for addressing the aforementioned limitations of the Top-K routing scheme. Without altering the classical Top-K routing scheme, incorporating E^0 allows the router to avoid unnecessary activations by activating experts from E^0, which contribute zero to the output and incur no computational cost. Additionally, the inclusion of E^{-1} enables the router to explore the potential benefits of flipping the signs of expert outputs.

Furthermore, we find that making a small improvement to the Top-K routing scheme by always activating experts from E^0 is beneficial. A detailed description of this improvement is provided in Appendix A.3.

3.3 Load Balancing Loss

In common MoE models (Fedus et al., 2022; Zoph et al., 2022), an auxiliary loss is typically introduced to encourage a balanced workload among experts. In our approach, however, experts are categorized into two types: $E^1 \cup E^{-1}$, which incur computational costs, and E^0, which does not incur any computational cost. Therefore, reasonable workload balance considerations in our scenario are as follows: (1) experts from E^0 do not need to be balanced with other experts since they do not contribute to computational costs, and (2) the sum of the workloads of expert E_i^1 and expert E_i^{-1} should be balanced, as E_i^1 and E_i^{-1} are distributed on the same device in scenarios involving expert parallelism (Lepikhin et al., 2021). Based on these considerations, we propose a new formulation for the load balancing loss:

$$f_i = \frac{1}{KT} \sum_{j=1}^{T} \mathbb{1}(\text{Token } j \text{ selects expert } E_i^1 \text{ or } E_i^{-1}), \tag{6}$$

$$\overline{f} = \frac{1}{N} \sum_{i=1}^{N} f_i, \tag{7}$$

$$p_i = \frac{1}{T} \sum_{j=1}^{T} \left[p_{E_i^1}(\mathbf{h}_j) + p_{E_i^{-1}}(\mathbf{h}_j) \right], \tag{8}$$

$$\mathcal{L}_{aux} = \sum_{i=1}^{N} (f_i - \overline{f}) \cdot p_i, \tag{9}$$

where T is the sequence length, f_i represents the sum of the activation frequencies of experts E_i^1 and E_i^{-1}, and p_i denotes the sum of the average probabilities assigned to experts E_i^1 and E_i^{-1}.

3.4 Flexible Trade-off between efficiency and effectiveness

Since E^0 represents a special class of experts that incurs no computational cost, it is crucial to understand how the router learns to allocate gate values to these experts. Based on our analysis, we propose a novel auxiliary loss, termed the reward loss, to achieve a flexible trade-off between efficiency and effectiveness by tuning the activated ratio of experts from E^0.

During the backward pass, the gradient of the gate value for each expert is computed as follows:

$$\frac{\partial \mathcal{L}}{\partial g_i(\mathbf{h})} = \begin{cases} \left\langle \frac{\partial \mathcal{L}}{\partial \mathbf{O}}, E_i(\mathbf{h}) \right\rangle, & i \in \mathcal{E} \\ 0. & i \notin \mathcal{E} \end{cases} \tag{10}$$

For each activated expert E_i, the term $-\frac{\partial \mathcal{L}}{\partial g_i(\mathbf{h})}$ indicates the impact of increasing the expert's gate value on reducing the loss function. Since the sum of the gate values is constrained to 1, a competitive dynamic arises among the activated experts. Experts that significantly contribute to reducing the loss function are assigned higher gate values, as verified in Figure 1a.

Table 1: Comparison of performance across evaluation benchmarks. "**Avg. K**" denotes the average number of activated experts that incurs computational costs during inference. "**#FLOPs ↓**" denotes the reduction ratio of FLOPs compared to the Top-K baseline. The **bold** number indicates the highest value for each benchmark.

Pre-trained Dataset	Method	Avg. K	#FLOPs ↓	ARC-Easy	BoolQ	MMLU	LAMBADA	HellaSwag	OpenBookQA	PIQA	SIQA	WinoGrande	Avg
RedPajama	*Base model*												
	Top-K	2.00	-	**57.03**	58.75	25.24	**50.40**	42.76	39.40	68.17	43.91	52.72	48.71
	Random drop	1.85	5.4%	56.48	58.62	25.35	50.13	42.83	39.00	**69.53**	44.68	51.30	48.66
	Top-P	1.99	0.3%	55.26	**59.54**	**25.74**	50.30	42.22	41.00	68.66	43.55	53.20	48.83
	TC-MoE	1.82	6.5%	**57.03**	59.20	25.58	50.16	**43.51**	**42.00**	68.66	**44.88**	**54.85**	**49.54**
	Fine-grained base model												
	Top-K	4.00	-	56.69	55.35	25.16	50.16	42.72	39.6	**68.93**	44.11	**52.49**	48.36
	TC-MoE	3.87	2.3%	**57.58**	**58.56**	**26.80**	**50.46**	**43.16**	**41.80**	68.28	**45.19**	52.09	**49.32**
FineWeb	*Tiny model*												
	Top-K	2.00	-	55.13	56.76	26.02	48.32	46.46	37.60	71.33	44.68	52.41	48.75
	TC-MoE	1.83	5.8%	**55.93**	**58.53**	**26.2**	**48.85**	**46.65**	**41.20**	**71.71**	**46.32**	**53.99**	**49.93**
	Base model												
	Top-K	2.00	-	60.19	50.76	26.46	53.95	53.23	43.00	**74.48**	45.60	55.33	51.44
	TC-MoE	1.86	5.1%	**60.56**	**57.4**	**26.67**	**54.01**	**54.05**	**44.00**	73.45	**47.24**	**56.12**	**52.61**

Following Equation 10, for activated expert E_i^0, we have

$$\frac{\partial \mathcal{L}}{\partial g_{E_i^0}(\mathbf{h})} = \left\langle \frac{\partial \mathcal{L}}{\partial \mathbf{O}}, E_i^0(\mathbf{h}) \right\rangle = \left\langle \frac{\partial \mathcal{L}}{\partial \mathbf{O}}, \mathbf{0} \right\rangle = 0. \tag{11}$$

This indicates that expert E_i^0 have no impact on reducing the loss function. Therefore, when competing with other activated experts, expert E_i^0 will tend to receive higher gate values than experts with negative impacts but lower than those with positive impacts. This effectively helps in avoiding unnecessary activations.

Based on the above analysis, we propose extending our method to achieve a flexible trade-off between efficiency and effectiveness. Specifically, we manually assign a negative value to $\frac{\partial \mathcal{L}}{\partial g_{E_i^0}(\mathbf{h})}$ instead of 0, thereby giving expert E_i^0 a positive contribution in reducing the loss function. Consequently, the router will learn to promote the activation of these experts, while selectively deactivating other types of experts with minimal positive contributions. To achieve this, we introduce a new auxiliary loss, termed the reward loss, defined as follows:

$$\mathcal{L}_{rwd} = -\frac{1}{T} \sum_{i=1}^{K} \sum_{j=1}^{T} g_{E_i^0}(\mathbf{h}_j), \tag{12}$$

where T is the sequence length, and $g_{E_i^0}(\mathbf{h}_j)$ represents the gate values of expert E_i^0 on token $\mathbf{h}_j$.

Linearly combining the language modeling loss ($\mathcal{L}_{lm}$), the load balance loss, and the reward loss, we yield the total loss, formulated as follows:

$$\mathcal{L} = \mathcal{L}_{lm} + \alpha_1 \mathcal{L}_{aux} + \alpha_2 \mathcal{L}_{rwd}, \tag{13}$$

where α_1 is a hyper-parameter known as the load balance factor, and α_2 is a hyper-parameter known as the reward factor.

4 Experiments

4.1 Experimental Settings

Pre-trained Datasets. We train our models using the RedPajama dataset (Computer, 2023) and the FineWeb dataset (Penedo et al., 2024). The RedPajama dataset includes diverse sources such as Common Crawl (CC), C4, Wikipedia, Github, books, arxiv, and Stackexchange. The FineWeb dataset is an open-source, high-quality training dataset consisting of cleaned and deduplicated english web data from CC. In our experiments, all models are trained on 100B tokens.

Architecture. We employ a decoder-only transformer model, primarily based on the LLaMA architecture (Touvron et al., 2023). Each transformer layer includes both an attention layer and an MoE layer. RMSNorm (Zhang & Sennrich, 2019) is applied to the inputs of both attention layers and MoE layers. Within the attention layer, we adopt the Group-Query Attention (GQA) (Ainslie et al., 2023a). Additionally, RMSNorm is used to normalize each key vector. Each FFN expert employs the SwiGLU activation function (Shazeer, 2020). In our experiments, We employ three types

Table 2: Configurations of our MoE models.

Model	#Layers	#Hidden Size	#Heads	#KV Heads	#Intermediate Size	#Activated Experts/ #Total Experts	#Activated Params/ #Total Params
Tiny	24	768	12	2	2048	2/8	298M/978M
Base	32	1024	16	2	2816	2/8	681M/2.3B
Fine-grained base	32	1024	16	2	1280	4/16	631M/2.1B

Table 3: Ablation study on the contribution of different types of experts. "Multiplication Set" denotes the set used to multiply the original expert space, "Average K" denotes the average number of activated experts. Specifically, $\{1\}$ represents the Top-K baseline.

Multiplication Set	Average K	#FLOPs ↓	Average Performance
$\{1\}$	2.00	-	48.71
$\{-1,1\}$	2.00	0.0%	49.00 (+0.29)
$\{0,1\}$	1.81	6.9%	49.23 (+0.52)
$\{-1,0,1\}$	1.82	6.4%	49.54 (+0.83)

of models: tiny, base, and fine-grained base. Table 2 summarizes their respective configurations. We use the same tokenizer as GPT-NeoX-20B (Black et al., 2022), which has a vocabulary size of 50257.

Competitors. We pre-train three baseline methods alongside our proposed TC-MoE using the aforementioned architecture:

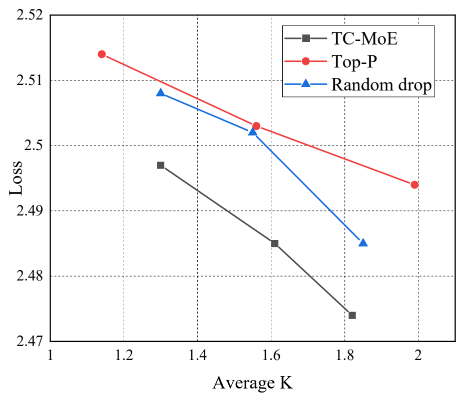
1. **Top-K**: A standard Top-K routing scheme that activates the top K experts for each token. We select $K = 2$ or $K = 4$ as these are the most common configurations in modern MoE architectures (Zoph et al., 2022; Jiang et al., 2024; Wei et al., 2024; Wu et al., 2024).
2. **Random drop**: A variant of the Top-K routing scheme that, with probability p, does not activate the expert with the second highest probability.
3. **Top-P**: The Top-P routing scheme (Huang et al., 2024), which activates the smallest set of experts whose cumulative probabilities surpass a threshold P for each token.
4. **TC-MoE**: Our proposed method, which expands the expert space and adopts the standard Top-K routing scheme to activate the top K experts within this expanded expert space for each token.

The Top-K baseline adopts a fixed number of activated experts, whereas Random drop, Top-P, and TC-MoE allow for a flexible trade-off between effectiveness and efficiency by tuning specific hyper-parameters. Details of these hyper-parameters are provided in Appendix A.2.

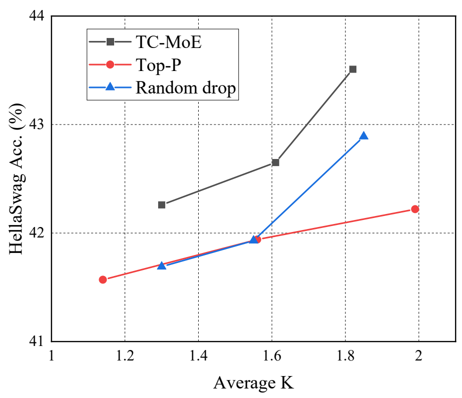
Evaluation. We evaluate these models on seven different benchmarks: ARC (Clark et al., 2018), BoolQ (Clark et al., 2019), MMLU (Hendrycks et al., 2021), LAMBADA (Paperno et al., 2016), HellaSwag (Zellers et al., 2019), OpenBookQA (Mihaylov et al., 2018), PIQA (Bisk et al., 2020), SIQA (Sap et al., 2019), and WinoGrande (Sakaguchi et al., 2021). These tasks examine models' logical reasoning, language understanding, commonsense reasoning, and knowledge utilization. Additionally, we measure the average number of activated experts that incur computational costs to demonstrate the efficiency of each model. Note that in our TC-MoE, only the activations of E^{-1} and E^1 are counted since E^0 incurs no computational cost. For simplicity, we refer to the average number of activated experts as the average number of activated experts that incur computational costs in the following sections.

4.2 Main Results

Table 1 summarizes the performance of various models across different evaluation benchmarks. The results highlight the superior performance of our proposed TC-MoE compared to competing methods.



(a)



(b)

Figure 3: Comparison of (a) the language modeling loss and (b) the HellaSwag accuracy under different budgets for the average number of activated experts. The results demonstrate TC-MoE outperforms other competitors under all settings.

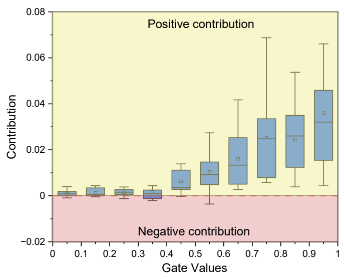
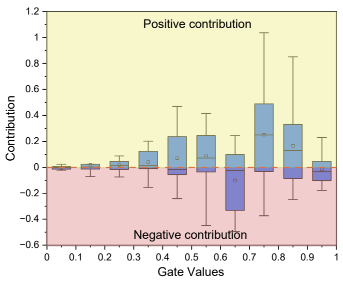
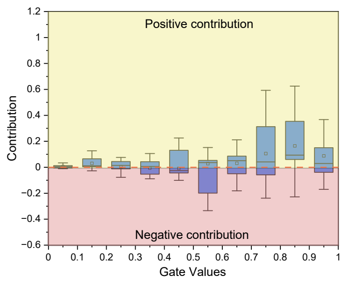


Figure 4: Distribution of contributions from activated experts in TC-MoE on pre-trained data.



(a)



(b)

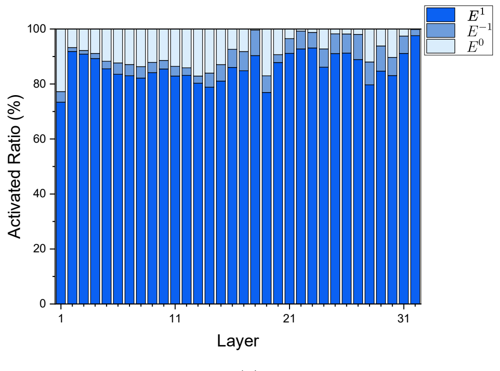
Figure 5: Distribution of contributions from activated experts in (a) baseline and (b) TC-MoE on ARC-Easy. The results show a significant alleviation of unnecessary activations by TC-MoE.

Specifically, when pre-trained the base model on the RedPajama dataset, TC-MoE outperforms competitors on ARC-Easy, HellaSwag, OpenBookQA, SIQA, and WinoGrande, while achieving comparable results on BoolQ, MMLU, LAMBADA and PIQA. Notably, TC-MoE achieves an average accuracy of 49.54%, surpassing the Top-K baseline by 0.83%, Random drop by 0.88% and Top-P by 0.71%. For the fine-grained base model pre-trained on the RedPajama dataset, TC-MoE also outperforms the Top-K baseline, improving the average accuracy by 0.96%.

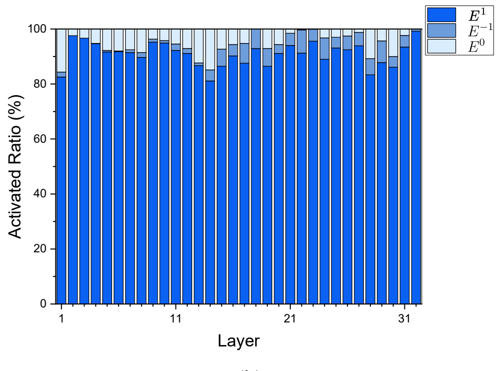
When pre-training on the FineWeb dataset, TC-MoE demonstrates even greater accuracy improvements. For the tiny model, TC-MoE surpasses the Top-K baseline by 1.18%. Similarly, for the base model, TC-MoE outperforms the Top-K baseline by 1.17%.

Beyond improved accuracy, TC-MoE consistently demonstrates greater efficiency. Specifically, it reduces the average number of activated experts by 9.0% and the required FLOPs by 6.5% compared to the Top-K baseline on the base model pre-trained on the RedPajama dataset. On the FineWeb dataset, TC-MoE reduces the average number of activated experts by 7.0% and the required FLOPs by 5.1%. These results demonstrate that our method achieves significant gains in both effectiveness and efficiency over the Top-K baseline.

Additionally, we conduct a thorough comparison of these methods under different budgets for the average number of activated experts. The results are shown in Figure 3. The figure demonstrates that the TC-MoE consistently outperforms the other two competitors across all settings regarding the average number of activated experts. Notably, in terms of the language modeling loss, TC-MoE reduces the loss by approximately 0.017 compared to competitors. For HellaSwag accuracy, TC-MoE improves accuracy by up to 0.7% compared to other methods.

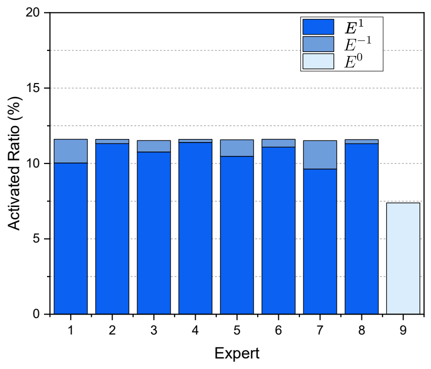


(a)

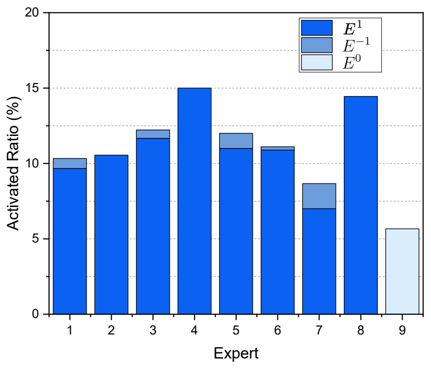


(b)

Figure 6: The activated ratio of different types of experts across layers on (a) the pre-trained data and (b) the test data (ARC-Easy). The results show a similar activated pattern on different data.



(a)



(b)

Figure 7: The activated ratio of different experts in layer 16 on (a) the pre-trained data and (b) the test data (ARC-Easy). The results show the effectiveness of our load balance loss during both training and inference.

4.3 Ablation Study

We conduct an ablation study by evaluating the performance of our method using only a subset of $\{-1, 0, 1\}$ to multiply the original expert space. As demonstrated in Table 3, expanding the expert space with either $\{-1, 1\}$ or $\{0, 1\}$ improves the model performance. Specifically, expanding the expert space with $\{-1, 1\}$ results in an average performance increase of 0.29% while the average number of activated experts remains at 2.0. When expanding the expert space with $\{0, 1\}$, the average performance increase by 0.52%, and the average activated experts reduced by 0.19. When expanding the expert space with the complete set $\{-1, 0, 1\}$, the model achieve the best performance, with both improved results and a reduced number of activated experts. In summary, both type E^{-1} and type E^0 contribute to the improvement of model performance, while type E^0 also significantly enhances model efficiency.

4.4 Analysis

Unnecessary Activations. We investigate the effect of our method on addressing unnecessary activations. Figure 4 shows the distribution of contributions from activated experts in TC-MoE on the pre-trained data. Compared to the distribution of contributions in the baseline model, shown in Figure 1a, our TC-MoE significantly alleviates the occurrence of unnecessary activations. Additionally, when analyzing unnecessary activations on ARC-Easy, we observe many more activations that contribute negatively. In this scenario, as shown in Figure 5, TC-MoE alleviates the occurrence of unnecessary activations even more significantly.

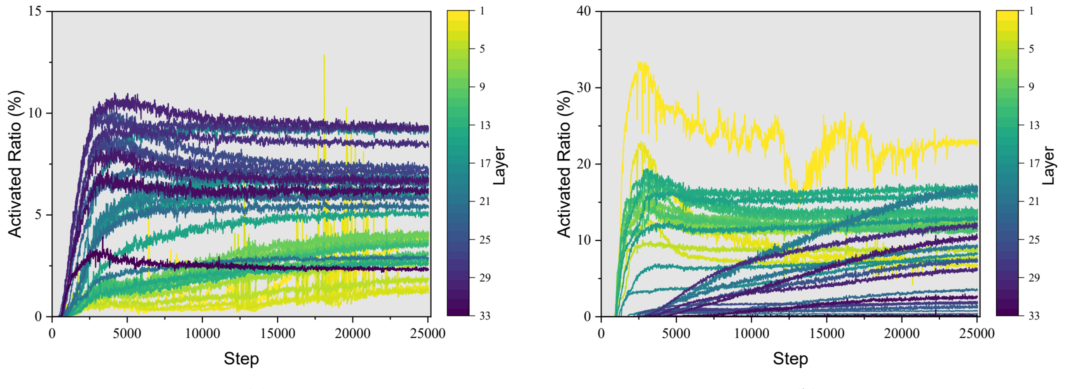


Figure 8: The changing curves of activated ratios of (a) type E^{-1} and (b) type E^0 of different layers.

Activated Ratio of Different Types of Experts. To analyze the activated ratio of different types of experts, we visualize the activated ratios across layers in TC-MoE. The results are shown in Figure 6. We observe that type E^1 has the highest activated ratio, indicating a major contribution to the output. Additionally, the model spontaneously learns to allocate around 20% activated ratio to type E^0 and type E^{-1}, highlighting the necessity of these experts for more powerful routing. The activation ratios on ARC-Easy are similar as that on pre-trained data, demonstrating the generalization of our method.

The distribution of activated ratios varies significantly across different layers. Figure 8 visualizes the changing curves of the activated ratios across layers during the entire training process. We find that the activation ratios of type E^{-1} and type E^0 across layers exhibit a converse pattern. For type E^{-1}, the model tends to activate more of them in the deeper layers. In contrast, type E^0 is primarily activated in the shallow layers.

Load Balance. To explore the effectiveness of our load balancing loss, we visualize the activated ratios of different experts in our model during the training process. Figure 7 shows the activated ratios in layer 16. To observe the actual load balancing distribution, we stack the activated ratios of each E_i^{-1} and E_i^1, as they are distributed on the same device when involving expert parallelism. Additionally, we plot the sum of the activated ratio of type E^0 at the position of expert 9, since they do not contribute to computational costs. We observe that our TC-MoE achieves a near-perfect workload balance with our designed load balancing loss on the pre-trained dataset. The sum of the workloads of experts E_i^1 and experts E_i^{-1} are balanced, each around 11.5%. Additionally, the activated ratios of expert E_i^1 and expert E_i^{-1} are not fixed but are instead learned dynamically by the model. Furthermore, experts from E^0 do not participate in the load balancing, allowing the model to activate E^0 without any constraints. On ARC-Easy, we observe a slight deviation in load balance, with the maximum expert workload reaching approximately 15.0%, while the minimum workload is around 8.5%.

5 Conclusions

In this paper, we present Ternary Choice MoE (TC-MoE), a novel approach designed to address the limitations of traditional Top-K routing in MoE architectures. By expanding the expert space through a simple yet effective method of multiplying each expert by the ternary set $\{-1, 0, 1\}$, we introduce greater flexibility and diversity into the expert activation process without incurring significant additional costs. Our approach enables more effective use of experts, mitigating issues such as unnecessary activations and underutilization of existing experts that are common in conventional MoE models. Extensive experiments across various benchmarks demonstrate consistent improvements in both effectiveness and efficiency, outperforming existing methods. These results highlight the potential of TC-MoE as a scalable, computationally efficient method for MoE models. We believe this work opens new perspectives for further development in the design and optimization of MoE models, paving the way for more advanced and resource-efficient large-scale models.

REFERENCES

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. GPT-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

Joshua Ainslie, James Lee-Thorp, Michiel de Jong, Yury Zemlyanskiy, Federico Lebrón, and Sumit Sanghai. GQA: Training generalized multi-query transformer models from multi-head checkpoints. *arXiv preprint arXiv:2305.13245*, 2023a.

Joshua Ainslie, Tao Lei, Michiel de Jong, Santiago Ontañón, Siddhartha Brahma, Yury Zemlyanskiy, David Uthus, Mandy Guo, James Lee-Thorp, Yi Tay, et al. COLT5: Faster long-range transformers with conditional computation. *arXiv preprint arXiv:2303.09752*, 2023b.

Yonatan Bisk, Rowan Zellers, Jianfeng Gao, Yejin Choi, et al. PIQA: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2020.

Sid Black, Stella Biderman, Eric Hallahan, Quentin Anthony, Leo Gao, Laurence Golding, Horace He, Connor Leahy, Kyle McDonell, Jason Phang, et al. GPT-NeoX-20B: An open-source autoregressive language model. *arXiv preprint arXiv:2204.06745*, 2022.

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. In *Advances in Neural Information Processing Systems*, 2020.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. BoolQ: Exploring the surprising difficulty of natural yes/no questions. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologiess, Volume 1 (Long and Short Papers)*, 2019.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*, 2018.

Together Computer. RedPajama: an open dataset for training large language models, 2023. URL `https://github.com/togethercomputer/RedPajama-Data`.

Damai Dai, Chengqi Deng, Chenggang Zhao, RX Xu, Huazuo Gao, Deli Chen, Jiashi Li, Wangding Zeng, Xingkai Yu, Y 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.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. In *International Conference on Learning Representations*, 2021.

Quzhe Huang, Zhenwei An, Nan Zhuang, Mingxu Tao, Chen Zhang, Yang Jin, Kun Xu, Kun Xu, Liwei Chen, Songfang Huang, and Yansong Feng. Harder task needs more experts: Dynamic routing in MoE models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2024.

Robert A Jacobs, Michael I Jordan, Steven J Nowlan, and Geoffrey E Hinton. Adaptive mixtures of local experts. *Neural Computation*, 3(1):79–87, 1991.

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.

Michael I Jordan and Robert A Jacobs. Hierarchical mixtures of experts and the em algorithm. *Neural Computation*, 6(2):181–214, 1994.

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. In *International Conference on Learning Representations*, 2021.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. Can a suit of armor conduct electricity? a new dataset for open book question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, 2018.

Denis Paperno, Germán Kruszewski, Angeliki Lazaridou, Ngoc Quan Pham, Raffaella Bernardi, Sandro Pezzelle, Marco Baroni, Gemma Boleda, and Raquel Fernández. The LAMBADA dataset: Word prediction requiring a broad discourse context. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2016.

Guilherme Penedo, Hynek Kydlíček, Loubna Ben allal, Anton Lozhkov, Margaret Mitchell, Colin Raffel, Leandro Von Werra, and Thomas Wolf. The FineWeb datasets: Decanting the web for the finest text data at scale, 2024. URL `https://arxiv.org/abs/2406.17557`.

David Raposo, Sam Ritter, Blake Richards, Timothy Lillicrap, Peter Conway Humphreys, and Adam Santoro. Mixture-of-Depths: Dynamically allocating compute in transformer-based language models. *arXiv preprint arXiv:2404.02258*, 2024.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. WinoGrande: An adversarial winograd schema challenge at scale. *Communications of the ACM*, 64(9):99–106, 2021.

Maarten Sap, Hannah Rashkin, Derek Chen, Ronan LeBras, and Yejin Choi. SocialIQa: Commonsense reasoning about social interactions. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*, 2019.

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

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. In *International Conference on Learning Representations*, 2017.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

An Wang, Xingwu Sun, Ruobing Xie, Shuaipeng Li, Jiaqi Zhu, Zhen Yang, Pinxue Zhao, JN Han, Zhanhui Kang, Di Wang, et al. Hmoe: Heterogeneous mixture of experts for language modeling. *arXiv preprint arXiv:2408.10681*, 2024.

Tianwen Wei, Bo Zhu, Liang Zhao, Cheng Cheng, Biye Li, Weiwei Lü, Peng Cheng, Jianhao Zhang, Xiaoyu Zhang, Liang Zeng, et al. Skywork-MoE: A deep dive into training techniques for mixture-of-experts language models. *arXiv preprint arXiv:2406.06563*, 2024.

Shaohua Wu, Jiangang Luo, Xi Chen, Lingjun Li, Xudong Zhao, Tong Yu, Chao Wang, Yue Wang, Fei Wang, Weixu Qiao, et al. Yuan 2.0-M32: Mixture of experts with attention router. *arXiv preprint arXiv:2405.17976*, 2024.

Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. Efficient streaming language models with attention sinks. In *International Conference on Learning Representations*, 2024.

Yuanhang Yang, Shiyi Qi, Wenchao Gu, Chaozheng Wang, Cuiyun Gao, and Zenglin Xu. XMoE: Sparse models with fine-grained and adaptive expert selection. In *Findings of the Association for Computational Linguistics: ACL 2024*, 2024.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. Hellaswag: Can a machine really finish your sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 2019.

Zihao Zeng, Yibo Miao, Hongcheng Gao, Hao Zhang, and Zhijie Deng. AdaMoE: Token-adaptive routing with null experts for mixture-of-experts language models. *arXiv preprint arXiv:2406.13233*, 2024.

Biao Zhang and Rico Sennrich. Root mean square layer normalization. In *Advances in Neural Information Processing Systems*, 2019.

Yanqi Zhou, Tao Lei, Hanxiao Liu, Nan Du, Yanping Huang, Vincent Zhao, Andrew M Dai, Quoc V Le, James Laudon, et al. Mixture-of-experts with expert choice routing. In *Advances in Neural Information Processing Systems*, 2022.

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 Calculating the contribution of each activation

To evaluate the effectiveness of the routing scheme, we conduct experiments to analyze the impact of routing decisions on model outputs. It is important to note that we are not discussing the contribution of each expert, but rather the effect of each individual activation decision made by the router.

Specifically, we first randomly select 15 sequences from the training set of RedPajama (Computer, 2023) and the test set of ARC-Easy (Clark et al., 2018), respectively. The average sequence length of samples in the training set is 1608, while it is 30 in the test set, We use the language modeling loss on these sequences to measure the quality of model outputs. For a specific activation A, a straightforward method to obtain its contribution is to forward the model on the input sequence twice: once with this activation and once masking this activation. We then calculate the difference in the loss function. This can be formulated as follows:

$$\text{Contribution}_A \coloneqq \mathcal{L}(M_{\setminus\{A\}}(x)) - \mathcal{L}(M(x)), \tag{14}$$

where x denotes the input sequence, $\mathcal{L}$ denotes the loss function, M denotes the function of the model, and $M_{\setminus\{A\}}$ denotes the function of the model when masking the activation A. For sequences sampled from the pre-trained data, we compute the loss across all positions, whereas for sequences sampled from the test data, we calculate the loss only over the tokens constructing the answer. When masking activation A results in a higher loss, we calculate a positive contribution for activation A. This indicates that the activation decision has a beneficial impact on model performance.

However, we observe empirically that the impact of masking a single activation is too small to analyze effectively. Therefore, we alternatively categorize the activations into groups based on their gate values and calculate the contribution of each group. Specifically, we divide the gate values from 0 to 1 into 10 intervals: 0 to 0.1, 0.1 to 0.2, ..., 0.9 to 1.0. Nevertheless, the unequal size of different groups still makes it unfair to compare the contributions across groups. To address this, we randomly select the same number of activations to mask within each group for a fair comparison. Specifically, we mask 20 activations in each layer of each group for sequences sampled from the training set, and 5 activations in each layer of each group for sequences sampled from the test set. We then calculate the loss difference as shown in Equation 14.

For the experiments involving the flipping of expert output signs, we use a similar method. For a specific activation A, we forward the model on the input sequence twice: once with this activation and once with the flipped sign activation, then calculate the difference in the loss function. We define this as:

$$\text{Contribution}_{-A} \coloneqq \mathcal{L}(M_{\setminus\{A\},\cup\{-A\}}(x)) - \mathcal{L}(M(x)), \tag{15}$$

where $M_{\setminus\{A\},\cup\{-A\}}$ denotes the function of the model when the sign of activation A is flipped. We randomly flip the sign of 20 activations with gate values lower than 0.2 in each layer, obtaining the distribution shown in Figure 1b.

A.2 Hyper-parameters

We use the AdamW optimizer with a first-moment decay of $\beta_1 = 0.9$ and a second-moment decay of $\beta_2 = 0.95$. The weight decay is set to 0.1 in our experiments. The learning rate is gradually increased from 0 to 3e-4 in the first 10% of training steps and then decays to 3e-5 using a cosine decay schedule for the remaining steps. We set the sequence length to 2048 and the global batch size to 2048.

To achieve a flexible trade-off between effectiveness and efficiency for Random Drop, Top-P, and TC-MoE, we tune specific hyperparameters:

- **Random drop**: We set the drop probability p to 15%, 45%, and 70% to achieve average activation numbers of 1.85, 1.55, and 1.30, respectively.
- **Top-P**: We set the threshold P to 0.4 as in the original paper (Huang et al., 2024), and the dynamic loss weight to 1e-5, 2e-5, and 5e-5 to achieve average activation numbers of 1.99, 1.56, and 1.14, respectively.
- **TC-MoE**: We set the load balance factor α_1 to 0.01, and the reward factor α_2 to 0, 1e-5, and 2e-5 to achieve average activation numbers of 1.82, 1.61, and 1.30, respectively.

For initialization, we adopt an initializer range of 0.006. The weight matrix W_g of the router is also initialized with a standard deviation of 0.006. The bias term $b_g \in \mathbb{R}^{2N+K}$ of the router is initialized with 0 for experts of type E^1, -1 for experts of type E^{-1}, and -10 for experts of type E^0. This initialization is designed to make the router concentrate on type E^1 at the beginning of the training process.

A.3 Improving the Top-K routing scheme

Intuitively, experts from E^0 share similarities with attention sinks (Xiao et al., 2024). The intuition behind attention sinks is that the attention scores calculated by the Softmax operation sum up to one for all tokens. As a result, the model naturally learns to assign useless attention scores to sink tokens. Similarly, in MoE models, the Softmax operation is used by the router to calculate gate values. Therefore, we believe that experts from E^0 serve as sink experts in this case to collect unneeded gate values.

However, in the classical Top-K routing mechanism, as demonstrated in Equation 10, only the activated experts participate in the competition for the gate values. This means that experts from E^0 are effective only when they are among the Top-K experts. To address this, we improve our design by always activating experts of type E^0, allowing them to participate fully in the competition for gate values and better serve as sinks. Specifically, we modify the activation set $\mathcal{E}$ by taking the union with E^0. The updated calculation of gate values is formulated as follows:

$$g_i(\mathbf{h}) = \begin{cases} p_i(\mathbf{h}) / \sum_{j \in \mathcal{E} \cup E^0} p_j(\mathbf{h}), & i \in \mathcal{E} \cup E^0 \\ 0, & i \notin \mathcal{E} \cup E^0 \end{cases} \tag{16}$$

where $\mathcal{E}$ denotes the set of the top K experts with the highest probabilities.

A.4 Effect of the reward loss

We also investigate the effect of our designed reward loss. As illustrated in Figure 9, we vary the reward factor from 0 to 2e-5, The average number of activated experts shows different changing curves. By increasing the reward factor, we encourage the model to select experts from E^0, which incur no computational cost. Consequently, the model tends to have a lower average number of activated experts. Specifically, the average number of activated experts converges to 1.82 when the reward factor is 0 and to 1.30 when the reward factor is 2e-5. These results demonstrate that tuning the reward factor enables a flexible trade-off between effectiveness and efficiency.

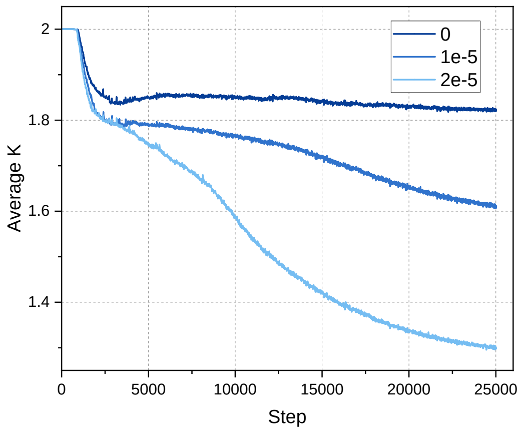


Figure 9: The changing curves of the average number of activated experts when varying the reward factor.

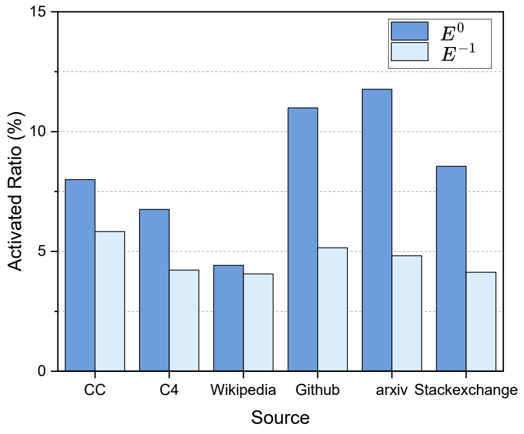


Figure 10: The activated ratios across different sources of training data.

A.5 Activated Ratio on Different Sources

We also investigate the activated ratio on different sources of training data. The results are shown in Figure 10. We observe that the activated ratio of E^0 exhibits significant variance across different sources. Notably, The activated ratio of E^0 is around 11% on data from Github and arxiv, while it is only 4% on data from Wikipedia. Similarly, the activated ratio of E^{-1} also varies across different sources. The activated ratio of E^{-1} is only 4% on data from Wikipedia and Stackexchange, while it is 6% on data from CC. The variance across different sources indicates some specialization of experts from E^0 and E^{-1}.