$\mathcal{A}da$MoE: Token-Adaptive Routing with Null Experts for Mixture-of-Experts Language Models

Anonymous ACL submission

Abstract

Mixture of experts (MoE) has become the standard for constructing production-level large language models (LLMs) due to its promise to boost model capacity without causing significant overheads. Nevertheless, existing MoE methods usually enforce a constant top-k routing for all tokens, which is arguably restrictive because various tokens (e.g., "<EOS>" vs. "apple") may require various numbers of experts for feature abstraction. Lifting such a constraint can help make the most of limited resources and unleash the potential of the model for downstream tasks. In this sense, we introduce $\mathcal{A}da$MoE to realize token-adaptive routing for MoE, where different tokens are permitted to select a various number of experts. $\mathcal{A}da$MoE makes minimal modifications to the vanilla MoE with top-k routing—it simply introduces a fixed number of *null experts*, which do not consume any FLOPs, to the expert set and increases the value of k. $\mathcal{A}da$MoE does not force each token to occupy a fixed number of null experts but ensures the average usage of the null experts with a load-balancing loss, leading to an adaptive number of null/true experts used by each token. $\mathcal{A}da$MoE exhibits a strong resemblance to MoEs with expert choice routing while allowing for trivial auto-regressive modeling. $\mathcal{A}da$MoE is easy to implement and can be effectively applied to pre-trained (MoE-)LLMs. Extensive studies show that $\mathcal{A}da$MoE can reduce average expert load (FLOPs) while achieving superior performance. For example, on the ARC-C dataset, applying our method to fine-tuning Mixtral-8x7B can reduce FLOPs by 14.5% while increasing accuracy by 1.69%.

1 Introduction

Large language models (LLMs) have exhibited exceptional performance across diverse tasks and domains (Touvron et al., 2023; Chiang et al., 2023; Chowdhery et al., 2023; Zhang et al., 2022). Nevertheless, LLMs' efficacy is heavily impacted by the substantial number of parameters they possess, with some high-performing LLMs containing up to 540B parameters (Chowdhery et al., 2023). The mixture of experts (MoE) mechanism (Shazeer et al., 2017) offers a compelling way to enhance model capability without a corresponding increase in computational overhead. Recent research further underscores the merits of MoE, vividly demonstrating its potential to support production-level applications (Jiang et al., 2024; Qwen, 2024).

MoE operates on the core assumption that a (small) subset of experts is sufficient to handle a single token effectively. MoE-LLMs, with Mixtral (Jiang et al., 2024) and DeepSeekMoE (Dai et al., 2024) as popular examples, often replace the feed-forward network (FFN) in the model with a set of FFN experts. A token-level router is introduced to sparsely activate the experts for various tokens, so the computational cost is constrained to a low level. We can also build experts with parameter-efficient fine-tuning (PEFT) modules (Hu et al., 2021; Liu et al., 2022) like LoRA, giving rise to Mo-LoRA approaches (Zadouri et al., 2023).

MoE routinely routes each token to a fixed amount of experts, typically the k ones with top routing probabilities. However, not all tokens require the same number of experts for feature abstraction. Intuitively, the semantic tokens deserve a higher concentration of experts, while others with less significant meaning can be processed more swiftly. Lifting the top-k routing constraint can help make the most of limited resources and unleash the potential of the model. To achieve this, MoE with expert choice routing (Zhou et al., 2022) performs expert-level routing, where each expert chooses a fixed number of tokens for processing and different tokens could be processed by different numbers of experts. Yet, an unacceptable drawback is that it is not suited to casual language modeling due to the reliance on future tokens for the top-k token selection (Zhou et al., 2022).

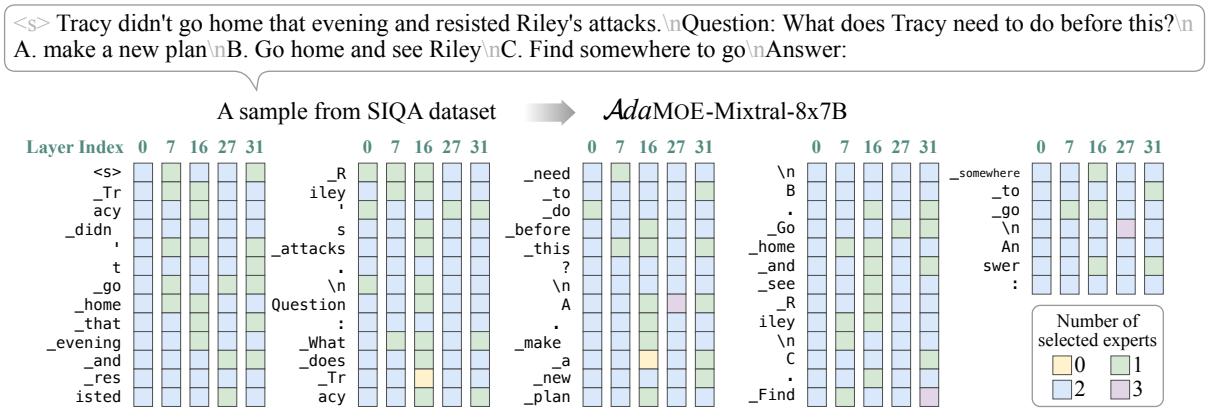

Figure 1: The number of selected experts for various tokens in an *Ada*MoE variant of Mixtral-8x7b. As shown, after applying *Ada*MoE, the model possesses the ability to perform token-adaptive routing. Also note that some tokens only require 1 expert for feature abstraction, which offers the opportunity for inference acceleration.

This work introduces *Ada*MoE, a novel method designed to achieve token-level adaptive routing in MoE, allowing different tokens to select varying numbers of experts. An illustrative example is presented in Figure 1. *Ada*MoE requires minimal changes to the vanilla MoE with top-k routing by incorporating a fixed number of *null experts* into the expert set. These null experts do not consume any computational resources. By increasing the value of k, more experts can be activated. To encourage the average usage rate of the null experts, *Ada*MoE minimizes a load balancing loss. This leads to an adaptive number of null experts and true experts being employed by each token. Notably, *Ada*MoE shares similarities with existing MoE with expert choice routing while also enabling straightforward causal language modeling.

*Ada*MoE is easy to implement and can be applied to both pre-trained regular LLMs and MoE-LLMs for supervised fine-tuning. For the former, we experiment on Llama2-7B (Touvron et al., 2023) by introducing LoRA experts and corresponding routers to the model. For the latter, we experiment on Mixtral-8x7B (Jiang et al., 2024) by augmenting the original router with extra weights for the null experts. The results underscore the effectiveness of *Ada*MoE's token-adaptive mechanism in enhancing both computational efficiency and model performance. For example, when fine-tuning Llama2-7B, *Ada*MoE achieves much higher accuracy across almost all evaluated datasets. Moreover, when fine-tuning Mixtral-8x7B with *Ada*MoE on ARC-Challenge (Clark et al., 2018), we observed a 14.5% reduction in total FLOPs, accompanied by a 1.69% increase in accuracy.

2 Related Works

2.1 Mixture of Experts

Mixture of Experts (MoE) (Jacobs et al., 1991; Shazeer et al., 2017) is an efficient scaling technique that allows for larger model sizes with less computation, resulting in enhanced performance. MoE models can be trained and used for inference more efficiently compared to dense models, requiring substantially fewer computational resources. Due to these advantages, pioneering works (Jiang et al., 2024; Dai et al., 2024) have applied MoE to transformer-based language models and demonstrated their superiority. Typically, they replace the feed-forward network (FFN) in each layer of the model with a routing function and multiple FFNs, referred to as experts, with only a subset of these experts being activated at any time. We refer to these models, which combine MoE and large language models (LLMs), as MoE-LLMs.

In addition to MoE-LLMs, fine-tuning techniques have also seen significant advancements. Pre-trained LLMs are often fine-tuned for downstream tasks. However, as models increase in size, full fine-tuning becomes increasingly computationally expensive (Brown et al., 2020; Chang et al., 2024). LoRA (Hu et al., 2021) addresses this challenge by providing an effective fine-tuning methodology for scenarios with constrained computational resources. LoRA freezes model weights and injects trainable rank decomposition matrices, thereby modifying the behavior of dense linear layers without substantially changing the original model parameters (Lester et al., 2021; An et al., 2022). Recent studies (Zadouri et al., 2023; Liu et al., 2023; Dou et al., 2023) convincingly show that integrat-

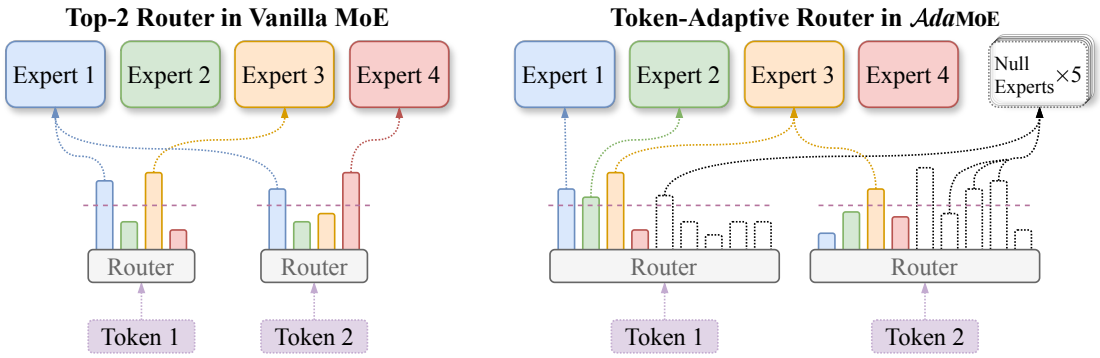

Figure 2: Comparison of Routing Mechanisms: vanilla MoE v.s. *Ada*MoE. **Left:** In vanilla MoE, each token selects the top 2 experts based on the routing probabilities. **Right:** *Ada*MoE introduces an additional set of *null experts* and makes each token select the top 4 experts, which can include both the true and null experts. For example, token 1 selects three true experts, while token 2 selects only one true expert. Despite this variation, the average number of true experts selected per token remains two, maintaining parity with the vanilla method.

ing LoRA with MoE offers a promising approach for achieving high performance with minimal parameter updates. Methods like MixLoRA (Li et al., 2024), MoLE (Wu et al., 2023), and LoRAMoE (Dou et al., 2023) combine MoE with LoRA by learning multiple pairs of low-rank matrices, known as LoRA experts, and use a router to compute the probabilities of each expert for the inputs. MoLA (Gao et al., 2024) explores the relationship between the number of LoRA experts and the depth of model layers. For consistency and convenience, we will refer to these methods collectively as Mo-LoRA in the following text.

2.2 Routing Strategies

The early MoE architecture utilized gate units as the router to select experts for each token (Shazeer et al., 2017; Lepikhin et al., 2020). Following the success of the Switch Transformer (Fedus et al., 2022) in large-scale pre-training, MoE received increased attention, leading to the development of more advanced routing algorithms. For example, BASE Layers (Lewis et al., 2021) use a linear assignment problem to maximize token-expert affinities while ensuring each expert receives an equal number of tokens. Hash layers (Roller et al., 2021) employ hashing techniques on input tokens to allocate different sets of weights. A different approach, Expert-Choice Routing (Zhou et al., 2022), allows experts to select their preferred tokens, achieving a more balanced expert load and better cost-effectiveness. Furthermore, DeepMind's Mixture-of-Depths (MoD) (Raposo et al., 2024) introduces a router to determine the necessity of computation for each input token at each layer.

3 Method

In this section, we introduce *Ada*MoE, which incorporates null experts to allow for more flexible and efficient expert selection for various tokens. An illustrative comparison between vanilla MoE and *Ada*MoE is presented in Figure 2.

3.1 Preliminary on MoE

MoE has been widely applied in two scenarios for large language models: MoE-LLMs (Jiang et al., 2024; Dai et al., 2024) and Mo-LoRA (Dou et al., 2023; Gao et al., 2024; Li et al., 2024) , as briefly introduced in Section 2. The core component of both is the MoE layer, which consists of n specialized experts $E_i : \mathbb{R}^{d_{in}} \to \mathbb{R}^{d_{out}}, i = 1, \ldots, n$ and a router $G : \mathbb{R}^{d_{in}} \to \mathbb{R}^n$. The experts often have the same parameterization, such as feed-forward neural networks (FFNs) in MoE-LLMs or LoRA modules in Mo-LoRA. The router usually activates the k ($k < n$) experts with the highest routing probabilities (i.e., the top-k experts) and distributes input tokens to corresponding experts.

Given an input token $x \in \mathbb{R}^{d_{in}}$, the routing process works as:

$$G(x) \in \mathbb{R}^n := \text{Softmax}\big(\text{TopK}(x \cdot W_g, k)\big) \ , \quad (1)$$

where $W_g \in \mathbb{R}^{d_{in} \times n}$ is the parameter matrix of the router, and $\text{TopK}(\cdot, k)$ retains only the top-k elements, setting the rest to $-\infty$ (so that after Softmax, the corresponding routing probabilities are zero).

The output of the MoE layer is then computed as:

$$y \in \mathbb{R}^{d_{out}} := \sum_{i=1}^{n} G(x)_i \cdot E_i(x) \ . \tag{2}$$

Additionally, an auxiliary loss is applied during the training stage to encourage a balanced load across experts within the same MoE layer (Fedus et al., 2022). Given a batch $\mathcal{B}$ of tokens, this load balancing loss for a MoE layer is defined as:

$$\ell_{load} := \alpha \cdot n \cdot \sum_{i=1}^{n} f_i \cdot P_i \ , \tag{3}$$

where α is a hyperparameter, and f_i represents the fraction of tokens dispatched to expert E_i,

$$f_i = \frac{1}{|\mathcal{B}|} \sum_{x \in \mathcal{B}} \mathbb{1}\{G(x)_i \neq 0\} \ , \tag{4}$$

and P_i denotes the average fraction of the router probability allocated for expert E_i, i.e.,

$$P_i = \frac{1}{|\mathcal{B}|} \sum_{x \in \mathcal{B}} \text{Softmax}\big(x \cdot W_g\big)_i \ . \tag{5}$$

3.2 Drawback of Top-k Router

Almost all traditional MoE methods adopt a top-k routing strategy for expert selection (Fedus et al., 2022; Lepikhin et al., 2020; Jiang et al., 2024). Therefore, each token passes through exactly k experts and occupies the same amount of computation. We first question the rationality of the fixed top-k routing with the following studies.

Concretely, the SocialIQA dataset (Sap et al., 2019) is fed into Mixtral-8x7B (Jiang et al., 2024), which employs a top-2 routing strategy for expert selection. We record the routing distribution for all tokens in each MoE layer of the model. To evaluate the sharpness of the routing distribution, we count the number of top experts whose cumulative routing probabilities exceed 50% and according to this, all tokens can be divided into four categories. The proportions of the tokens are displayed in Figure 3.

As shown, the proportions of tokens within different counts show substantial variation. Namely, the sharpness of the routing distribution varies significantly. A considerable number of tokens have highly uneven routing distributions. Some tokens tend towards a single expert, while a significant proportion of tokens distribute attention to more than 2 experts. These observations imply that the traditional fixed top-k routing strategy, which selects the same number of experts for each token, may not be optimal. This is also implied by the argument in MoD (Raposo et al., 2024) that some tokens may not need to pass through all MoE layers.

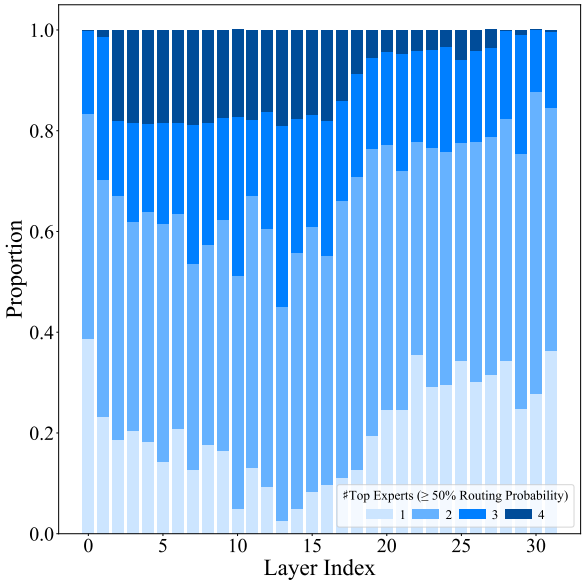

Figure 3: Proportions of the number of top experts with cumulative routing probabilities exceeding 50% for tokens in the SocialIQA dataset. Each bar represents the proportion of different counts of tokens at the corresponding MoE layer in Mixtral-8x7B.

3.3 Null Experts for Token-Adaptive Router

*Ada*MoE achieves token-adaptive expert selection by incorporating *null experts*, which are defined as an empty operation requiring zero FLOPs to process the token feature. In the context of LLMs, common operations satisfying this requirement include a constant zero mapping and an identity mapping (we take the zero mappings null expert as the default choice in the following just for simplicity). Consequently, an *Ada*MoE layer includes $n + m$ experts, where $\{E_i\}_{i=1}^{n}$ are true experts and $\{E_i\}_{i=n+1}^{n+m}$ are null experts, and a top-k router $G : \mathbb{R}^d \to \mathbb{R}^{n+m}$, which functions the same as the vanilla MoE router except for its output dimension.

Token-level adaptive routing. The router still performs fixed top-k selection but with k larger than in vanilla MoE. When null experts are chosen, no additional computation occurs due to their definition. Consequently, the number of true experts selected varies for different tokens.

Prespecified expert load. We can adjust the number of null experts according to the compute budget, and then reinforce the usage of null experts with a load balancing loss (see Section 3.4). This

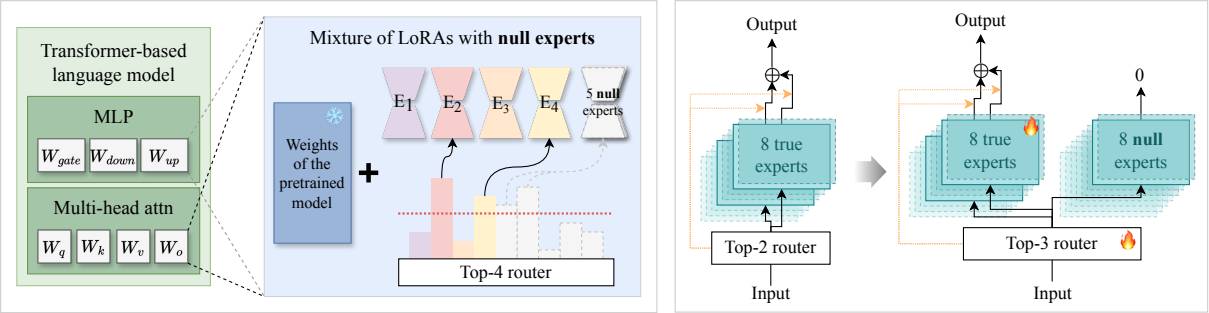

Figure 4: **Left**: Adding null experts to Mo-LoRA. **Right**: Adding null experts to the MoE layer of MoE-LLMs.

way, the load of true experts (or the overall FLOPs) can be easily adjusted to an appropriate degree.

Autoregressive task suitability. Expert-choice routing (Zhou et al., 2022) also allows varying numbers of experts for different tokens but struggles with autoregressive text generation since it requires considering both past and future tokens. In contrast, our token-choice method avoids this issue.

Bypassing MoE layers. MoD (Raposo et al., 2024) uses expert-choice routing to let tokens bypass some FFN layers, speeding up inference. Similarly, in *Ada*MoE, if all selected experts for a token are null experts, the token effectively bypasses the *Ada*MoE layer, achieving a similar effect.

3.4 More Details

Load balancing loss with null experts. Including null experts in the load balancing loss is necessary to prevent tokens from disproportionately selecting true experts. However, since all null experts are identical in nature, it is unnecessary to balance the load among them. Treating null experts as distinct entities for load balancing can significantly hinder performance, as shown in Table 3.

To address this, we modify the load balancing loss in Equation (3) as

$$\ell_{null} = \alpha \cdot (n+m) \cdot \sum_{i=1}^{n+m} \tilde{f}_i \cdot P_i \ , \qquad (6)$$

where

$$\tilde{f}_i = \begin{cases} f_i & \text{if } i \leq n \\ \frac{1}{m} \sum_{j=n+1}^{n+m} f_j & \text{if } i > n \end{cases} .$$

By using an average load among the null experts, we make no distinction between them, which can avoid unnecessary constraints on the router.

Load balancing constraints: from tight to loose. In practice, we anneal the weight α of our load balancing loss to chase a better balance-efficiency trade-off. In particular, we first set a larger α to enforce strict load balancing, ensuring tokens do not disproportionately select true experts, leading to a more even load distribution among all experts. In the latter, we use a smaller α to give tokens greater freedom in choosing experts. The empirical efficacy of doing so is verified in Table 5.

Normalization of routing probabilities. In vanilla MoE, $\text{TopK}(x \cdot W_g, k)$ is normalized using the Softmax activation function. With null experts, we have two options: 1) normalizing over all selected top-k experts, or 2) normalizing over only the true experts within the top-k ones. We choose the latter to ensure that the weighted average output by the *Ada*MoE layer remains consistent with the scale of that from the vanilla MoE layer.

3.5 Compatibility with Vanilla (MoE-)LLMs

*Ada*MoE is designed to be plug-and-play, able to be seamlessly integrated with pre-trained LLMs and MoE-LLMs, as illustrated in Figure 4. Due to resource constraints, we mainly focus on fine-tuning such models. For fine-tuning regular LLMs with the Mo-LoRA architecture, we need to add a randomly initialized router and multiple LoRA experts to the corresponding module. When applying *Ada*MoE to MoE-LLMs, the router's output dimensions are expanded to provide corresponding probabilities for null experts. The parameters for the new dimensions can be derived from the original parameter values. This ensures the expanded router balances the load across all experts, including both true and null experts, at the beginning of the fine-tuning process. For more specific implementation details, see Section 4.2.1. To fine-tune *Ada*MoE, we need to adjust the router and experts to meet our token-adaptive routing strategy and follow the detailed modifications outlined in Section 3.4 to achieve adaptive routing.

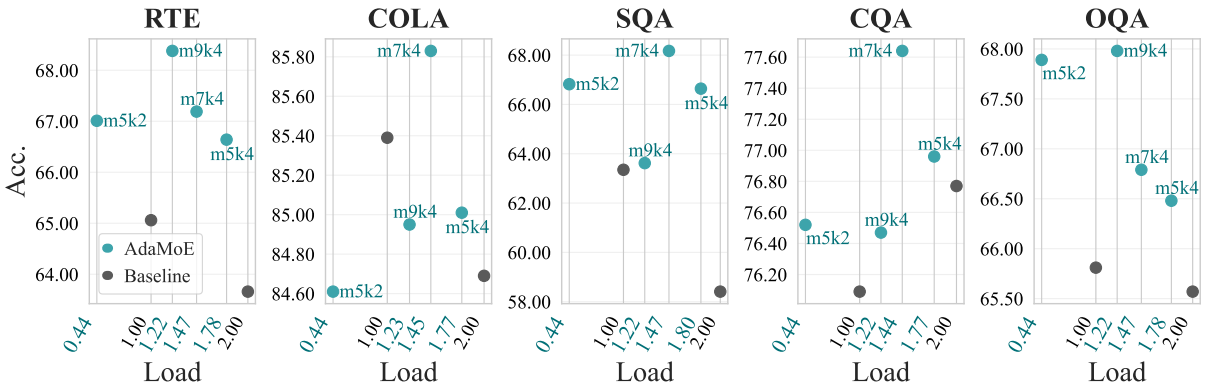

Figure 5: Performance comparison across five datasets: RTE, COLA, SQA, CQA, and OQA. The baseline is fine-tuned Llama2-7B using the vanilla Mo-LoRA method with top-1/top-2 routing. Acc. represents accuracy, and Load represents the average number of experts used per Mo-LoRA module or *Ada*MoE layer. *Ada*MoE use different configurations: m5k2 (5 null experts, top-2 selection), m9k4, m7k4 and m5k4. As shown, *Ada*MoE achieves higher accuracy across almost all datasets compared to the baseline. The exact accuracy values can be found in Table 7.

4 Experiments

In this section, we demonstrate the superior performance of *Ada*MoE across various benchmarks, particularly in reducing expert load and enhancing task performance through its token-adaptive routing strategy. We first apply our method to regular LLMs with the Mo-LoRA architecture (Section 4.1). Then apply it to traditional MoE-LLMs (Section 4.2). Additionally, We also include extensive ablation studies to provide further insights into our approach's effectiveness.

4.1 Application to Regular LLMs

4.1.1 Experiments Setup

Model and datasets. We select Llama2-7B (Touvron et al., 2023) as our base model due to its strong performance and popularity within the AI community. To validate the effectiveness of our method, we evaluate it on two distinct task types using five widely recognized datasets. The first task focuses on semantic understanding, for which we use two datasets from the renowned GLUE Benchmark (Wang et al., 2018): Recognizing Textual Entailment (RTE) and the Corpus of Linguistic Acceptability (COLA). The second task involves commonsense reasoning and includes the following datasets: ScienceQA (SQA) (Lu et al., 2022), CommonsenseQA (CQA) (Talmor et al., 2018), and OpenBookQA (OQA) (Mihaylov et al., 2018).

Baseline and implementation details. To highlight our method's significance, we use the typical Mo-LoRA method as the baseline for comparison. For each MoE/*Ada*MoE layer, we set $n = 4$ (4 true experts). For the baseline, we set $k = 1, 2$ for the top-k routing strategy, which are the most common choices. For our *Ada*MoE, we selected various configurations for k (the number of top-k experts) and m (the number of null experts). We use AdamW (Loshchilov and Hutter, 2017) as the optimizer with a learning rate of 3e-4. The rank of each LoRA expert is set to 8, and the initialization of the LoRA modules follows the original LoRA implementation (Hu et al., 2021). For each LLM layer, we applied LoRA to (W_q, W_k, W_v, W_o) in the self-attention modules and $(W_{gate}, W_{down}, W_{up})$ in the MLP modules. We trained on each dataset for 2 epochs, using 3 random seeds, and averaged the results to obtain the final performance metrics.

4.1.2 Experiments Results

The results are shown in Figure 5. We use accuracy as the main metric to evaluate the model's performance [1]. It is evident that *Ada*MoE achieves higher accuracy across almost all datasets compared to the traditional baseline. For instance, on the RTE and OQA datasets, all configurations of *Ada*MoE surpass the baseline in accuracy. This trend continues across the other datasets, demonstrating the robustness and effectiveness of *Ada*MoE in achieving better performance with more adaptive expert utilization.

4.2 Application to MoE-LLMs

4.2.1 Experiments Setup

Model and datasets. We use Mixtral-8x7B (Jiang et al., 2024) as the base model, where each

[1] LoRA expert load has minimal impact on total FLOPs; therefore, it is not considered a primary evaluation metric.

	Metric	WINO	HELLA	PIQA	SIQA	OQA	ARC-C	Avg.
Original Mixtral-8x7B	Acc.	55.96	53.62	68.06	64.59	65.40	83.73	65.23
Fine-tuned Mixtral-8x7B	Acc.	80.43	84.10	**90.48**	76.36	**89.00**	87.46	84.64
*Ada*MoE	Acc.	**81.93**	**85.50**	90.32	**76.97**	88.20	**89.15**	**85.35**
	%FLOPs↓	14.99	14.10	18.07	16.31	13.22	14.55	15.21
	Load	1.66	1.68	1.59	1.63	1.70	1.67	1.66

Table 1: Comparison of performance and computational efficiency across six datasets: WINO, HELLA, PIQA, SIQA, OQA and ARC-C. Metrics include Acc. (accuracy), %FLOPs↓ (percentage of FLOPs reduction by *Ada*MoE compared to the baselines), and Load (the average number of experts used per MoE/*Ada*MoE layer). The baselines are original/fine-tuned Mixtral-8x7B, both using the top-2 routing strategy (Load = 2.00). *Ada*MoE not only reduces FLOPs but also achieves better accuracy across most datasets compared to the fine-tuned Mixtral-8x7B with LoRA.

MoE layer has 8 FFN experts and a top-2 router. We selected six well-known datasets from different categories for our experiments: WinoGrande (WINO) (Sakaguchi et al., 2021) for coreference resolution, Hellaswag (HELLA) (Zellers et al., 2019), PIQA (Bisk et al., 2020), and SIQA (Sap et al., 2019) for commonsense reasoning, OpenBookQA (OQA) (Mihaylov et al., 2018) for reading comprehension, and ARC-Challenge (ARC-C) (Clark et al., 2018) for science examination.

Baseline and implementation details. Due to the substantial resources required for pre-training, we focus on fine-tuning. To save memory, we use 4-bit quantization and the QLoRA method (Dettmers et al., 2024). The LoRA target modules for the baseline are `gate`, `w1`, `w2`, and `w3`. For our *Ada*MoE, we modify this architecture as described in Section 3.5. Specifically, we add null experts to each MoE layer, and the router expands its output dimension to assign probabilities to all experts. To simplify the modification, we define an additional module, `gate2`, whose parameters can be derived from `gate`. [2] Together, `gate` and `gate2` form the router that assigns weights to all experts. Thus, the LoRA target modules for our method are `gate`, `gate2`, `w1`, `w2`, and `w3`. The rank of the LoRA module is set to 8, and the learning rate is 5e-5. Due to the tendency of MoE-LLMs to overfit during fine-tuning, we use 1000 samples for training on each dataset and train for 2 epochs. In the 2 epochs, we set different values of α in Equation (6) to $\alpha_1 = 0.02$, $\alpha_2 = 0.0001$, as described in Section 3.4. All evaluations are conducted using OpenCompass (Contributors, 2023) to assess accuracy.

[2]For instance, if `gate2` has an output dimension of 16, meaning there are 16 null experts, the parameters of `gate2` can be copied from `gate` in two segments.

4.2.2 Experiment Results

In this section, we present the results for the configuration with $m = 8$ and $k = 3$ (i.e., 8 null experts and top-3 expert selection), as shown in Table 1. Additional results are in Section 4.3 and Table 8.

Accuracy. *Ada*MoE outperforms the baseline on WinoGrande, HellaSwag, SIQA, and ARC-Challenge. Although the baseline slightly surpasses *Ada*MoE on PIQA and OpenBookQA, *Ada*MoE achieves a higher average accuracy.

FLOPs. FFNs account for the majority of the FLOPs during inference. This issue is exacerbated in Mixtral-8x7B, which replaces the FFN with a set of 8 FFNs and selects the top-2 during each inference step. This greatly increases the computational load. *Ada*MoE significantly reduces FLOPs across all datasets, achieving an average reduction of 15.21% compared to the baseline. This demonstrates that *Ada*MoE is more computationally efficient while maintaining competitive performance.

Load. The Load metric indicates the average number of experts used per MoE/*Ada*MoE layer. The baseline method has a Load of 2. In contrast, *Ada*MoE achieves a lower average Load of 1.66, indicating more efficient utilization of experts.

Overall, the results confirm the effectiveness of the token-adaptive mechanism in improving both computational efficiency and model performance.

4.3 Ablation

In this section, we provide results for various m and k, beyond the single configuration shown in Section 4.2. We also present ablation studies for *Ada*MoE, corresponding to Section 3.4. Additional ablation experiments can be found in Appendix A.

More results for Section 4.2. We tested different combinations of m and k on the SIQA

	Baseline	$\mathcal{A}da$MoE							
m, k	$0, 2$	$8, 3$	$16, 4$	$32, 4$	$32, 5$	$32, 6$	$40, 6$	$40, 7$	$40, 8$	$48, 8$
Acc.	76.36	**76.97**	**76.92**	66.27	72.93	**76.46**	69.86	76.05	**77.23**	74.67
Load	2.00	1.63	1.66	0.77	1.05	1.54	1.01	1.49	1.64	1.48

Table 2: Performance of different m and k combinations on the SIQA dataset. The Baseline represents fine-tuned Mixtral-8x7B using LoRA method, with a Load of 2. Bold values indicate accuracy higher than the baseline.

	RTE		COLA		SQA		OQA	
	Acc.	Load	Acc.	Load	Acc.	Load	Acc.	Load
ℓ_{bal}	56.68	1.77	83.68	1.77	65.65	1.78	69.80	1.76
ℓ_{null}	**67.51**	1.77	**85.01**	1.77	**66.64**	1.80	**71.40**	1.77

Table 3: Comparison of accuracy and load on four datasets using load balancing loss with and without balancing among null experts. ℓ_{bal} represents the loss with load balancing constraints among null experts, and ℓ_{null} represents the loss without these constraints. Bold values indicate higher accuracy.

	SIQA	
Option	Acc.	Load
1)	80.19	1.50
2)	81.27	1.54

Table 4: 1) Normalizing all selected top-k experts, and 2) normalizing only the true experts within the top-k.

dataset, with results shown in Table 2. Compared to the $m = 8, k = 3$ configuration in Section 4.2, $\mathcal{A}da$MoE can further reduce the expert load (FLOPs) while maintaining competitive performance. For example, with $m = 32, k = 6$, the expert load is 1.54 (79.57% of baseline FLOPs), yet accuracy remains higher than the baseline. There are also accuracy differences among configurations with similar loads. For instance, $m = 40, k = 7$ and $m = 48, k = 8$ have nearly identical loads but differ in accuracy. This discrepancy highlights areas for further exploration in future research.

Load balancing loss with null experts. To verify the effectiveness of the modified load balancing loss introduced in Equation (6), we selected two datasets from each of the semantic understanding and commonsense reasoning tasks. The results, illustrated in Table 3, show that lifting the load balancing constraints among null experts significantly improves the performance of the fine-tuned model on the RTE, COLA, SQA, and OQA datasets.

Load balancing constraints: from tight to loose. The effectiveness of the annealing training process described in Section 3.4 is validated in Table 5. The tight load balancing constraints in the first epoch effectively control the expert load in $\mathcal{A}da$MoE, meeting our expectations. The loose constraints in the second epoch allow tokens greater freedom in selecting experts, thereby enhancing performance with almost no increase in expert load. For example, on the WINO dataset, the accuracy increased by 5.69% compared to the result after epoch 1, with almost no increase in expert load.

		WINO		SIQA	
		Acc.	Load	Acc.	Load
α_1	Baseline	78.14	2.00	75.38	2.00
	$\mathcal{A}da$MoE	76.24	1.65	75.90	1.62
α_2	Baseline	80.43	2.00	76.36	2.00
	$\mathcal{A}da$MoE	81.93	1.66	76.97	1.63

Table 5: Performance for finetuning Mixtral-8x7B with $\mathcal{A}da$MoE on the WINO and SIQA datasets for two epochs with $\alpha_1 = 0.02$ and $\alpha_2 = 0.0001$.

Normalization of routing probabilities. We tried the two options mentioned in Section 3.4 on the SIQA dataset, and the results are shown in Table 4. As we can see, option 2) is a superior choice, showing a significant improvement in accuracy , with only a minor change in expert load.

5 Conclusion

MoE has been a promising method for training powerful models with fewer parameters. In this paper, we introduced $\mathcal{A}da$MoE, which uses null experts to enable token-level adaptive expert allocation and overcome the drawbacks of fixed expert allocation. Extensive experiments validate its effectiveness. The $\mathcal{A}da$MoE approach significantly enhances efficiency and adaptability, paving the way for more capable large language models.

Limitations

One potential drawback of this work is that we did not pre-train a MoE-LLM using our *Ada*MoE method. Pre-training an MoE-LLM would have allowed us to thoroughly evaluate the full capabilities and performance improvements of our method, but the significant resources required made it impractical for our current study. Additionally, we did not explore the scenario of null experts as identity mappings, where null experts would also need zero FLOPs to process input tokens. We hypothesize that this approach might accelerate training convergence because null experts as identity mappings would potentially update their corresponding router parameters more frequently.

We acknowledge these limitations and leave these aspects for future work. Addressing these issues could provide a more comprehensive evaluation of the *Ada*MoE method and potentially uncover additional benefits or areas for improvement.

References

Shengnan An, Yifei Li, Zeqi Lin, Qian Liu, Bei Chen, Qiang Fu, Weizhu Chen, Nanning Zheng, and Jian-Guang Lou. 2022. Input-tuning: Adapting unfamiliar inputs to frozen pretrained models. *arXiv preprint arXiv:2203.03131*.

Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. 2020. Piqa: Reasoning about physical commonsense in natural language. In *Thirty-Fourth AAAI Conference on Artificial Intelligence*.

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

Yupeng Chang, Xu Wang, Jindong Wang, Yuan Wu, Linyi Yang, Kaijie Zhu, Hao Chen, Xiaoyuan Yi, Cunxiang Wang, Yidong Wang, et al. 2024. A survey on evaluation of large language models. *ACM Transactions on Intelligent Systems and Technology*, 15(3):1–45.

Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E. Gonzalez, Ion Stoica, and Eric P. Xing. 2023. Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. 2023. Palm: Scaling language modeling with pathways. *Journal of Machine Learning Research*, 24(240):1–113.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv:1803.05457v1*.

OpenCompass Contributors. 2023. Opencompass: A universal evaluation platform for foundation models. https://github.com/open-compass/opencompass.

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

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. 2024. Qlora: Efficient finetuning of quantized llms. *Advances in Neural Information Processing Systems*, 36.

Shihan Dou, Enyu Zhou, Yan Liu, Songyang Gao, Jun Zhao, Wei Shen, Yuhao Zhou, Zhiheng Xi, Xiao Wang, Xiaoran Fan, et al. 2023. Loramoe: Revolutionizing mixture of experts for maintaining world knowledge in language model alignment. *arXiv preprint arXiv:2312.09979*.

William Fedus, Barret Zoph, and Noam Shazeer. 2022. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *Journal of Machine Learning Research*, 23(120):1–39.

Chongyang Gao, Kezhen Chen, Jinmeng Rao, Baochen Sun, Ruibo Liu, Daiyi Peng, Yawen Zhang, Xiaoyuan Guo, Jie Yang, and VS Subrahmanian. 2024. Higher layers need more lora experts. *arXiv preprint arXiv:2402.08562*.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*.

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

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. 2024. Mixtral of experts. *arXiv preprint arXiv:2401.04088*.

Dmitry Lepikhin, HyoukJoong Lee, Yuanzhong Xu, Dehao Chen, Orhan Firat, Yanping Huang, Maxim Krikun, Noam Shazeer, and Zhifeng Chen. 2020. Gshard: Scaling giant models with conditional computation and automatic sharding. *arXiv preprint arXiv:2006.16668*.

Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The power of scale for parameter-efficient prompt tuning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3045–3059.

Mike Lewis, Shruti Bhosale, Tim Dettmers, Naman Goyal, and Luke Zettlemoyer. 2021. Base layers: Simplifying training of large, sparse models. In *International Conference on Machine Learning*, pages 6265–6274. PMLR.

Dengchun Li, Yingzi Ma, Naizheng Wang, Zhiyuan Cheng, Lei Duan, Jie Zuo, Cal Yang, and Mingjie Tang. 2024. Mixlora: Enhancing large language models fine-tuning with lora based mixture of experts. *arXiv preprint arXiv:2404.15159*.

Haokun Liu, Derek Tam, Mohammed Muqeeth, Jay Mohta, Tenghao Huang, Mohit Bansal, and Colin A Raffel. 2022. Few-shot parameter-efficient fine-tuning is better and cheaper than in-context learning. *Advances in Neural Information Processing Systems*, 35:1950–1965.

Qidong Liu, Xian Wu, Xiangyu Zhao, Yuanshao Zhu, Derong Xu, Feng Tian, and Yefeng Zheng. 2023. Moelora: An moe-based parameter efficient fine-tuning method for multi-task medical applications. *arXiv preprint arXiv:2310.18339*.

Ilya Loshchilov and Frank Hutter. 2017. Decoupled weight decay regularization. *Preprint*, arXiv:1711.05101.

Pan Lu, Swaroop Mishra, Tanglin Xia, Liang Qiu, Kai-Wei Chang, Song-Chun Zhu, Oyvind Tafjord, Peter Clark, and Ashwin Kalyan. 2022. Learn to explain: Multimodal reasoning via thought chains for science question answering. *Advances in Neural Information Processing Systems*, 35:2507–2521.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. 2018. Can a suit of armor conduct electricity? a new dataset for open book question answering. *arXiv preprint arXiv:1809.02789*.

Qwen. 2024. Qwen1.5-moe: Matching 7b model performance with 1/3 activated parameters. https://qwenlm.github.io/blog/qwen-moe.

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

Stephen Roller, Sainbayar Sukhbaatar, Jason Weston, et al. 2021. Hash layers for large sparse models. *Advances in Neural Information Processing Systems*, 34:17555–17566.

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

Maarten Sap, Hannah Rashkin, Derek Chen, Ronan LeBras, and Yejin Choi. 2019. Socialiqa: Commonsense reasoning about social interactions. *arXiv preprint arXiv:1904.09728*.

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. 2017. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *arXiv preprint arXiv:1701.06538*.

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. 2018. Commonsenseqa: A question answering challenge targeting commonsense knowledge. *arXiv preprint arXiv:1811.00937*.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. 2018. Glue: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461*.

Xun Wu, Shaohan Huang, and Furu Wei. 2023. Mole: Mixture of lora experts. In *The Twelfth International Conference on Learning Representations*.

Ted Zadouri, Ahmet Üstün, Arash Ahmadian, Beyza Ermiş, Acyr Locatelli, and Sara Hooker. 2023. Pushing mixture of experts to the limit: Extremely parameter efficient moe for instruction tuning. *arXiv preprint arXiv:2309.05444*.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. Hellaswag: Can a machine really finish your sentence? *arXiv preprint arXiv:1905.07830*.

Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, et al. 2022. Opt: Open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068*.

Yanqi Zhou, Tao Lei, Hanxiao Liu, Nan Du, Yanping Huang, Vincent Zhao, Andrew M Dai, Quoc V Le, James Laudon, et al. 2022. Mixture-of-experts with expert choice routing. *Advances in Neural Information Processing Systems*, 35:7103–7114.

A Additional Ablation

Robustness. We primarily considered the impact of two hyperparameters, the number of epochs and the rank of LoRA module. We evaluated their impact on the SQA dataset, as shown in Table 6. Regardless of the number of training epochs and rank of the LoRA, our method outperforms the baseline consistently. Therefore, we can conclude that our method demonstrates strong robustness across different epochs and ranks of LoRAs.

B Additional Results

Exact values for results in Figure 5. Table 7 presents the exact values corresponding to Figure 5, averaged from results with three random seeds. The performance metrics evaluated include accuracy and average number of experts used per Mo-LoRA module/*Ada*MoE layer across five datasets: RTE, COLA, SQA, CQA, and OQA.

For the baseline model, two configurations (k1 and k2) are tested. The k1 configuration, with a fixed top-1 routing strategy, achieves an accuracy of 65.06 on RTE, 85.39 on COLA, 63.35 on SQA, 76.09 on CQA, and 65.81 on OQA. The k2 configuration, with a fixed top-2 routing strategy, results in slightly lower accuracy values of 63.66 on RTE, 84.69 on COLA, 58.41 on SQA, 76.77 on CQA, and 65.57 on OQA.

The *Ada*MoE model is evaluated with four configurations (m5k4, m7k4, m9k4, and m5k2). The m5k4 configuration achieves an accuracy of 66.64 on RTE, 85.01 on COLA, 66.64 on SQA, 76.96 on CQA, and 66.48 on OQA, with a load ranging from 1.77 to 1.80. The m7k4 configuration shows improved accuracy, reaching 67.19 on RTE, 85.83 on COLA, 68.17 on SQA, 77.64 on CQA, and 66.79 on OQA, with a load between 1.44 and 1.47. The m9k4 configuration presents an accuracy of 68.38 on RTE, 84.95 on COLA, 63.62 on SQA, 76.47 on CQA, and 67.98 on OQA, with a load consistently around 1.22. Lastly, the m5k2 configuration records an accuracy of 67.01 on RTE, 84.61 on COLA, 66.82 on SQA, 76.52 on CQA, and 67.89 on OQA, with a significantly lower load of 0.44.

Additional results for Section 4.2. Table 8 presents the performance of various m and k combinations for the *Ada*MoE model across different datasets, including ARC-C, HELLA, OQA, PIQA, and WINO. These results supplement the experimental findings discussed in Section 4.2.

For the **ARC-C** dataset, the baseline model with $m = 0$ and $k = 2$ achieved an accuracy of 87.46 and a load of 2.00. The *Ada*MoE configurations demonstrated varying performance, with the highest accuracy of 89.15 observed for $m = 8$ and $k = 3$, accompanied by a load of 1.67. As the values of m and k increased, the load generally decreased, with the lowest load of 1.34 recorded for $m = 40$ and $k = 8$. On the **HELLA** dataset, the baseline model achieved an accuracy of 84.10 and a load of 2.00. The *Ada*MoE model's best performance was observed with $m = 8$ and $k = 3$, achieving an accuracy of 85.50 and a load of 1.68. Similar to the ARC-C results, higher values of m and k led to reduced loads, with a minimum load of 1.37 for $m = 40$ and $k = 7$. For the **OQA** dataset, the baseline model achieved an accuracy of 89.94 and a load of 2.00. The *Ada*MoE configurations showed varying results, with the highest accuracy of 89.2 observed for $m = 16$ and $k = 4$, and the load varying between 1.49 and 1.71. The lowest load of 1.50 was recorded for $m = 40$ and $k = 8$. In the case of the **PIQA** dataset, the baseline model reached an accuracy of 90.48 and a load of 2.00. The best accuracy among the *Ada*MoE configurations was 90.32 for $m = 8$ and $k = 3$, with a load of 1.59. The load decreased as m and k values increased, reaching a minimum of 1.32 for $m = 40$ and $k = 7$. Finally, on the **WINO** dataset, the baseline model achieved an accuracy of 80.43 and a load of 2.00. The highest accuracy of 81.93 was observed for $m = 8$ and $k = 3$, with a load of 1.66. The load showed a decreasing trend with increasing values of m and k, with the lowest load of 1.45 recorded for $m = 40$ and $k = 8$.

C Additional Discussion

The top-p router can also implement token-adaptive expert selection. It selects experts based on the sum of routing probabilities exceeding a threshold p. This allows for a variable number of experts to be chosen for different tokens. However, compared to our *Ada*MoE, this approach has the following drawbacks:

1. The value of p cannot be predefined according to the compute budget, and finding an appropriate p often requires multiple attempts.

2. It cannot enable tokens to bypass some layers.

Moreover, our method is actually compatible with the top-p approach. We can incorporate null experts and simultaneously use top-p. This compatibility opens up avenues for further exploration in the future.

		Epoch		Rank of LoRAs	
		1	10	8	32
Baseline	Acc.	45.95	87.19	45.95	46.72
	Load	2.00	2.00	2.00	2.00
$\mathcal{A}da$MoE	Acc.	**48.88**	**88.54**	**48.88**	**49.01**
	Load	**1.92**	**1.88**	**1.92**	**1.89**

Table 6: Robustness of our method under different epochs and ranks of LoRAs.

		Metric	**RTE**	**COLA**	**SQA**	**CQA**	**OQA**
Baseline	k1	Acc.	65.06	85.39	63.35	76.09	65.81
		Load	1.00	1.00	1.00	1.00	1.00
	k2	Acc..	63.66	84.69	58.41	76.77	65.57
		Load	2.00	2.00	2.00	2.00	2.00
$\mathcal{A}da$MoE	m5k4	Acc.	66.64	85.01	66.64	76.96	66.48
		Load	1.78	1.77	1.80	1.77	1.78
	m7k4	Acc.	67.19	85.83	68.17	77.64	66.79
		Load	1.47	1.45	1.47	1.44	1.47
	m9k4	Acc.	68.38	84.95	63.62	76.47	67.98
		Load	1.22	1.23	1.22	1.22	1.22
	m5k2	Acc.	67.01	84.61	66.82	76.52	67.89
		Load	0.44	0.44	0.44	0.44	0.44

Table 7: Exact values for Figure 5, averaged from results with 3 random seeds.

ARC-C	Baseline			$\mathcal{A}da$MoE			
m, k	0,2	8,3	16,4	24,5	32,6	40,7	40,8
Acc.	87.46	89.15	87.12	86.10	85.08	86.10	85.76
Load	2.00	1.67	1.70	1.56	1.49	1.59	1.34
HELLA	Baseline			$\mathcal{A}da$MoE			
m, k	0,2	8,3	16,4	24,5	32,6	40,7	40,8
Acc.	84.10	85.50	83.10	81.30	80.40	82.50	79.20
Load	2.00	1.68	1.64	1.45	1.39	1.37	1.44
OQA	Baseline			$\mathcal{A}da$MoE			
m, k	0,2	8,3	16,4	24,5	32,6	40,7	40,8
Acc.	89 94	88.2	89.2	86.6	86.8	85	82.6
Load	2.00	1.70	1.71	1.49	1.54	1.56	1.50
PIQA	Baseline			$\mathcal{A}da$MoE			
m, k	0,2	8,3	16,4	24,5	32,6	40,7	40,8
Acc.	90.48	90.32	89.99	88.30	86.67	86.78	85.42
Load	2.00	1.59	1.53	1.46	1.39	1.32	1.33
WINO	Baseline			$\mathcal{A}da$MoE			
m, k	0,2	8,3	16,4	24,5	32,6	40,7	40,8
Acc.	80.43	81.93	79.32	78.17	77.66	71.43	79.16
Load	2.00	1.66	1.72	1.71	1.73	1.59	1.45

Table 8: Performance of more m and k combinations on various datasets. As a supplement to the experimental results in Section 4.2.