Union-of-Experts: Neurons in Mixture-of-Experts are Secretly Routers

Songhao Wu[1] **Ang Lv**[1] **Ruobing Xie**[2]*
Xingwu Sun[2] **Di Wang**[2] **Rui Yan**[1]* **Yankai Lin**[1]*
[1] Gaoling School of Artificial Intelligence, Renmin University of China
[2] Large Language Model Department, Tencent
{songhaowu, yankailin}@ruc.edu.cn xrbsnowing@163.com

Abstract

Mixture-of-Experts (MoE) models rely on an external router to assign tokens to experts. This design inherently separates the routing decision from each expert's internal capabilities, leading to suboptimal performance. In this work, we address this limitation with Union-of-Experts (UoE), an MoE variant that performs "expert-autonomous routing". The core mechanism of UoE is to pre-designate a minute fraction of neurons within each expert as routing neurons. Experts autonomously select relevant tokens by comparing the activation intensity of these neurons, aligning routing decisions with each expert's functional profile. To prevent the waste of activations from unselected experts, we aggregate all routing neuron outputs and sum them into the final layer output. This aggregation acts as a novel virtual shared expert whose parameters are distributed across the individual experts, and improves overall parameter efficiency. We pre-train UoE models with up to 3B parameters, demonstrating that they outperform traditional MoEs with matched efficiency. Furthermore, our analysis of the routing neurons provides valuable insights into expert-autonomous selection and advances the understanding of MoE routing.

1 Introduction

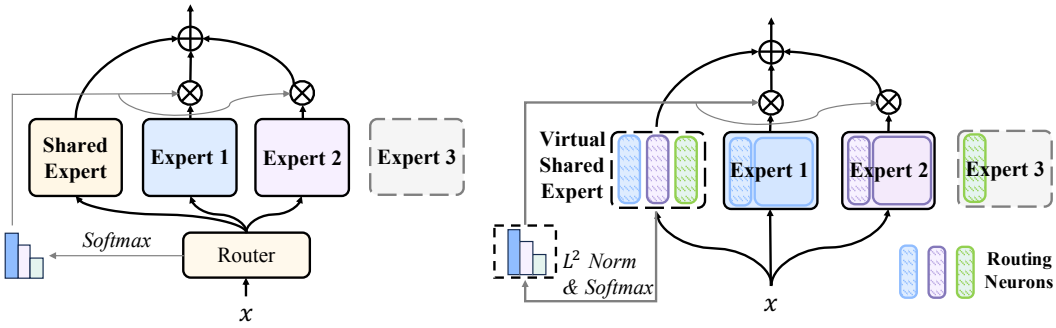
Mixture-of-Experts (MoE) has emerged as a centerpiece of language model research. A series of LLMs built on the MoE architecture have been proposed (DeepSeek-AI et al., 2025; OpenAI, 2025; Jiang et al., 2024; Muennighoff et al., 2025), demonstrating competitive performance across a wide range of downstream tasks. MoE-based Transformers replace the traditional feed-forward network (FFN) with an ensemble of expert modules, utilizing a router to dispatch the input tokens to a select subset of experts. Such sparse activation mechanism enables the scaling of models to trillions of parameters while maintaining manageable overhead (Team et al., 2025), establishing MoE as a cornerstone of modern LLM landscape.

However, a limitation persists in MoE router design: routing decisions are decoupled from expert execution. As router operates in isolation, it lacks direct access to the experts' weights and remains unaware of their capabilities. When tokens are inappropriately routed, experts are forced to accommodate mismatched inputs, leading to inefficient trial-and-error learning. To address this, the concept of "expert autonomy" was proposed (Lv et al., 2025), wherein all experts process each token and those with the largest activation norm are selected. While this is conceptually sound, it incurs a prohibitive computational overhead, particularly as the number of experts continues to scale.

In this paper, we introduce Union-of-Experts (**UoE**), an MoE framework designed to reconcile the benefits of expert autonomy with affordable computational overhead. Figure 1 provides a comparative overview of our UoE against the traditional MoE architecture. Specifically, we designate a select subset of N_s neurons within each expert to serve as a proxy for its global activation level. Routing weights are then allocated given the activation intensity of each token across these "routing neurons", thereby enabling expert autonomy without the high cost of full-expert execution across the entire hidden dimension D. The learned routing strategy uncover a surprising finding: only a minute fraction of neurons is required to parameterize the routing function. This scales down the computational cost of expert autonomy to N_s/D, where N_s is much smaller than the hidden dimension D, achieving a significant reduction in overhead. By simply pre-designating the first N_s neurons in expert's weight matrix as routing neurons, their activations spontaneously exhibit high correlation with those of the entire weight matrix. Through an anal-

*Correspondence to: Ruobing Xie, Rui Yan, Yankai Lin.

Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 36193–36206



(a) Mixture-of-Experts

(b) Union-of-Experts

Figure 1: Schematic comparison of token routing between MoE and UoE, with inactive modules shaded in gray. (a) Traditional MoE dispatches tokens to experts with the highest routing logits using an external router. (b) UoE routes tokens to the experts whose routing neurons are top-activated. Hatching denotes the routing neurons in each expert, which collectively form a virtual shared expert.

ysis of model weights, we further interpret how this expert autonomy ability emerges and is encoded in the routing neurons during pre-training.

Routing neurons endow UoE with expert autonomy and obviate the challenges of accurate activation intensity calculation. We provide both empirical and theoretical evidence that UoE is more efficient than AoE, the state-of-the-art expert autonomy practice. However, pre-computing the routing neurons for each expert still incurs overhead. Since unselected routing neurons will terminate early within the computation graph, their pre-computation leads to redundant activations and wasted FLOPs. To address this, we propose to reuse these neurons and treat them as a conceptual shared expert. Specifically, we perform a reduction across all routing neuron outputs and integrate this aggregate into the final weighted sum of experts. This behaves equivalent to packing routing neurons from all experts to recover a standard shared expert. Figure 1 provides an illustration, and we refer to this architecture a virtual shared expert. While this design was intended to improve computational efficiency, we find these units can function comparably to a vanilla shared expert design. More empirical evidence can be found in the subsequent sections.

We refer to our routing design the Union-of-Experts (UoE). This nomenclature draws an analogy in which routing neurons serve as representatives for experts, while the virtual shared expert functions as a labor union to facilitate autonomous selection and collective synergy.

We pre-train UoE with up to 3B parameters, achieving superior performance over both MoE and AoE, while keeping the inference cost on par with vanilla MoE. We also investigate the behavior of UoE with properties of general interest, such as load balancing. Additionally, we provide a post-hoc analysis into the routing neurons to provide an explanation for how expert autonomy emerges within UoE models. We hope this finding serves as a catalyst for future research and can help inform the future design of MoE routing. Our code is available at `https://github.com/ericshwu/Union-of-Experts`.

2 Background and Notation

2.1 Mixture-of-Experts (MoE)

Following mainstream MoE designs (Dai et al., 2024; Jiang et al., 2024), we adopt the Gated Linear Unit (Shazeer, 2020) as the expert module. The i-th expert is parameterized by three matrices:

$$\mathbf{W}_g^i \in \mathbb{R}^{d\times D}, \mathbf{W}_p^i \in \mathbb{R}^{d\times D} \text{ and } \mathbf{W}_o^i \in \mathbb{R}^{D\times d},$$

with its forward pass defined as:

$$\mathrm{E}_i(\boldsymbol{x}) = \left(\texttt{SiLU}\left(\boldsymbol{x}\boldsymbol{W}_g^i\right) \odot \left(\boldsymbol{x}\boldsymbol{W}_p^i\right)\right) \boldsymbol{W}_o^i. \quad (1)$$

The MoE FFN layer consists of N experts, with K experts activated per input token $\boldsymbol{x}$. Following (Dai et al., 2024), we incorporate a shared expert E_s that processes all tokens. The output of an MoE FFN layer is the sum of two components: the output of a shared expert and a weighted sum of the selected expert outputs. The weights for the latter are given by a router parameterized by matrix $\boldsymbol{R} \in \mathbb{R}^{d\times N}$:

$$\mathrm{G}(\boldsymbol{x}) = \mathrm{softmax}\left(\boldsymbol{x}\boldsymbol{R}\right),$$
$$\mathrm{FFN}(\boldsymbol{x}) = \mathrm{E}_s(\boldsymbol{x}) + \sum_{i\in \mathrm{TopK}(\mathrm{G}(\boldsymbol{x}))} \mathrm{G}(\boldsymbol{x})[i]\ \mathrm{E}_i(\boldsymbol{x}). \quad (2)$$

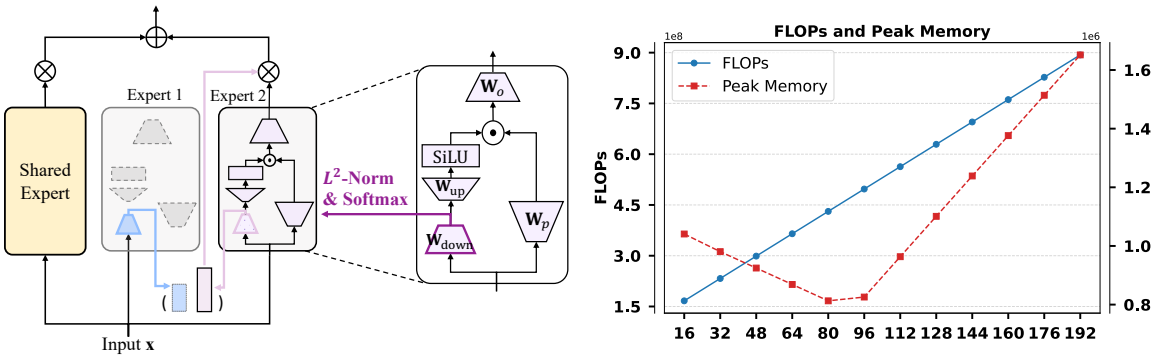


Figure 2: An overview of the AoE model is presented on the left. We contend that AoE faces an efficiency bottleneck inherent to its factorization scheme; specifically, for any rank r, the model is bounded by either computational or memory overhead. The plot on the right characterizes these theoretical costs as a function of r.

2.2 Autonomy-of-Experts (AoE)

AoE (Lv et al., 2025) resolves the discrepancy between token routing and expert execution by encoding the routing logic into the expert parameters. The grounded insight is that experts inherently "know" their domain of expertise; specifically, an expert's intermediate activations signals its alignment with a given input token.

Few studies have investigated similar solutions to this problem (Pham et al., 2024; Lv et al., 2026). While Pham et al. (2024) propose using experts' output activations as distillation labels for router logits, this approach requires full expert computation, rendering it computationally prohibitive. AoE factorizes $\boldsymbol{W}_g^i$ into two low-rank matrices to mitigate the computational overhead:

$$\boldsymbol{W}_{down}^i \in \mathbb{R}^{d\times r} \quad \text{and} \quad \boldsymbol{W}_{up}^i \in \mathbb{R}^{r\times D'}.$$

To maintain a parameter budget consistent with the baseline MoE, the dimension D' is given as:

$$D' = \frac{3Dd - dr}{r + 2d}.$$

Each token is multiplied by all $\boldsymbol{W}_{down}^i$ matrices, and the L^2-norms of the resulting N activations (each of dimension r) are used for expert selection. Experts with the top-K activation norms continue forward computation, while unselected experts terminate early. The routing function G and the forward pass for selected experts are defined as:

$$\mathrm{G}(\boldsymbol{x}) = \mathrm{softmax}\left([g_1, g_2, \cdots, g_N]\right)$$
$$\text{where } g_i = \|\boldsymbol{x}\boldsymbol{W}_{down}^i\|,$$

$$E_i(\boldsymbol{x}) = \left(\texttt{SiLU}(\boldsymbol{x}\boldsymbol{W}_{down}^i\boldsymbol{W}_{up}^i) \odot (\boldsymbol{x}\boldsymbol{W}_p^i)\right)\boldsymbol{W}_o^i.$$

While AoE's expert autonomy leads to better model performance than MoE, it introduces computational and memory overhead. The inefficiency arises because all experts compute activations, but only a fraction are used in the output. Therefore, this paper focuses on achieving autonomous selection with an efficiency comparable to MoE.

2.3 Efficiency Dilemma of AoE Factorization

AoE employs factorization of $\boldsymbol{W}_g$ to improve efficiency; however, this design traps into a dilemma: it contends with either substantial computational overhead or excessive memory access. Consequently, the factorization itself becomes the bottleneck to further efficiency-wise advancement.

We conduct a breakdown analysis to elucidate this dilemma theoretically. Specifically, AoE requires additional FLOPs per token that increase linearly with the rank r. Relative to an MoE of the same parameter size, this increase is defined as:

$$\Delta\mathrm{FLOPs} = 2 \cdot d \cdot r \cdot (N - K).$$

In addition to the computational cost, the per-token memory overhead is increased by:

$$\Delta\mathrm{Mem} = \max\left(Nr,\ 4K(D' - D)\right).$$

Details of the numerical derivation are provided in Appendix A. The right panel of Figure 2 characterizes AoE's computational and memory overhead as a function of r. We argue that the efficiency of AoE is bounded by either memory access or computational throughput. Although $r \in (48, 80)$ offers a more favorable trade-off between memory and computation, it results in unstable AoE training for wide models with larger d and D. This dilemma calls for a new realization of autonomous expert selection that is both practical and scalable.

3 Methodology: Union-of-Experts

3.1 Motivation

Through extensive preliminary experiments, we arrive at a surprising finding: a small subset of neurons is sufficient to parameterize the routing function and this capability emerges naturally during language model training. We term these elements as "routing neurons" and provide an analysis in Section 6, to uncover how this expert autonomy emerges. We name the proposed architecture as Union-of-Experts, where the union of routing neurons governs the experts routing. Figure 3 presents an overview of our UoE, with implementation details provided in the following sections.

3.2 Autonomous Expert Routing with Routing Neurons Acceleration

Specifically, we reserve a fixed subset of neurons within each expert for routing. For simplicity, we pre-designate the first N_s neurons of each expert weight matrix as routing neurons. These routing neurons can be formulated as follows:

$$\boldsymbol{W}_*^i = \left[\underbrace{\widetilde{\boldsymbol{W}}_*^i}_{N_s} \middle| \boldsymbol{W}_{*\text{others}}^i \right],$$

where $\boldsymbol{W}_*^i \in \{\boldsymbol{W}_g^i, \boldsymbol{W}_p^i, \boldsymbol{W}_o^{i\top}\}$ denotes the weight matrices of the i-th expert. The tilde notation $\widetilde{\boldsymbol{W}}_*^i$ identifies the routing neurons, as they are a subcomponent of the weight matrix. We also trialed with more flexible neurons selection strategy but found no consistent gain. We therefore just adopt the straightforward setup aforementioned. More empirical trials can be found in Appendix B.

Prior studies (Lv et al., 2025; Geva et al., 2021) suggest that higher activation magnitudes correspond to better input–module alignment. Building on this insight, UoE performs autonomous expert selection for any input $\boldsymbol{x}$ by computing the L^2 norm of the routing neurons. Formally, the routing function G in UoE is refined as:

$$\begin{aligned} \mathrm{G}(\boldsymbol{x}) &= \mathrm{softmax}\left(\mathrm{TopK}\left[g_1, g_2, \cdots, g_N\right]\right), \\ \text{where } g_i &= \|\mathtt{SiLU}(\boldsymbol{x}\widetilde{\boldsymbol{W}_g^i}) \odot (\boldsymbol{x}\widetilde{\boldsymbol{W}_p^i})\|. \end{aligned} \tag{3}$$

Since no factorization is applied, each expert in our model is parameterized identically to that in a vanilla MoE. This allows UoE to share the same forward computation with MoE (Equation 1).

Moreover, UoE dispenses with the routing module by leveraging routing neurons, mirroring the router-less architecture of AoE. We further demonstrate a strong correlation between the activation intensity of these routing neurons and that of the overall expert activation. This awareness of expert capability helps UoE for improved expert routing. Detailed discussion can be found in Section 6.

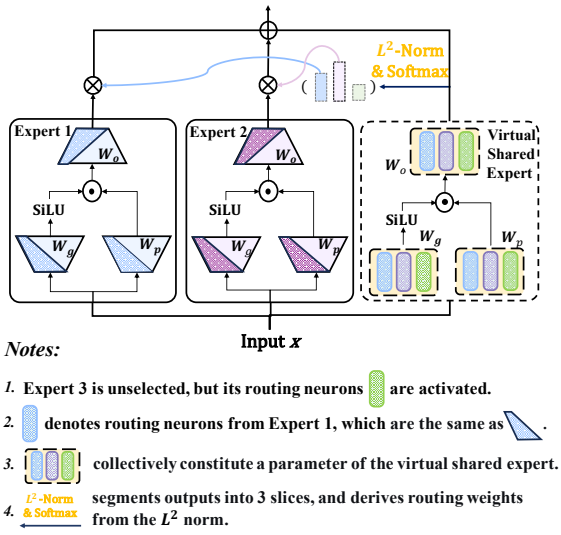


Figure 3: An overview of UoE.

3.3 Virtual Shared Expert Design

In the UoE framework, routing neurons are activated for every input token. While designed for autonomous expert routing, these neurons happen to parallel the pattern of a shared expert design. Given that, we propose a bold idea to allows routing neurons to additionally function as a shared expert. This avoids the waste of precomputation in unselected routing neurons and further improves the computational utilization. Despite being distributed across discrete experts, these neurons constitute a unified block and serve as a shared expert conceptually. Specifically, we set $N_s = \mathrm{round}(D/N)$ routing neurons per expert, matching the parameter count of a shared expert. This abstract shared expert can be formulated as:

$$\boldsymbol{W}_*^s = \left[\underbrace{\widetilde{\boldsymbol{W}}_*^1}_{N_s} \middle| \cdots \middle| \underbrace{\widetilde{\boldsymbol{W}}_*^N}_{N_s} \right],$$

where $\widetilde{\boldsymbol{W}}_*^i \in \{\widetilde{\boldsymbol{W}}_g^i, \widetilde{\boldsymbol{W}}_p^i, \widetilde{\boldsymbol{W}}_o^{i\top}\}$. In the UoE working pipeline, expert-wise routing weights are first computed with individual $\widetilde{\boldsymbol{W}}_g^i, \widetilde{\boldsymbol{W}}_p^i$ parameters, as defined in Equation 3. Subsequently, $\boldsymbol{W}_o^i$ are activated and reduced across all experts, serving as a shared expert output. We also provide a naive Py-

Model	Num.	ARC-E	PIQA	HELLA	SCIQ	WINO	MNLI	QNLI	RTE	AVG.
MoE	8	62.54	68.88	36.74	81.60	52.49	32.78	**51.04**	49.46	54.44
AoE	8	**64.60**	69.59	36.62	**83.30**	51.22	**34.13**	50.01	48.86	54.79
UoE	8	63.09	**69.64**	**37.07**	82.40	**52.88**	33.89	50.05	**51.50**	**55.07**
MoE	4	61.45	67.52	35.27	77.10	50.75	33.25	49.83	46.45	52.70
AoE	4	61.57	68.61	36.07	**82.40**	52.01	33.12	49.80	**50.30**	54.24
UoE	4	**62.25**	**68.66**	**35.67**	81.70	**54.70**	**33.62**	**50.20**	48.98	**54.47**

Table 1: Main validation results on 1B parameter models. We compare performance across different activated experts setups. Colored cells highlight improvements over MoE baseline, while bold values mark the best results.

Torch implementation of our UoE in Appendix C.3 to better illustrate the workflow.

The conceptual shared expert was primarily designed to boost UoE's efficiency via activation reuse. Ablation studies in Section 5 further confirm that it delivers performance comparable to the traditional shared experts. A more comprehensive analysis of this shared expert design is provided in the subsequent sections.

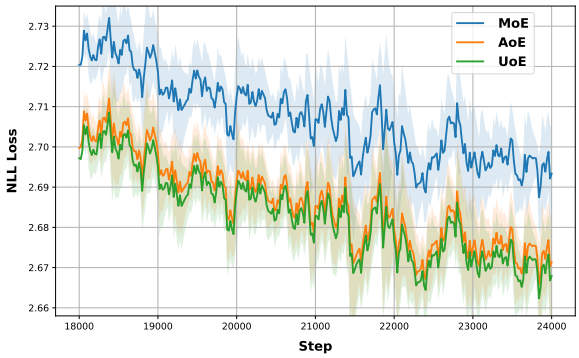


Figure 4: Pre-training loss comparison. UoE exhibits improved convergence over MoE and AoE baselines.

4 Experiment

4.1 Main Results and Analysis

General Setup. We evaluate UoE against two primary baselines: the vanilla MoE and AoE. To verify the effectiveness of UoE, we pre-train language models with 1B parameters from scratch based on the Torchtitan framework (Liang et al., 2025). Our language model consists of 8 Transformer layers with a hidden dimension of $d = 1024$ and an intermediate dimension of $D = 512$. For each Transformer layer, we employ the multi-head attention mechanism with 8 attention heads. We substitute all FNN layers with MoE layers and each MoE layer has $N = 64$ routed experts. The MoE baseline is configured with a shared expert following the setup in (Dai et al., 2024). Please refer to Appendix C for further implementation details.

We pre-train all models with 100B tokens from Fineweb-Edu datasets (Penedo et al., 2024). For optimization, we employ the AdamW optimizer with $(\beta_1, \beta_2) = (0.9.0.95)$, a gradient norm clipping threshold of 1, and weight decay as 0.1. We use a learning rate of 1×10^{-3} with 1000 steps linear warmup, followed by a cosine decay scheduler.

We evaluate these language models across 8 widely used benchmarks, including *ARC* (Clark et al., 2018), *PIQA* (Bisk et al., 2020), *HellaSwag* (Zellers et al., 2019), *SCIQ* (Welbl et al., 2017), *Winogrande* (Sakaguchi et al., 2019), *MNLI* (Wang et al., 2018), *QNLI* (Wang et al., 2018) and *RTE* (Wang et al., 2018). All evaluations are performed using the LM Evaluation Harness (Gao et al., 2024). The first five tasks are evaluated zero-shot. For the remaining three tasks, we report their average performance under 0-shot, 3-shot and 5-shot to reduce randomness.

Experimental Results. Table 1 summarizes the main results for the 1B-parameter models pre-trained under different configurations of active experts. UoE consistently outperforms both MoE and AoE models, demonstrating the effectiveness of the proposed architecture. Notably, the performance advantages of UoE are pronounced at lower expert activation ratio, with more significant improvements observed when activating 4 experts activated compared to 8. This improvement can be attributed to UoE's ability to optimize token-expert matching with the pool of routed experts grows larger. As modern MoE architectures trend towards higher sparsity, UoE demonstrates promise for increasingly sparse configurations. Figure 4 plots the pre-training loss for UoE under this configuration. UoE achieves lower training loss compared to both

	$\text{Ent}^1_{\text{load}}$	$\text{Ent}^2_{\text{load}}$	$\text{Ent}^3_{\text{load}}$	$\text{Ent}^4_{\text{load}}$	$\text{Ent}^5_{\text{load}}$	$\text{Ent}^6_{\text{load}}$	$\text{Ent}^7_{\text{load}}$	$\text{Ent}^8_{\text{load}}$
MoE	3.45	3.23	3.29	3.14	3.57	3.76	**3.66**	**3.42**
UoE	**3.70**	**3.62**	**3.71**	**3.71**	**3.88**	**3.84**	**3.66**	3.31

Table 2: Load Entropy of expert selection. Higher entropy indicates more balanced expert loads.

MoE and AoE, suggesting superior training efficiency. We report the standard errors for Table 9 in Appendix to demonstrate that the improvements of UoE are statistically reliable.

Load Balancing. Prior study shows that AoE achieves better load balancing (Lv et al., 2025). We compare UoE with MoE baseline to investigate whether it achieves enhanced load balancing in the absence of an auxiliary loss. Expert load statistics are examined using a calibration set of 1,000 instances from Wikitext-2 (Merity et al., 2016). Specifically, we take the load entropy Ent_{load} as the metric. For the i-th layer, the load entropy Ent_{load} is defined as:

$$\text{Ent}^i_{load} = -\sum_{i=1}^{N} p_i \log p_i,$$
$$\text{where } p_i = \text{softmax}([g_1, g_2, \cdots, g_N]).$$

where the routing weights g_i for UoE and MoE are given in Equations 2 and 3. Table 2 compares the layer-wise load entropy of MoE baseline and UoE to highlight their differences. Except for the final layer, UoE consistently exhibits lower load entropy, suggesting a more balanced distribution. To provide a more intuitive comparison of load balancing, we also visualize the expert load distribution f_i for UoE alongside MoE in Figure 6. The visualization align with our observation regarding the load entropy. Further analysis and implementation details can be found in Appendix D.

We currently attribute the enhanced load balancing of UoE and AoE to expert autonomy. We regard this as a favorable side effect of expert autonomy and leave it for future work to investigate how this routing mechanism optimizes expert utilization.

4.2 Efficiency Analysis of UoE

In this section, we analyze the efficiency of UoE in comparison with the baseline methods, focusing primarily on (1) training efficiency metrics and (2) the end-to-end inference performance.

Training Efficiency of UoE. We begin by conducting a comparative analysis of the training efficiency of UoE, MoE, and AoE. Table 3 reports the achieved training throughput and peak memory usage across the evaluated methods. UoE achieves a 19.8% improvement in training throughput over AoE while maintaining competitive downstream performance. Notably, UoE also surpasses the MoE baseline in end-to-end training throughput, despite a marginal increase in computation. We attribute this unexpected efficiency gain to the superior load-balancing of UoE, consistent with our earlier analysis in previous section.

Metric	MoE	AoE	UoE
TP. (K/s)	604.00	509.00	610.00
Mem. (GB)	63.93	71.51	63.96

Table 3: Training Throughput and Memory comparison.

We also provide a theoretical analysis of the communication overhead in UoE. We contend that the virtual shared expert design incurs identical communication costs to standard shared experts in MoE architectures. During the training phase, the forward pass requires an All-Gather operation on the routing neurons of the virtual shared expert, which is functionally analogous to the parameter synchronization in a parameter-sharded router module. We leave the empirical validation of this efficiency in large-scale settings for future work.

Configuration		**TP. (token/s) / Mem. (GB)**	
Model	**BS**	**1024**	**4096**
MoE	1	35.84 (2.15)	35.81 (2.15)
UoE		35.98 (2.15)	35.89 (2.15)
MoE	4	141.38 (2.21)	141.37 (2.21)
UoE		141.00 (2.21)	141.46 (2.21)
MoE	16	560.93 (2.48)	559.97 (2.48)
UoE		551.39 (2.47)	551.93 (2.47)

Table 4: TP. and Mem. comparisons at inference time.

	Config	ARC-E	PIQA	HELLA	SCIQ	WINO	MNLI	QNLI	RTE	AVG.
⓪	UoE	63.09	69.64	37.07	82.40	52.88	33.89	50.05	51.50	55.07
		Virtual shared expert variants								
①	Ablation	53.83	66.21	33.80	75.80	50.36	33.93	50.27	51.50	51.96
②	Ablation	62.42	69.48	37.16	81.90	52.09	33.76	49.86	51.74	54.80
③	Ablation	64.56	69.10	36.78	81.70	53.67	33.93	50.78	50.82	55.04
		Expert selection variants								
④	$\mathbf{x}\mathbf{W}_p$	63.72	70.08	36.69	82.50	51.70	32.95	50.00	50.18	54.73
⑤	$\mathbf{x}\mathbf{W}_g$	63.97	69.21	37.25	80.70	52.09	33.62	51.06	49.58	54.69
⑥	$\texttt{SiLU}(\mathbf{x}\mathbf{W}_g)$	63.51	69.48	36.76	82.40	53.35	33.73	49.58	49.22	54.75
		Routing neurons selection variants								
⑦	double N_s	63.72	68.28	36.58	84.20	51.30	34.11	49.97	50.66	54.85

Table 5: Ablation studies of UoE design variants.

End-to-end Generation Comparison. To evaluate the inference efficiency of UoE we develop an evaluation pipeline built on the Huggingface GenerationMixin library (Wolf et al., 2020). We conduct a comprehensive breakdown of efficiency by benchmarking both generation throughput and peak memory footprints. The results in Table 4 confirm that UoE incurs a computational overhead nearly identical to that of a standard MoE.

5 Ablation Studies

We perform ablation studies on our pre-trained UoE to evaluate the effectiveness of several designs.

Ablation of the Virtual Shared Expert. We devise ablation studies to validate the effectiveness of virtual shared expert. Three variants are designed to characterize the shared expert's contribution:
① **Abl.** We deactivate the virtual shared expert during inference and use only the routed experts.
② **Abl.** We restrict routing neurons to simply perform expert routing and pre-train it from scratch.
③ **Abl.** We extend ② with an additional standard shared expert and pre-train UoE from scratch.
Table 5 presents the ablation results, from which we draw the following observations:

(1) The virtual shared expert is critical to overall model performance and behaves similarly to the standard shared expert design;

(2) The virtual shared expert indeed acquires abilities that are not presented in the routed experts;

(3) The reuse of routing neurons attains performance comparable to a conventional shared expert and reduce the overhead.

Taken together, the virtual shared expert design improves activation utilization and provides an efficient alternative to the standard shared expert.

Ablation of Expert Selection Criteria. By default, we employ the activation intensity of experts for routing (Equation 3). For Configuration ④ to ⑥, we explore using alternative nodes in the computation graph as routing criteria. While the training overhead for these variants is nearly identical, they underperform the default setup ($\texttt{SiLU}(\mathbf{x}\mathbf{W}_g) \odot \mathbf{x}\mathbf{W}_p$). Consequently, we adopt activation intensity as our default routing criteria.

Ablation on the Selection of N_s. We conduct an ablation study on the number of routing neurons, N_s. Specifically, we double the value of N_s to examine its impact. The results in Table 5 indicate no performance gain and even a slight degradation. We do not explore larger settings further, as they would lead to a disproportionately high number of shared experts, compromising model efficiency.

6 Toward a Mechanistic Understanding of Routing Neurons

In this section, we present a post-hoc analysis to elucidate the mechanistic emergence of expert autonomy within the routing neurons. Our spectral analysis of model parameters uncovers an intriguing property of these neurons, which provides a preliminary understanding of the observed phenomenon. We hope these findings provide insights into the inner workings of UoE and inspire the community to design more powerful MoE models that leverage expert autonomy.

Model	Act.	ARC-E	PIQA	HELLA	SCIQ	WINO	MNLI	QNLI	RTE	AVG.
UoE	Neurons	63.09	69.64	37.07	82.40	52.88	33.89	50.05	51.50	55.07
	Expert	61.49	68.72	36.49	82.10	51.54	34.25	50.16	50.30	54.38

Table 6: Performance change when using experts' activation intensity instead of routing neurons.

Routing neurons function as proxies for global expert activation. Our analysis is motivated by a key insight: routing neurons act as proxies that closely track the overall activation profiles of individual experts. To validate this, we conduct a pilot experiment on pretrained UoE model, wherein the top-K experts are selected with the highest activation intensities of experts rather than routing neurons. Table 6 presents the results, showing only a marginal performance drop when the original activations are substituted. This provides empirical evidence of the functional alignment between routing neurons and their corresponding expert activations.

Routing neurons converge toward the principal singular vector of the expert parameters. We first derive a mathematical derivation in Appendix E, showing that the principal right singular vectors dominate the activation intensity of any input $\boldsymbol{x}$ under the matrix $\boldsymbol{W}_*$. Given that, we plot the similarity between the principle right singular vectors of $\boldsymbol{W}_*$ and $\widetilde{\boldsymbol{W}_*}$. Figure 5 visualize the results, for routing neurons of the j-th expert at layer i, the similarity term $S_*^{i,j}$ is given by:

$$S_*^{i,j} = \texttt{L}^2\texttt{-Norm}\left(< \boldsymbol{U}_*^r, \boldsymbol{U}_*[0] >^{i,j}\right),$$

the measure for router weights is defined as:

$$S_r^{i,j}{}_* = \texttt{L}^2\texttt{-Norm}\left(< \boldsymbol{R}^{i,j}, \boldsymbol{U}_*[0] >^{i,j}\right).$$

Appendix E provide further details on the definitions. As shown in Figure 5, the principal singular vectors exhibit a marked similarity, whereas no such correspondence is observed in the MoE models. While not entirely formal, this provides a heuristic mechanistic explanation for how routing neurons emerge as proxies for expert activations. We tentatively attribute this phenomenon to the use of expert-autonomous routing in UoE. We hope these findings provide insights to the community and catalyze further research into MoE routing.

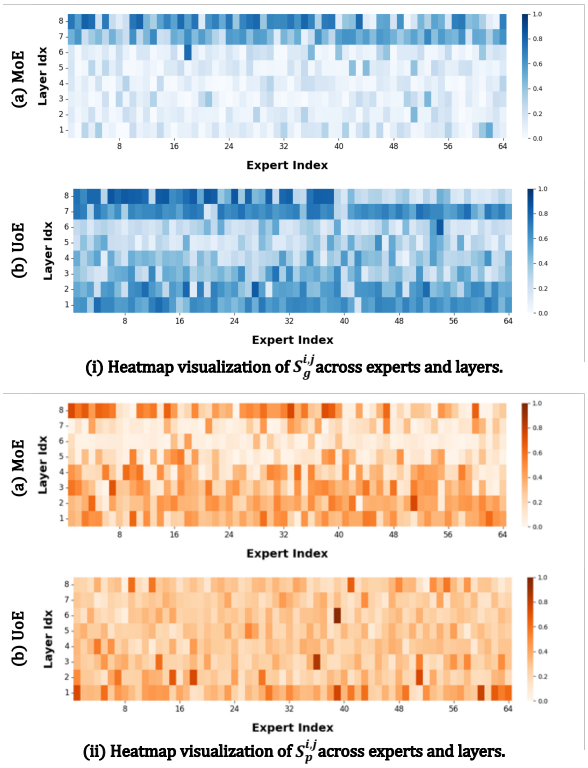


Figure 5: Heatmap visualization of $S_g^{i,j}$ and $S_p^{i,j}$.

7 Scalability Analysis: Scaling Potential of UoE to Larger Model Scale

To evaluate the scaling potential of UoE, we increase the model capacity by pre-training both UoE and baselines at a 3B-parameter scale. The pre-training protocol follows most of the configuration used for our 1B-parameter models. Detailed hyper-parameters regarding the model architectures are provided in Appendix C.1. As the model capacity increases, we expand evaluation to benchmarks more challenging than those for 1B models, including *SocialIQa* (Sap et al., 2019), *OpenbookQA* (Mihaylov et al., 2018), *CommonsenseQA* (Talmor et al., 2019) and *MMLU* (Hendrycks et al., 2021). We exclude the GLUE tasks (*MNLI*, *QNLI* and *RTE*) due to inconsistent performance scaling relative to model size.

Table 7 presents the comparative results. UoE

	ARC-E	ARC-C	PIQA	HELLA	SCIQ	WINO	SIQA	OBQA	CSQA	MMLU	AVG.
MoE	63.64	31.48	70.62	39.52	**89.40**	51.22	41.70	31.00	34.97	27.21	50.39
AoE	64.44	31.57	70.24	40.34	88.80	**53.35**	43.24	33.60	**38.17**	30.47	51.53
UoE	**69.07**	**33.11**	**73.18**	**41.96**	87.10	52.80	**43.40**	**34.20**	37.76	**30.54**	**52.51**

Table 7: Downstream performance comparison for 3B-parameter pretrained models. Colored entries highlight improvements of UoE over the MoE baseline, while bold text indicates the best results across all models.

consistently outperforms both MoE and AoE baselines at larger parameter scales, with the performance gap widening as model capacity increases. This highlights the scaling potential of the UoE, indicating that UoE can effectively leverage increased model size to achieve superior performance. We hope to empirically validate the effectiveness of UoE at a larger scale, given the availability of additional computational resources

8 Related Work

Expert Autonomy. Lv et al. (2025) first identified the separation between router decision and expert execution. To address the issue, they propose Autonomy-of-Experts, a new MoE paradigm as solution. In the AoE framework, experts compute activations for each input and perform token routing with their activation norms. While promising, this architectural design is constrained by an efficiency dilemma and faces scalability challenges as the number of experts increases. UoE addresses the efficiency bottlenecks of AoE by reformulating its routing paradigm with neurons. A virtual shared expert design is further proposed to exploit activation utilization and improve efficiency.

Few studies have explored alternative paradigms to achieve expert autonomy. Pham et al. (2024) uses the norms of experts' final output as distillation labels to supervise router logits. This requires computing the outputs of all experts during training, which is computationally prohibitive. A more recent study (Lv et al., 2026) devise an auxiliary loss to couple expert and router, thereby enabling expert autonomy. This work provides valuable insights, encouraging researchers to rethink expert autonomy from router-centric rather than token-centric perspective. We hope our work, alongside prior studies, will inspire further studies into this long-overlooked challenge in MoE routing.

Shared Expert. Modern MoE models face challenges regarding expert redundancy and functional overlap. To address the issues, Dai et al. (2024) introduce a shared expert design to capture general knowledge and facilitate expert specialization. In this paper, we propose a virtual shared expert design to reuse the activation of routing neurons. While this was primarily designed to improve computational efficiency, we surprisingly find that it perform competitively with a vanilla shared expert in empirical evaluations. We hope to uncover deeper mechanistic insights into the dual-role nature of these neurons in future work.

9 Conclusion

We introduce UoE, an MoE variant that performs expert autonomy routing. UoE pre-designates a specific subset of neurons for each expert and utilizes their activation norms to guide token routing. To minimize pre-computation overhead from unselected routing neurons, we propose a virtual shared expert design. Furthermore, we analyze several properties of UoE and provide a preliminary explanation for the emergent expert autonomy. We hope our findings inform future MoE routing designs.

Acknowledgement

This work was supported by The National Natural Science Foundation of China (No. 62376273 and No. U2436209). This work was conducted during Songhao Wu's internship at Tencent as part of the CCF-Tencent Rhino-Bird Elite Talent Program. Ruobing Xie is supported by the Young Elite Scientists Sponsorship Program by CAST (2023QNRC001). We are grateful to Yuxuan Liu for his proofreading and helpful writing suggestions. Additionally, we sincerely acknowledge the anonymous ARR reviewers for their constructive comments and questions, which have greatly improved this work.

Limitations

Due to current hardware constraints, this work is restricted to specific model sizes. We aim to evaluate UoE at a broader scale in future work to systematically observe its scaling behavior and performance under expanded computational budgets. Further, a practical limitation of UoE is the requirement for from-scratch pre-training. The lack of a direct conversion path from vanilla MoE checkpoints to UoE remains a constraint; developing such a transition is still a promising direction for future work.

References

Yonatan Bisk, Rowan Zellers, Jianfeng Gao, Yejin Choi, and 1 others. 2020. Piqa: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 7432–7439.

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*.

Damai Dai, Chengqi Deng, Chenggang Zhao, R. X. Xu, Huazuo Gao, Deli Chen, Jiashi Li, Wangding Zeng, Xingkai Yu, Y. Wu, Zhenda Xie, Y. K. Li, Panpan Huang, Fuli Luo, Chong Ruan, Zhifang Sui, and Wenfeng Liang. 2024. Deepseekmoe: Towards ultimate expert specialization in mixture-of-experts language models. *Preprint*, arXiv:2401.06066.

DeepSeek-AI, Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Daya Guo, Dejian Yang, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, and 181 others. 2025. Deepseek-v3 technical report. *Preprint*, arXiv:2412.19437.

Trevor Gale, Deepak Narayanan, Cliff Young, and Matei Zaharia. 2022. Megablocks: Efficient sparse training with mixture-of-experts. *Preprint*, arXiv:2211.15841.

Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, and 5 others. 2024. The language model evaluation harness.

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

Hao Gu, Wei Li, Lujun Li, Qiyuan Zhu, Mark Lee, Shengjie Sun, Wei Xue, and Yike Guo. 2025. Delta decompression for moe-based llms compression. *Preprint*, arXiv:2502.17298.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021. Measuring massive multitask language understanding. *Preprint*, arXiv:2009.03300.

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, Gianna Lengyel, Guillaume Bour, Guillaume Lample, Lélio Renard Lavaud, Lucile Saulnier, Marie-Anne Lachaux, Pierre Stock, Sandeep Subramanian, Sophia Yang, and 7 others. 2024. Mixtral of experts. *Preprint*, arXiv:2401.04088.

Wanchao Liang, Tianyu Liu, Less Wright, Will Constable, Andrew Gu, Chien-Chin Huang, Iris Zhang, Wei Feng, Howard Huang, Junjie Wang, Sanket Purandare, Gokul Nadathur, and Stratos Idreos. 2025. Torchtitan: One-stop pytorch native solution for production ready LLM pretraining. In *The Thirteenth International Conference on Learning Representations*.

Ang Lv, Jin Ma, Yiyuan Ma, and Siyuan Qiao. 2026. Coupling experts and routers in mixture-of-experts via an auxiliary loss. In *The Fourteenth International Conference on Learning Representations*.

Ang Lv, Ruobing Xie, Yining Qian, Songhao Wu, Xingwu Sun, Zhanhui Kang, Di Wang, and Rui Yan. 2025. Autonomy-of-experts models. *arXiv preprint arXiv:2501.13074*.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2016. Pointer sentinel mixture models. *Preprint*, arXiv:1609.07843.

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. In *EMNLP*.

Niklas Muennighoff, Luca Soldaini, Dirk Groeneveld, Kyle Lo, Jacob Morrison, Sewon Min, Weijia Shi, Evan Pete Walsh, Oyvind Tafjord, Nathan Lambert, Yuling Gu, Shane Arora, Akshita Bhagia, Dustin Schwenk, David Wadden, Alexander Wettig, Binyuan Hui, Tim Dettmers, Douwe Kiela, and 5 others. 2025. OLMoe: Open mixture-of-experts language models. In *The Thirteenth International Conference on Learning Representations*.

OpenAI. 2025. gpt-oss-120b & gpt-oss-20b model card. *Preprint*, arXiv:2508.10925.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward

Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, and 2 others. 2019. Pytorch: An imperative style, high-performance deep learning library. *Preprint*, arXiv:1912.01703.

Guilherme Penedo, Hynek Kydlíček, Anton Lozhkov, Margaret Mitchell, Colin A Raffel, Leandro Von Werra, Thomas Wolf, and 1 others. 2024. The fineweb datasets: Decanting the web for the finest text data at scale. *Advances in Neural Information Processing Systems*, 37:30811–30849.

Quang Pham, Giang Do, Huy Nguyen, TrungTin Nguyen, Chenghao Liu, Mina Sartipi, Binh T. Nguyen, Savitha Ramasamy, Xiaoli Li, Steven Hoi, and Nhat Ho. 2024. Competesmoe – effective training of sparse mixture of experts via competition. *Preprint*, arXiv:2402.02526.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2019. Winogrande: An adversarial winograd schema challenge at scale. *arXiv preprint arXiv:1907.10641*.

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

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

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. 2019. CommonsenseQA: A question answering challenge targeting commonsense knowledge. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4149–4158, Minneapolis, Minnesota. Association for Computational Linguistics.

Kimi Team, Yifan Bai, Yiping Bao, Guanduo Chen, Jiahao Chen, Ningxin Chen, Ruijue Chen, Yanru Chen, Yuankun Chen, Yutian Chen, and 1 others. 2025. Kimi k2: Open agentic intelligence. *arXiv preprint arXiv:2507.20534*.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2018. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 353–355, Brussels, Belgium. Association for Computational Linguistics.

Johannes Welbl, Nelson F. Liu, and Matt Gardner. 2017. Crowdsourcing multiple choice science questions. In *Proceedings of the 3rd Workshop on Noisy User-generated Text*, pages 94–106, Copenhagen, Denmark. Association for Computational Linguistics.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, and 3 others. 2020. Huggingface's transformers: State-of-the-art natural language processing. *Preprint*, arXiv:1910.03771.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. HellaSwag: Can a machine really finish your sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4791–4800, Florence, Italy. Association for Computational Linguistics.

Yanli Zhao, Andrew Gu, Rohan Varma, Liang Luo, Chien-Chin Huang, Min Xu, Less Wright, Hamid Shojanazeri, Myle Ott, Sam Shleifer, Alban Desmaison, Can Balioglu, Pritam Damania, Bernard Nguyen, Geeta Chauhan, Yuchen Hao, Ajit Mathews, and Shen Li. 2023. Pytorch fsdp: Experiences on scaling fully sharded data parallel. *Preprint*, arXiv:2304.11277.

A Breakdown of the Computational and Memory Overhead

We present the mathematical derivations for the computational overhead and memory consumption for both AoE and MoE as functions of the rank r. For a vanilla MoE layer, the total FLOPs involves three matrix multiplication of $\boldsymbol{W}_g, \boldsymbol{W}_p$ and $\boldsymbol{W}_o$:

$$\mathrm{FLOPs}_{MoE} = 3 \cdot \mathrm{TK}\left(2\,\mathrm{D} \cdot \mathrm{d}\right),$$

where T is the sequence length and the routing cost is omitted for simplification. The FLOPs for an AoE layer comprises two parts: (i) $\mathbf{W}_{up}$ and $\mathbf{W}_o$ for the k selected experts; (ii) $\mathbf{W}_{down}$ across all N experts, and the total FLOPs are formualted as:

$$\begin{aligned}\mathrm{FLOPs}_{AoE} = \quad & 2 \cdot \mathrm{TK}\left(2\,\mathrm{D}' \cdot \mathrm{d}\right) + \mathrm{TK}\left(2\,\mathrm{D}' \cdot \mathrm{r}\right) \\ & + \mathrm{TN}\left(2\,\mathrm{d} \cdot \mathrm{r}\right),\end{aligned}$$

where D' is the FFN hidden size of AoE to ensure the same number of parameters:

$$D' = \frac{3 \cdot \mathrm{D} \cdot \mathrm{d} - \mathrm{d} \cdot \mathrm{r}}{\mathrm{r} + 2 \cdot \mathrm{d}}.$$

We substitute D' and complete the reduction. Compared with MoE, the change of FLOPs is:

$$\Delta\mathrm{FLOPs} = 2\,\mathrm{T} \cdot \mathrm{d} \cdot \mathrm{r} \cdot (N - \mathrm{K}). \quad \square$$

We further analyze the memory overhead of UoE compared to vanilla MoE. UoE incurs additional memory usage in two stages:

1. The computation of all W_{down} necessitates a memory increase proportional to $T \cdot Nr$;
2. The increased D' leads to a larger intermediate hidden size, expand the experts' activation cache by a factor of $T \cdot 4K(D' - D)$.

Given that, the change of memory overhead is:

$$\Delta\mathrm{Mem} = \max\left(T \cdot Nr, T \cdot 4K\left(D' - D\right)\right). \quad \square$$

B Exploratory Trials on Routing Neuron Selection within Experts

This section describes our early explorations and provides the rationale for assigning routing neurons to the initial N_s positions. Our original goal was to isolate key neurons per expert that reflect the global activation pattern. We diverged from this line of exploration for two reasons:

(1) we have to iterate over the expert matrices to identify neurons, leading to inefficient training;

(2) the initial implementations of dynamic selection were unable to match the performance of our straightforward methods.

Given that, we designate a subset of neurons to serve as routing function. As these routing neurons are trained from scratch, selecting different neurons from the expert weights would only change the weight initialization. Since these routing neurons are trained from scratch, selecting different indices from the expert weights would merely result in a different weight initialization. To streamline the implementation, we simply assign the first contiguous N_s neurons to perform expert routing.

C Implementation Details of UoE

C.1 Hyper-parameters of Model Architecture

Table 8 presents details on the architecture hyper-parameters used across our experiments.

Hyper-Parameters	1B	3B
hidden size	1024	1280
MoE layers	8	20
FFN hidden size	512	512
attention heads	8	20
key-value heads	8	20
routed experts	64	64
vocab size	128,256	128,256
RoPE theta	500,000	500,000

Table 8: Hyper-parameters of model architecture.

C.2 Training setup details for UoE

We provide supplementary details regarding the training framework for UoE. The training pipeline is built upon TorchTitan framework (Liang et al., 2025). We employ PyTorch's native FSDP (Fully Sharded Data Parallel) for distributed training (Zhao et al., 2023), and all models are trained on NVIDIA A800-SXM4-80GB GPUs. Specifically, we use PyTorch's `scaled_dot_product_attention` for attention (Paszke et al., 2019), and MegaBlocks MLP for efficient MoE implementation (Gale et al., 2022). More details can be found in our code.

C.3 Pseudo Code Implementation

Figure 7 presents a naive PyTorch implementation of UoE. At inference time, we repack the routing neurons into layout of the MoE shared expert. This

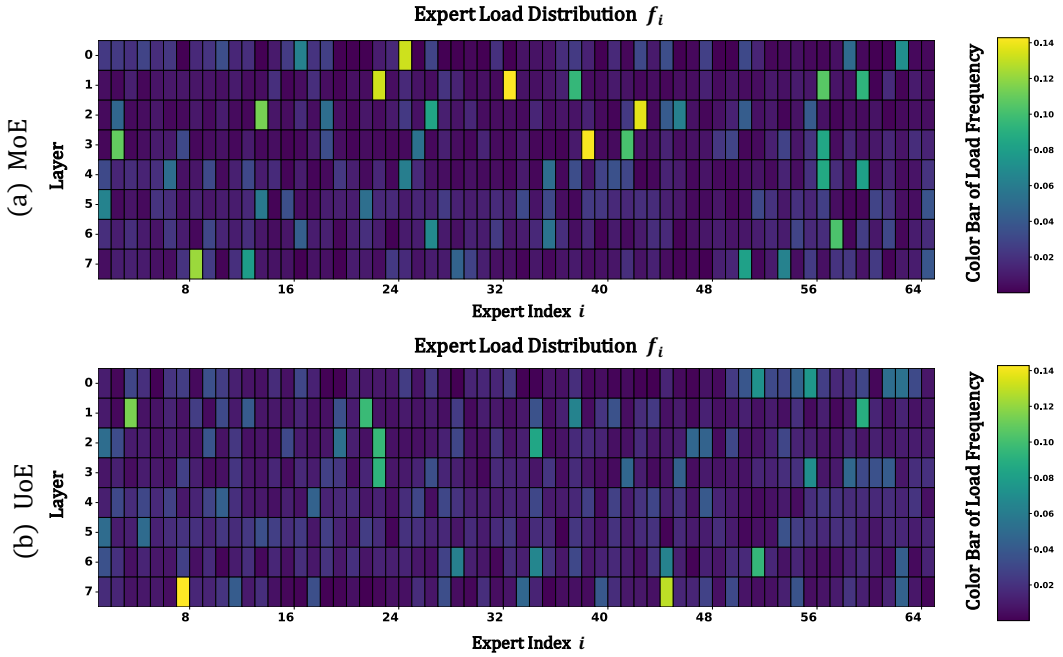


Figure 6: Expert Loading Distribution of UoE and MoE.

prevents non-contiguous parameter and improves UoE's compatibility with MoE deployments.

D Expert Loading Distribution of UoE

Figure 6 visualizes the expert loading for our pre-trained UoE and MoE. The load distribution f_i for the i-th expert on a batch of T tokens is defined as:

$$\mathbf{f}_i = \frac{1}{T} \sum_{\mathbf{x} \in \mathcal{B}} \mathbb{1} \left\{ i \in \texttt{argTopk} \left(G \left(\mathbf{x} \right) \right) \right\}.$$

Except for the final layer, UoE achieves consistently better load balance, with far fewer cases of the imbalance observed in the shallow layers of MoE. These results are in agreement with the observations in Table 2, offering additional support for the claim that UoE improves load balance.

E Protocol for Interpretability Studies

This section provides a theoretical explanation for how routing neurons can reflect expert activation. Specifically, we measure the activation of input $\boldsymbol{x}$ at weight matrices $\boldsymbol{W}_*$ using the $\texttt{L}^2\texttt{-Norm}$:

$$\texttt{L}^2\texttt{-Norm} \left(\boldsymbol{x} \boldsymbol{W}_* \right) = \sqrt{\boldsymbol{x} \boldsymbol{W}_* \boldsymbol{W}_*^\top \boldsymbol{x}^\top},$$

where $\boldsymbol{W}_* \in \{\boldsymbol{W}_g, \boldsymbol{W}_p\}$. Given the **SVD** of $\boldsymbol{W}_*$, we can expand the equation into:

$$\texttt{L}^2\texttt{-Norm} \left(\boldsymbol{x} \boldsymbol{W}_* \right) = \sqrt{\boldsymbol{x} \boldsymbol{U}_* {\boldsymbol{\Sigma}_*}^2 \boldsymbol{U}_*^\top \boldsymbol{x}^\top},$$
$$\texttt{where } \boldsymbol{W}_* = \boldsymbol{U}_* \, \boldsymbol{\Sigma}_* \, \boldsymbol{V}_*^\top.$$

In a similar manner, the activation magnitude of the input token $\boldsymbol{x}$ relative to $\widetilde{\boldsymbol{W}_*}$ is defined as:

$$\texttt{L}^2\texttt{-Norm} \left(\boldsymbol{x} \widetilde{\boldsymbol{W}_*} \right) = \sqrt{\boldsymbol{x} \boldsymbol{U}_*^r \boldsymbol{\Sigma}_*^{r\,2} \boldsymbol{U}_*^{r\top} \boldsymbol{x}^\top},$$
$$\texttt{where } \widetilde{\boldsymbol{W}_*} = \boldsymbol{U}_*^r \, \boldsymbol{\Sigma}_*^r \, \boldsymbol{V}_*^{r\top}.$$

As the expert weights in MoE are intrinsically low-rank (Lv et al., 2025; Gu et al., 2025), the $\texttt{L}^2\texttt{-Norm}(\boldsymbol{x}\boldsymbol{W}_*)$ is dominated by a small portion of the principle singular vectors. Specifically, we compute the similarity between $\boldsymbol{U}_*^r$ and $\boldsymbol{U}_*[0]$, the principal singular vectors of $\boldsymbol{W}_*$ with the largest singular value across experts. Figure 5 visualizes the results, where the similarity term $S_*^{i,j}$ of the j-th expert at layer i is defined as:

$$S_*^{i,j} = \texttt{L}^2\texttt{-Norm} \left(< \boldsymbol{U}_*^r, \boldsymbol{U}_*[0] >^{i,j} \right).$$

We also visualize the similarity term between router weight $\boldsymbol{R}^{i,j}$ and $\boldsymbol{U}_*[0]$ for comparison:

$$S_r^{i,j} * = \texttt{L}^2\texttt{-Norm} \left(< \boldsymbol{R}^{i,j}, \boldsymbol{U}_*[0] >^{i,j} \right).$$

Notably, the principal singular vectors of the routing neurons in UoE exhibit a higher degree of alignment with the expert weight matrices. Similar phenomenon is not observed in baseline MoEs. This offers initial insights into how expert autonomy emerges, as seen through a weight-analysis lens.

Model	Num.	ARC-E	PIQA	HELLA	SCIQ	WINO	MNLI	QNLI	RTE
MoE	8	0.01	0.01	0.004	0.01	0.01	0.004	0.006	0.03
AoE	8	0.01	0.01	0.005	0.01	0.01	0.005	0.007	0.03
UoE	8	0.01	0.01	0.005	0.01	0.01	0.005	0.007	0.03
MoE	4	0.01	0.01	0.005	0.01	0.01	0.005	0.007	0.03
AoE	4	0.01	0.01	0.005	0.01	0.01	0.005	0.007	0.03
UoE	4	0.01	0.01	0.005	0.01	0.01	0.005	0.007	0.03

Table 9: Standard errors for the results reported in Table 1.

```python
class MoE(nn.Module):
    def __init__(self, args):
        super().__init__()
        self.N, self.d, self.D, self.N_s, self.K = args.N, args.d, args.D, args.N_s, args.K

        assert self.N * self.N_s == self.D # Section 3.3

        # experts.wg / wp / wo: nn.Parameter (shape: (N, D, d))
        self.experts = ParallelMLP(args)
        # shared_expert.wg / wp / wo: nn.Parameter (shape: (D, d))
        self.shared_expert = MLP(args)

    def create_virutal_shared_expert_params(self):
        for param_name in ["wg", "wp", "wo"]:
            src_param = getattr(self.experts,  param_name)
            tgt_param = getattr(self.shared_expert, param_name)

            reshaped_param = src_param[:, :self.N_s, :].reshape(self.D, self.d)
            tgt_param = reshaped_param

    def shared_experts_forward(self, x):
        expert_acts = F.silu(self.shared_expert.wg @ x) * (self.shared_expert.wp @ x)
        out = self.shared_expert.wo.transpose(-1,-2) @ expert_acts

        routing_logits = torch.norm(expert_acts, p=2, dim=-1) # Eq.3
        expert_weights, top_experts = torch.topk(routing_logits, k=self.K, dim=-1)
        expert_weights = expert_weights.softmax(-1)

        return out, expert_weights, top_experts

    def forward(self, x):
        self.create_virutal_shared_expert_params()

        out, expert_weights, top_experts = self.shared_experts_forward(x)

        return out + self.experts(x, expert_weights, top_experts)
```

Figure 7: Pseudo code for UoE implementation in PyTorch.