GMoE: Global Mixture of Experts with Logit Propagation

Geonwoo Hong[1] **Taehwan Kim**[1]*
[1]Ulsan National Institute of Science and Technology
{redgil77, taehwankim}@unist.ac.kr

Abstract

Sparse Mixture of Experts (SMoE) architectures reduce computational cost by activating only a subset of experts per token, yet they often retain large memory footprints and exhibit significant redundancy, both within and across layers. We propose GMoE, a sparse MoE architecture designed to explicitly address these inefficiencies. Instead of maintaining separate expert sets for each layer, GMoE uses Global Experts shared across all layers and adds a Local Expert per layer for layer-specific adaptation. This architecture reuses Global Experts across layers, thereby mitigating inter-layer redundancy while substantially reducing model parameters. In addition, we introduce a Global Router with a GRU-based recurrent component shared across layers and layer-specific routing heads that propagate routing logits across layers. This routing mechanism couples routing decisions across layers, progressively refines routing paths, and helps mitigate intra-layer redundancy. Across diverse language modeling corpora and downstream benchmarks, GMoE remains competitive while using substantially fewer parameters. Routing path analyses and an ablation study show that GMoE reduces cross-layer routing concentration and increases path diversity, with the Global Experts, the Local Expert, and the Global Router all contributing to the gains. The code is available at https://github.com/GEONWOOHONG/GMoE.

1 Introduction

In the era of large language models (LLMs), scaling model parameters and training data in accordance with scaling laws has yielded remarkable performance improvements (Kaplan et al., 2020; Hoffmann et al., 2022). However, as dense Transformers scale, both training and inference costs increase substantially. To address these issues, Sparse Mixture of Experts (SMoE) architectures activate only a subset of experts per token, thereby reducing computational cost while maintaining strong performance (Shazeer et al., 2017; Lepikhin et al., 2021; Fedus et al., 2022).

In a conventional MoE architecture, a single feed-forward network (FFN) in each Transformer block is replaced with multiple experts, and a router activates only a subset per token. An MoE with N experts across L layers therefore admits a combinatorial routing space of up to N^L possible routing paths. However, recent studies suggest that only a limited subset of these routing paths may be exercised in practice (Cai et al., 2024; Yi et al., 2025). Prior work has shown that MoE layers often suffer from intra-layer redundancy, where independently parameterized experts within the same layer may fail to sufficiently differentiate in practice (Chi et al., 2022; Do et al., 2025; Dai et al., 2022). In parallel, dense Transformers are known to exhibit substantial inter-layer redundancy, where FFN blocks learn repeated or near-duplicate transformations in different layers (Dalvi et al., 2020; Dai et al., 2020; Pires et al., 2023). We hypothesize that, because conventional MoE architectures instantiate separate expert sets at every layer, they are susceptible to this behavior, potentially limiting the diversity of realized computations across layers. As a result, increasing the number of experts and layers may primarily increase model parameters rather than effective computational diversity.

Our analysis in Section 5 reveals that these two forms of redundancy—intra-layer and inter-layer—interact in conventional MoE architectures. Independent routers can produce highly correlated and concentrated routing behaviors across layers (Cai et al., 2024), which shrink the effective routing capacity far below the theoretical N^L space. We use the term *compounded redundancy* to denote the coexistence of intra-layer and inter-layer redundancy, where the model parameters disproportionately ex-

*Corresponding author

Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 44599–44614

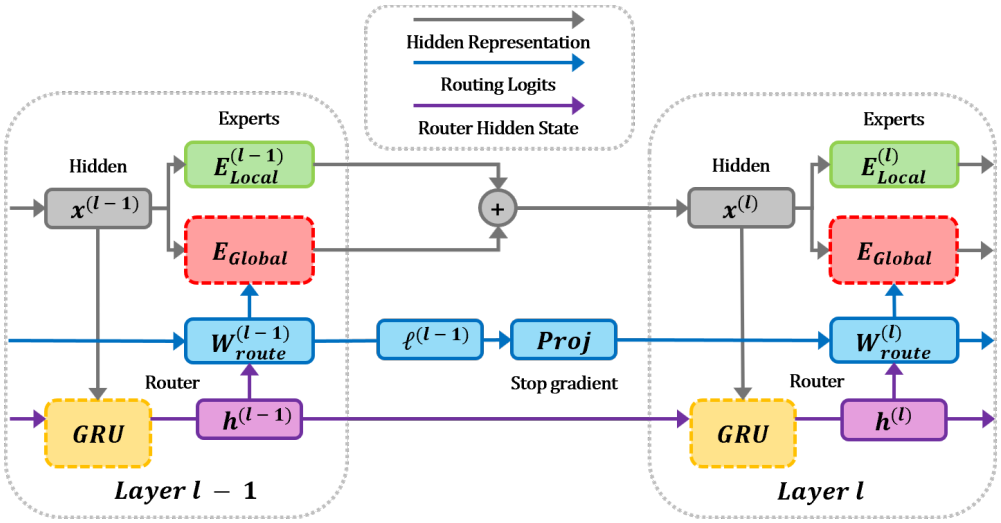

Figure 1: Overview of the proposed GMoE architecture. At each layer l, the Transformer hidden representation $\mathbf{x}^{(l)}$ is processed by a layer-specific Local Expert, while a Global Router produces routing over Global Experts shared across layers. The router, whose GRU-based recurrent component is shared across layers, updates the router hidden state $\mathbf{h}^{(l)}$ from $\mathbf{x}^{(l)}$ and the previous router hidden state $\mathbf{h}^{(l-1)}$. A layer-specific routing head then forms routing logits $\boldsymbol{\ell}^{(l)}$ by combining $\mathbf{h}^{(l)}$ with a stop-gradient projection of the previous layer's routing logits $\mathrm{sg}(\tilde{\boldsymbol{\ell}}^{(l-1)})$. Tokens are dispatched via top-1 routing to a Global Expert, and its output is added to the always-on Local Expert.

ceed its effective capacity—the diversity of expert computations the trained model actually executes. We quantify this effective capacity using path-level routing statistics. In particular, we define *path collapse* as a regime in which the empirical distribution over cross-layer routing paths is highly concentrated, so that most tokens are routed through only a small set of repeated expert sequences. This concentrates traffic on a few experts and leaves much of the combinatorial routing space underutilized, thereby exacerbating compounded redundancy.

To address these issues, we propose Global Mixture of Experts (GMoE), which replaces independent per-layer expert sets with Global Experts shared across all layers while retaining a Local Expert per layer for layer-specific adaptation. The Local Expert is conceptually inspired by shared expert mechanisms in prior work (Dai et al., 2024; Yang et al., 2024). By using Global Experts, GMoE reuses them across layers, thereby reducing inter-layer redundancy while significantly decreasing model parameters.

This cross-layer reuse of Global Experts also enables a new routing mechanism, which we term *logit propagation*. To further reduce intra-layer redundancy, we introduce a Global Router whose GRU-based recurrent component (Cho et al., 2014) is shared across layers, while the routing heads remain layer-specific and propagate routing logits across layers. At each layer l, the routing head takes the current router hidden state $\mathbf{h}^{(l)}$ and a stop-gradient version of the previous layer's routing logits to produce routing logits over the Global Experts. These logits are passed to layer $l{+}1$, enabling progressive logit propagation. Because Global Experts are reused across layers, logit propagation is well-defined: logits from one layer can be directly propagated to the next within the same expert-index space. In contrast, conventional MoE architectures with independent per-layer expert sets do not provide a natural way to propagate logits across layers, since expert indices no longer align. This new routing mechanism encourages progressive refinement of routing paths, thereby alleviating path collapse and reducing intra-layer redundancy.

Our contributions are summarized as follows:

(1) We identify compounded redundancy—both inter-layer and intra-layer—in MoE architectures, and connect it to a concrete routing phenomenon we call path collapse.

(2) We propose GMoE, which integrates Global Experts shared across layers, a Local Expert per layer, and a Global Router with a shared GRU-based recurrent component and layer-specific routing heads.

(3) We demonstrate, through experiments and analyses, that GMoE mitigates compounded redundancy, yielding strong parameter efficiency while maintaining competitive performance.

2 Related Work

In this section, we first review prior MoE architectures that aim to improve expert utilization. Representative examples include Expert Choice Routing (Zhou et al., 2022), which reverses the conventional token-to-expert routing by letting each expert select its own top tokens, yielding strict load balancing and higher training efficiency without auxiliary balancing losses. HyperMoE (Zhao et al., 2024) generates HyperExperts via shared hypernetworks that transfer knowledge from unselected to selected experts, thereby enhancing specialization while preserving sparsity. SimSMoE (Do et al., 2025) mitigates representation collapse by minimizing inter-expert similarity through a centered kernel alignment (CKA) loss, which promotes diversity and stable expert specialization. RMoE (Qiu et al., 2025) introduces a layerwise recurrent router that leverages a GRU to establish dependencies between routing decisions across consecutive layers, thereby improving cross-layer information sharing.

Next, we review MoE architectures that improve parameter efficiency. MoE-UT (Csordás et al., 2024), a Universal-Transformer–style MoE with layerwise parameter sharing, reduces model parameters and memory footprint under sparse activation while maintaining competitive performance. Mixture of Lookup Experts (MoLE) (Jie et al., 2025) reparameterizes trained experts into offloaded lookup tables, reducing memory and communication overhead during inference while preserving dense-like decoding latency. MoNE (Zhang et al., 2026) performs redundancy-aware structured pruning by replacing low-usage, highly correlated experts with lightweight novices, resulting in more compact MoE models with minimal accuracy loss and, in some cases, even accuracy gains.

Overall, while prior work often targets either expert utilization or parameter efficiency, GMoE is designed to jointly improve both by reusing Global Experts across layers and coupling routing decisions across layers via logit propagation.

3 Methodology

3.1 Preliminaries

Mixture of Experts. An MoE layer replaces a standard FFN layer with N parallel expert networks and a trainable router that produces a distribution over experts (Jacobs et al., 1991; Jordan and Jacobs, 1994). Modern sparsely gated variants activate only a subset of experts per token, enabling high capacity without a proportional increase in per-token compute (Shazeer et al., 2017). Given a token hidden representation $\mathbf{x} \in \mathbb{R}^d$, the router produces routing logits $\boldsymbol{\ell}(\mathbf{x}) = \mathbf{W}_{\text{route}}\mathbf{x}$ and a softmax distribution $p_j(\mathbf{x}) = \frac{\exp(\ell_j)}{\sum_{j'=1}^{N}\exp(\ell_{j'})}$ over experts. Let $\mathcal{S}_t^{(l)}$ denote the indices of the top-k experts selected for token t at layer l. The layer output is then computed as the weighted mixture

$$\mathbf{y}_t^{(l)} = \sum_{j \in \mathcal{S}_t^{(l)}} p_j(\mathbf{x}_t^{(l)}) E_j(\mathbf{x}_t^{(l)}). \tag{1}$$

In practice, top-2 routing and other routing variants are commonly used to improve load balancing and training efficiency.

Gated Recurrent Unit. To propagate router hidden states across layers, we employ a recurrent neural network (RNN) (Medsker et al., 2001), specifically the Gated Recurrent Unit (GRU) (Cho et al., 2014). For a routing input $\mathbf{x}^{(l)}$ at layer l and the previous router hidden state $\mathbf{h}^{(l-1)}$, the GRU computes reset and update gates

$$\mathbf{r}^{(l)} = \sigma(\mathbf{W}_r\mathbf{x}^{(l)} + \mathbf{U}_r\mathbf{h}^{(l-1)}), \tag{2}$$

$$\mathbf{z}^{(l)} = \sigma(\mathbf{W}_z\mathbf{x}^{(l)} + \mathbf{U}_z\mathbf{h}^{(l-1)}), \tag{3}$$

followed by the candidate state

$$\tilde{\mathbf{h}}^{(l)} = \tanh\big(\mathbf{W}_h\mathbf{x}^{(l)} + \mathbf{U}_h(\mathbf{r}^{(l)} \odot \mathbf{h}^{(l-1)})\big), \tag{4}$$

and the updated router hidden state

$$\mathbf{h}^{(l)} = (1 - \mathbf{z}^{(l)}) \odot \mathbf{h}^{(l-1)} + \mathbf{z}^{(l)} \odot \tilde{\mathbf{h}}^{(l)}. \tag{5}$$

This recurrent update propagates information across layers.

3.2 Global Mixture of Experts

As illustrated in Figure 1, GMoE combines Global Experts shared across layers with a layer-specific Local Expert and a Global Router whose GRU-based recurrent component is shared across layers, while the routing heads remain layer-specific.

Global Experts and Local Expert. Conventional MoE layers instantiate an expert set per layer, so the identity of expert j at layer l has no direct relation to expert j at layer $l{+}1$. In contrast, GMoE defines Global Experts $\{E_1, \ldots, E_N\}$ that are shared across all layers. This design reuses

Global Experts across layers, thereby reducing inter-layer redundancy while requiring substantially fewer model parameters. We further demonstrate in Appendix A that despite this sharing, individual layers in GMoE maintain high functional distinctiveness compared to baselines. Each layer also includes a Local Expert $E^{(l)}_{\text{local}}$, structurally identical to a Global Expert, to capture layer-specific components that are difficult to share globally. The Local Expert is always active, and its output is added to the routed output from the Global Experts. Formally, for token t at layer l:

$$\mathbf{y}_t^{(l)} = \sum_{j \in \mathcal{S}_t^{(l)}} p_{t,j}^{(l)} \, E_j(\mathbf{x}_t^{(l)}) + E^{(l)}_{\text{local}}(\mathbf{x}_t^{(l)}), \quad (6)$$

where $\mathbf{p}_t^{(l)} = \text{Softmax}(\boldsymbol{\ell}_t^{(l)}) \in \mathbb{R}^N$ is a distribution over only the Global Experts and $\sum_j p_{t,j}^{(l)} = 1$. The Local Expert does not participate in the routing distribution.

Global Router with Logit Propagation. Routing is computed by a Global Router with a GRU-based recurrent component shared across layers and layer-specific routing heads. This design introduces two complementary forms of sharing within the router. First, the GRU-based recurrent component of the Global Router is shared across all MoE layers, while the routing heads remain layer-specific, allowing routing dynamics to remain consistent across layers. Second, within the routing process itself, each layer's routing head receives a stop-gradient projection of the previous layer's routing logits, enabling information sharing across layers and iterative updates of routing decisions. Let $\mathbf{h}_t^{(l)} \in \mathbb{R}^H$ denote the router hidden state for token t at layer l. We concatenate it with a projection of the previous layer's routing logits and feed the result to a routing head:

$$\begin{aligned} \mathbf{h}_t^{(l)} &= \text{GRU}\big(\mathbf{x}_t^{(l)}, \mathbf{h}_t^{(l-1)}\big), \\ \boldsymbol{\ell}_t^{(l)} &= \mathbf{W}^{(l)}_{\text{route}}\big[\mathbf{h}_t^{(l)}; \, \text{sg}\big(\tilde{\boldsymbol{\ell}}_t^{(l-1)}\big)\big], \\ \mathbf{p}_t^{(l)} &= \text{Softmax}\big(\boldsymbol{\ell}_t^{(l)}\big), \end{aligned} \quad (7)$$

where $\tilde{\boldsymbol{\ell}}_t^{(l-1)}$ is a layer-normalized and linearly projected copy of the previous layer's routing logits, defined as $\tilde{\boldsymbol{\ell}}_t^{(l-1)} = \mathbf{W}_\ell \, \text{LN}\big(\boldsymbol{\ell}_t^{(l-1)}\big)$ with $\mathbf{W}_\ell \in \mathbb{R}^{D_\ell \times N}$, $\mathbf{W}^{(l)}_{\text{route}} \in \mathbb{R}^{N \times (H + D_\ell)}$, and $\text{sg}(\cdot)$ denotes the stop-gradient operator. Each token dispatches to its top-1 Global Expert for computation, while the full routing distribution $\mathbf{p}_t^{(l)}$ is used only for the balancing loss. Propagating stop-gradient routing logits across layers facilitates recurrent modeling of router hidden states: gradients do not flow through the propagated routing logits, while the Global Router is trained end-to-end via backpropagation. We empirically found that blocking gradient flow through the propagated routing logits stabilizes training and yields slight performance gains compared to propagating full gradients.

4 Experiments

4.1 Tasks and Datasets

Language Modeling. Following prior works (Xie et al., 2023; Su et al., 2025), we use The Pile dataset (Gao et al., 2020) as our primary pre-training corpus. The Pile is a large-scale, publicly available dataset comprising 22 diverse domains and over 825 GB of English text, designed for language modeling. To assess generalization performance, we additionally evaluate on held-out subsets of the English portion of CC100 and OpenWebText (Conneau et al., 2020; Gokaslan and Cohen, 2019), as well as the test set of WikiText-103 (Merity et al., 2017). These corpora cover a broad range of linguistic styles, allowing a comprehensive evaluation across varying text distributions. Due to limited computational resources, all models are pre-trained on 50B tokens unless otherwise specified.

Benchmark. We evaluate all models using the lm-evaluation-harness framework (Gao et al., 2024). We report zero-shot accuracy on ARC-C and ARC-E (Clark et al., 2018), BLiMP (Warstadt et al., 2020), HellaSwag (Zellers et al., 2019), OBQA (Mihaylov et al., 2018), and PIQA (Bisk et al., 2020), and 5-shot accuracy on RACE (Lai et al., 2017).

4.2 Experimental Setup

All models follow a GPT-style decoder-only Transformer architecture (Radford et al., 2019). Following prior works (Fedus et al., 2022; Lepikhin et al., 2021), we replace the FFN layer in each Transformer block with an MoE layer, applied to every other block for the baselines. In contrast, GMoE applies an MoE layer at every Transformer block, as its logit propagation mechanism requires uninterrupted routing across layers. To further investigate the scalability and parameter efficiency of our approach, we introduce GMoE-32 and GMoE-64, which increase the number of Global Experts to

32 and 64, respectively. We further define GMoE-Max as a variant that maximizes the number of Global Experts without exceeding the parameter budget of the corresponding baselines. For language modeling, we set the maximum sequence length to 1,024 and train for 95K steps. We employ the GPT-2 tokenizer (Radford et al., 2019) implemented via the tiktoken[1] library, with a vocabulary size of 50,257. All models are optimized using AdamW (Loshchilov and Hutter, 2019) with an initial learning rate of 3×10^{-4} and a cosine decay schedule. We employ the standard load balancing loss from Switch Transformer (Fedus et al., 2022) with a coefficient of $\alpha = 0.01$, add a probe regularizer for numerical stability, and set the capacity factor to 1.25 unless otherwise specified. Additional hyperparameters are provided in Appendix B.

We compare the proposed GMoE against six baselines: Dense and five representative MoE architectures—Switch (Fedus et al., 2022), Switch-SE (Switch augmented with an independent always-on expert per layer, referred to as a shared expert), GShard (Lepikhin et al., 2021), Hash (Roller et al., 2021), and DeepSeek-MoE (Dai et al., 2024). For DeepSeek-MoE, we follow the DeepSeek-MoE 2B configuration: we reduce the expert hidden size to 0.25×, increase the number of experts by 4×, and use exactly one shared expert. Under this configuration, Dense, Switch, and Hash have comparable per-token compute. In contrast, Switch-SE, GShard, DeepSeek-MoE, and GMoE incur higher per-token cost because of the additional shared expert, top-2 routing, fine-grained expert partitioning with a shared expert, and every-layer MoE with a GRU-based router for logit propagation, respectively. All experiments are performed on a single node with eight NVIDIA RTX PRO 6000 Blackwell GPUs.

4.3 Results

Language Modeling. As summarized in Table 1, we compare GMoE against selected baselines. Following prior work (Jie et al., 2025), we use the number of activated parameters as a proxy for per-token FFN compute, calculated by summing the FFN parameters involved in the forward pass across all layers. Note that the number of activated parameters can exceed the model parameters due to the reuse of Global Experts. Across the Small, Base, and Large models, MoE architectures consistently achieve lower perplexity than Dense, with particularly pronounced gains for the Small model. While consistently outperforming Dense, GMoE distinguishes itself from conventional MoEs by achieving competitive perplexity with substantially fewer model parameters. Furthermore, we observe that as the number of Global Experts increases, GMoE demonstrates consistent performance improvements with only a moderate increase in model parameters. Notably, GMoE-Max achieved the best performance across all model scales. These results indicate that GMoE delivers strong parameter efficiency, offering a viable lightweight alternative to conventional MoE models, as further evidenced by the memory efficiency analysis in Appendix C. Additional controlled comparisons under matched model parameters and matched activated parameters further support this conclusion: GMoE outperforms the baselines under matched model parameters, while GMoE-Max remains competitive under matched activated parameters despite using fewer model parameters, as shown in Appendix D.

Benchmark. As summarized in Table 2, we compare the previously evaluated models across seven downstream benchmarks. Overall, GMoE achieves competitive accuracy while using substantially fewer model parameters. For the Small model, GMoE achieves higher accuracy than all baselines except Switch-SE while using only 0.57× the model parameters. For the Base model, GMoE performs comparably to Switch with merely 0.37× the model parameters. Furthermore, the GMoE variants highlight the architecture's scalability, as GMoE-32 and GMoE-64 exhibit strong parameter efficiency while achieving accuracy comparable to or surpassing the baselines. Notably, GMoE-Max consistently outperforms the baselines across all model scales. In addition, comparison with the recent fine-grained MoE architecture DeepSeek-MoE further highlights the parameter efficiency of GMoE. For the Small and Base models, DeepSeek-MoE underperforms GMoE despite using 1.76× to 2.73× more model parameters, suggesting that fine-grained routing may degrade parameter efficiency in smaller, resource-constrained models. For the Large model, while DeepSeek-MoE becomes comparable at larger scales, GMoE-Max still achieves better accuracy under an equivalent parameter budget.

Given that DeepSeek-MoE becomes comparable in the Large model, a natural question is whether

[1] https://github.com/openai/tiktoken

Model	# Params	# Activated Params	# Experts	The Pile	CC100	OpenWebText	WikiText-103	Avg ↓
Small Model								
Dense	45M	45M	-	16.64	71.38	40.37	84.07	53.12
Switch	140M	45M	16	13.67	62.39	33.22	72.42	45.43
Switch-SE	152M	51M	16	13.01	63.71	31.62	68.33	44.17
GShard	140M	51M	16	13.44	64.32	32.57	71.29	45.41
Hash	140M	45M	16	14.10	63.99	34.36	79.04	47.87
DeepSeek-MoE	141M	47M	64	16.07	70.31	37.17	82.12	51.42
GMoE	80M	58M	16	13.98	64.97	33.88	74.14	46.74
GMoE-32	114M	58M	32	13.04	**60.69**	31.35	68.70	43.45
GMoE-Max	139M	58M	44	**12.64**	60.98	**30.60**	**67.29**	**42.88**
Base Model								
Dense	124M	124M	-	11.94	52.36	27.05	51.97	35.83
Switch	549M	124M	16	9.72	46.58	22.53	44.17	30.75
Switch-SE	606M	153M	16	9.26	46.32	21.29	41.82	29.67
GShard	549M	153M	16	9.50	47.49	22.03	41.69	30.18
Hash	549M	124M	16	10.11	50.62	23.65	48.12	33.13
DeepSeek-MoE	557M	131M	64	10.18	65.65	23.29	48.29	36.86
GMoE	204M	181M	16	10.39	50.16	24.15	48.58	33.32
GMoE-32	280M	181M	32	9.87	48.17	22.83	44.96	31.46
GMoE-64	432M	181M	64	9.38	49.38	21.67	44.36	31.20
GMoE-Max	546M	181M	88	**9.19**	**46.09**	**21.11**	**40.78**	**29.29**
Large Model								
Dense	355M	355M	-	8.96	43.20	20.23	37.49	27.47
Switch	1.9B	355M	16	7.79	40.91	17.33	31.65	24.42
Switch-SE	2.1B	455M	16	7.62	40.15	17.15	31.45	24.09
GShard	1.9B	455M	16	7.76	39.44	17.28	31.50	24.00
Hash	1.9B	355M	16	8.22	46.84	18.60	35.10	27.19
DeepSeek-MoE	1.9B	380M	64	7.90	43.22	17.62	34.57	25.83
GMoE	496M	556M	16	8.53	43.08	19.20	36.35	26.79
GMoE-32	632M	556M	32	8.26	43.23	18.47	34.61	26.14
GMoE-64	903M	556M	64	7.96	42.11	17.74	32.85	25.17
GMoE-Max	1.9B	556M	177	**7.52**	**39.28**	**16.84**	**31.25**	**23.72**

Table 1: Comparison of MoE architectures across Small, Base, and Large models using perplexity, evaluated on held-out subsets of the English portions of The Pile, CC100, OpenWebText, and WikiText-103 test sets. The # Params column reports model parameters, the # Activated Params column reports the number of parameters activated during inference, and the # Experts column reports the number of routed experts per MoE layer. Lower values indicate better performance. Top-1 results are shown in **bold**, and Top-2 results are underlined.

GMoE is also compatible with fine-grained experts. Since our earlier results suggest that fine-grained routing is less effective in the Small and Base models, we restrict this additional experiment to the Large model. Specifically, we adopt the same fine-grained configuration as DeepSeek-MoE, and the results are summarized in Table 3. As shown in the table, applying this fine-grained variant improves performance, indicating that the proposed GMoE architecture is also compatible with fine-grained experts.

5 Analysis

5.1 Ablation Study

We conduct ablation studies on various components using the Base model, where GMoE achieves 10.39 PPL on The Pile, as shown in Table 4. First, to evaluate the Local Expert, we reallocate its model parameter budget to the Global Experts, increasing their count from 16 to 28. To match per-token compute, we apply top-2 routing over this expanded set. Removing the Local Expert results in the largest performance degradation to 11.52 PPL (+1.13), confirming the necessity of layer-specific adaptation. Second, we analyze the routing components. Removing the cross-layer logit propagation pathway leads to 11.34 PPL (+0.95), and disabling the router hidden state results in 10.87 PPL (+0.48). These results indicate that the path-aware routing components of GMoE, namely maintaining router hidden states across layers and explicit logit propagation, are important to its performance gains. Third, we investigate the impact of MoE layer frequency. GMoE applies MoE layers at every block

Model	# Params	# Activated Params	ARC-C	ARC-E	BLiMP	HellaSwag	OBQA	PIQA	RACE	Avg ↑
Small Model										
Dense	45M	45M	17.41	37.75	71.23	26.49	13.00	58.16	24.31	35.48
Switch	140M	45M	16.89	40.03	73.49	26.67	12.40	**59.90**	25.26	36.38
Switch-SE	152M	51M	17.58	39.56	73.95	27.13	12.60	59.47	26.79	36.73
GShard	140M	51M	17.58	39.06	72.59	26.86	13.80	59.03	24.88	36.26
Hash	140M	45M	17.58	38.93	72.39	26.96	12.80	58.81	25.17	36.09
DeepSeek-MoE	141M	47M	18.94	39.14	72.51	26.98	12.80	58.87	25.17	36.34
GMoE	80M	58M	18.69	38.51	73.45	27.11	**14.20**	59.30	24.31	36.51
GMoE-32	114M	58M	17.92	40.03	**74.26**	27.22	13.20	59.74	**27.08**	37.06
GMoE-Max	139M	58M	**19.45**	**41.04**	74.23	**27.26**	13.20	58.49	26.79	**37.21**
Base Model										
Dense	124M	124M	18.26	42.51	75.44	27.59	12.60	61.43	27.46	37.90
Switch	549M	124M	18.94	45.45	79.45	28.73	14.20	62.24	27.85	39.55
Switch-SE	606M	153M	20.14	45.03	**79.67**	29.55	16.20	63.17	27.92	40.24
GShard	549M	153M	19.97	44.95	77.88	29.02	14.00	63.17	26.79	39.40
Hash	549M	124M	19.45	44.49	76.17	28.54	14.00	62.13	27.37	38.88
DeepSeek-MoE	557M	131M	20.14	44.28	77.53	28.96	14.80	61.70	27.37	39.25
GMoE	204M	181M	19.97	45.08	77.49	28.39	15.00	61.64	**29.00**	39.51
GMoE-32	280M	181M	**20.65**	44.44	77.52	28.82	**17.20**	61.92	27.94	39.78
GMoE-64	432M	181M	20.22	46.25	79.37	29.17	16.00	63.06	27.94	40.29
GMoE-Max	546M	181M	20.05	**46.42**	79.38	**29.78**	16.80	**63.44**	26.89	**40.39**
Large Model										
Dense	355M	355M	19.88	47.73	79.96	30.12	14.80	63.76	30.14	40.91
Switch	1.9B	355M	20.14	48.99	80.88	32.42	16.20	65.89	30.14	42.09
Switch-SE	2.1B	455M	21.25	49.10	81.15	32.60	17.00	66.05	30.45	42.51
GShard	1.9B	455M	21.16	48.78	81.61	32.48	17.80	65.89	30.33	42.58
Hash	1.9B	355M	20.65	50.25	79.92	31.06	17.20	64.85	29.76	41.96
DeepSeek-MoE	1.9B	380M	21.59	**50.71**	81.51	32.71	**18.80**	66.21	29.81	43.05
GMoE	496M	556M	21.84	48.78	80.74	30.85	15.60	65.34	29.47	41.80
GMoE-32	632M	556M	21.50	49.20	81.24	31.54	15.20	64.91	29.95	41.93
GMoE-64	903M	556M	21.08	48.70	**81.76**	32.13	17.20	**66.43**	29.86	42.45
GMoE-Max	1.9B	556M	**22.05**	50.60	81.45	**32.90**	17.60	66.15	**31.10**	**43.12**

Table 2: Comparison of MoE architectures across Small, Base, and Large models using accuracy, evaluated on seven downstream benchmarks. Higher values indicate better performance.

to maintain continuous routing dynamics. To verify the necessity of this design, we evaluated a variant that applies MoE layers only to every other block. This variant yields a perplexity of 10.94 (+0.55), indicating that continuous routing dynamics are important for GMoE. This suggests that the effectiveness of logit propagation relies on the continuous refinement of routing decisions, and that interleaving dense layers disrupts the propagation of routing information across layers. Taken together, these ablations indicate that the architectural components enabling path-aware routing in GMoE are important to its overall efficacy.

5.2 Mitigating Path Collapse

GMoE couples routing decisions across layers via logit propagation, raising the question of whether such coupling induces path collapse—a regime in which a small number of cross-layer routing paths dominate most tokens—or instead sustains diverse routing. As discussed in Section 1, we interpret path collapse as a manifestation of compounded redundancy. Unlike weight similarity, which captures static parameter redundancy, compounded redundancy is a dynamic phenomenon best operationalized by the utilization of the combinatorial routing space; when most tokens are funneled through a small set of cross-layer routing paths, many experts are exercised under a limited range of routing contexts, thereby limiting the model's effective capacity.

Mechanistically, GMoE does not merely couple routing decisions across layers; it conditions each routing decision on the previous layer's router hidden state. In conventional MoEs, routers at each layer operate independently based only on the current hidden representation, so tokens with similar hidden representations are likely to be mapped to the same experts repeatedly, which can induce path collapse. By contrast, in GMoE, the router

Model	# Activated Params	# Experts	ARC-C	ARC-E	BLiMP	HellaSwag	OBQA	PIQA	RACE	Avg ↑
Large Model										
GMoE	556M	16	21.84	48.78	80.74	30.85	15.60	65.34	**29.47**	41.80
+ Fine-grained	556M	64	**22.29**	**50.58**	**81.19**	**31.00**	**17.60**	**65.59**	28.72	**42.42**

Table 3: Comparison of the GMoE architecture and its fine-grained variant on the Large model using accuracy, evaluated across seven downstream benchmarks. Higher values indicate better performance.

Model	Perplexity	Δ Perplexity
GMoE	10.39	–
w/o Local Expert	11.52	+1.13
w/o Logit Propagation	11.34	+0.95
w/o Router Hidden State	10.87	+0.48
MoE in Every Other Block	10.94	+0.55

Table 4: Ablation study on the Base model. We report perplexity on The Pile and the change relative to GMoE.

Model	# Unique	$N_{\text{eff,path}}$	Top-1	Top-10
Switch	25,587	286.0	25.65%	49.50%
GShard	17,641	110.7	45.55%	53.05%
GMoE	81,561	1217.7	11.15%	43.16%

Table 5: Routing path statistics across six MoE layers under top-1 routing. We report the number of unique paths, the effective number of paths, and concentration measured by the cumulative mass of the top paths.

computes routing logits conditioned on the previous layer's router hidden state and the projected previous-layer routing logits, as defined in Eq. (7). Thus, even tokens with similar hidden representations at the current layer can be routed differently. This additional path-aware conditionality helps break the symmetry that would otherwise funnel many tokens into the same cross-layer routes, thereby mitigating path collapse and encouraging more diverse path exploration.

Accordingly, we analyze the empirical distribution of top-1 routing paths under the Base model. For each token t, we record the top-1 expert index at each MoE layer. Let $\mathcal{L} = \{l_1, \ldots, l_M\}$ denote the set of MoE layers under consideration. For fair comparison with Switch and GShard, which apply MoE to every other Transformer block, we evaluate all models on the same six MoE layers $\mathcal{L} = \{0, 2, 4, 6, 8, 10\}$ and use the same subset for GMoE. The routing path of token t, denoted as $\mathbf{r}_t$, is defined as the sequence of top-1 expert indices selected across these layers, as formally defined in Appendix E. Although GShard employs top-2 routing, we track only the primary (top-1) expert at each layer to obtain a consistent path definition across models.

Let $\{\bar{\mathbf{r}}_k\}_{k=1}^{K}$ denote the set of unique paths observed in $\{\mathbf{r}_t\}_{t=1}^{T}$, and let $n(\bar{\mathbf{r}}_k)$ be the number of tokens assigned to $\bar{\mathbf{r}}_k$. Based on the empirical distribution of these paths, we compute the path entropy H_{path} and the corresponding effective number of paths $N_{\text{eff,path}} = 2^{H_{\text{path}}}$. To characterize path concentration directly, we also report the cumulative probability mass of the top-k most frequent paths, $M_k = \sum_{i=1}^{k} p(\bar{\mathbf{r}}_{[i]})$ for $k \in \{1, 10\}$, where $[i]$ indexes paths sorted in descending order of $p(\cdot)$.

Table 5 summarizes routing diversity and concentration across models under identical evaluation conditions. Overall, GMoE exhibits substantially lower path concentration than Switch and GShard while realizing a much richer routing space. In particular, GMoE increases path entropy to $H_{\text{path}} = 10.25$ bits ($N_{\text{eff,path}} = 1217.7$) and reduces the Top-1 mass to 11.15%, compared to 25.65% for Switch and 45.55% for GShard. These results are consistent with GMoE mitigating path collapse, which we view as a manifestation of compounded redundancy.

5.3 Path Specialization

While the analysis in Section 5.2 shows that GMoE exploits a larger and more diverse set of routing paths than the baselines, a critical question remains: does this increased routing diversity reflect meaningful specialization or merely stochastic routing noise? If routing paths are semantically meaningful, tokens assigned to the same cross-layer path should share coherent semantic representations; otherwise, tokens grouped within a path would be semantically heterogeneous.

This specialization is also supported by the architectural design of GMoE. Because Global Experts are shared across layers, the expert index space is semantically aligned throughout the network, unlike conventional MoEs where expert indices are layer-specific and not directly comparable across layers. As a result, the propagated routing logits remain semantically meaningful across layers.

Combined with the recurrent router hidden state, this turns routing into a sequential decision process rather than a sequence of independent greedy assignments. Consequently, the router can learn to select experts that functionally complement those chosen in preceding layers, encouraging stable multi-hop routing paths specialized for coherent semantic patterns.

To investigate this question, we analyze the semantic consistency of routing paths by treating the set of tokens assigned to a specific path $\mathbf{r}$ as a cluster in the representation space. We sample 1,000 tokens from each of the top-10 most frequent paths and evaluate the resulting clustering structure. Let $\mathcal{C}_k$ be the set of tokens belonging to the k-th path ($k = 1, \ldots, K$), with $n_k = |\mathcal{C}_k|$, and let $\boldsymbol{\mu}_k$ denote the centroid of cluster $\mathcal{C}_k$. The global centroid is denoted by $\boldsymbol{\mu}$. We quantify clustering quality using three complementary metrics, fully detailed in Appendix E:

(1) kNN Purity: This metric measures the local label consistency of clusters. For a sample $\mathbf{x}_i$ with path label y_i, purity is the fraction of its k nearest neighbors sharing the same label.

(2) Inter/Intra Ratio: To explicitly measure cluster separation relative to compactness in the representation space, we define the intra-cluster distance $\mathcal{D}_{\text{intra}}$ and inter-cluster distance $\mathcal{D}_{\text{inter}}$ using cosine distance, consistent with the geometry of the representation space. We report the ratio $\mathcal{D}_{\text{inter}}/\mathcal{D}_{\text{intra}}$ as a single summary statistic. Higher values indicate that path-induced clusters are compact internally and well-separated from each other.

(3) Calinski-Harabasz Index (CHI): We also report CHI, which evaluates the ratio of between-cluster dispersion to within-cluster dispersion.

Quantitatively, GMoE achieves a kNN Purity of 0.9971, substantially exceeding Switch (0.8085) and GShard (0.6628), which suggests markedly stronger local semantic consistency. GMoE also achieves the highest Inter/Intra Ratio of 5.2146 and a CHI of 2565.7, indicating stronger cluster separation and global cluster structure than the baselines.

Figure 2 further examines the robustness of path specialization by varying the number of top-k routing paths from k=2 to 100, using 200 sampled tokens per path. Across this range, GMoE consistently achieves higher kNN Purity, Inter/Intra Ratio, and CHI, indicating that the observed path specialization is robust to the choice of k rather than being an artifact of a fixed evaluation setting.

Qualitatively, Figure 3 visualizes routing path

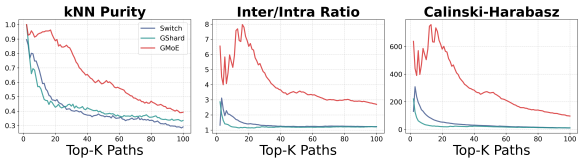

Figure 2: Trends of path specialization metrics as the number of top-k routing paths increases. We report kNN Purity, Inter/Intra Ratio, and CHI while varying k from 2 to 100, with 200 sampled tokens per path.

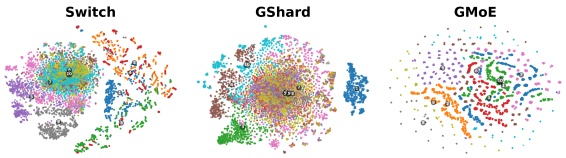

Figure 3: t-SNE visualization of token representations grouped by their assigned routing paths among the top-10 most frequent paths, where numbered labels indicate the centroids of each path cluster.

clusters using t-SNE (van der Maaten and Hinton, 2008). Switch and GShard exhibit substantial overlap across different routing paths, with dispersed and entangled clusters, suggesting less distinct routing patterns, where semantically similar tokens may be assigned to divergent paths. In contrast, GMoE produces well-separated and compact clusters, indicating that semantically similar tokens are assigned to specialized cross-layer routing paths.

6 Conclusion

We identify compounded redundancy in conventional MoE architectures, where independent per-layer experts lead to inefficient parameter utilization and path collapse. To address this, we propose GMoE, which employs Global Experts shared across layers and a Global Router with logit propagation. This design reuses Global Experts across layers and couples routing decisions, helping mitigate both inter-layer and intra-layer redundancy. Our experiments demonstrate that GMoE achieves competitive performance relative to the baselines while using substantially fewer model parameters. Furthermore, our analysis shows that GMoE induces diverse and semantically specialized routing paths, suggesting meaningful expert specialization. Consequently, GMoE represents a promising direction for deploying efficient MoE architectures in resource-constrained environments, potentially including on-device/edge settings where conventional MoE models are often impractical due to their large memory footprint.

Limitations

While GMoE demonstrates promising parameter efficiency and competitive performance, several limitations remain. First, due to computational resource constraints, our experiments were confined to models with up to 1.9B parameters trained on 50B tokens. Given that state-of-the-art LLMs are trained at substantially larger scales, it remains important for future work to verify whether the advantages of GMoE persist at larger model sizes and with much larger training corpora, and whether they continue to align with scaling-law trends. We provide additional analysis of expert utilization across the Small, Base, and Large models, measured by load balance entropy, in Appendix F. Second, the core mechanism of logit propagation relies on the continuous evolution of router hidden states, necessitating the placement of MoE layers in every Transformer block. This design deviates from the conventional practice of interleaving MoE layers (e.g., every other block) and may increase routing overhead. Future work should explore extending the architectural flexibility of GMoE to non-continuous MoE layer configurations while preserving its benefits.

Acknowledgments

This work was supported by Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No. RS-2022-II220608/2022-0-00608, Artificial Intelligence research about multimodal interactions for empathetic conversations with humans, No. IITP-2026-RS-2024-00360227, Leading Generative AI Human Resources Development, No. RS-2025-25442824, AI Star Fellowship Program (Ulsan National Institute of Science and Technology), & No. RS-2020-II201336, Artificial Intelligence graduate school support (UNIST)).

References

Yonatan Bisk, Rowan Zellers, Ronan Le bras, Jianfeng Gao, and Yejin Choi. 2020. Piqa: Reasoning about physical commonsense in natural language. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(05):7432–7439.

Ruisi Cai, Yeonju Ro, Geon-Woo Kim, Peihao Wang, Babak Ehteshami Bejnordi, Aditya Akella, and Zhangyang Wang. 2024. Read-me: Refactorizing llms as router-decoupled mixture of experts with system co-design. In *Advances in Neural Information Processing Systems*, volume 37, pages 116126–116148. Curran Associates, Inc.

Zewen Chi, Li Dong, Shaohan Huang, Damai Dai, Shuming Ma, Barun Patra, Saksham Singhal, Payal Bajaj, XIA SONG, Xian-Ling Mao, Heyan Huang, and Furu Wei. 2022. On the representation collapse of sparse mixture of experts. In *Advances in Neural Information Processing Systems*, volume 35, pages 34600–34613. Curran Associates, Inc.

Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha, Qatar. Association for Computational Linguistics.

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 preprint arXiv:1803.05457*.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2020. Unsupervised cross-lingual representation learning at scale. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8440–8451, Online. Association for Computational Linguistics.

Róbert Csordás, Kazuki Irie, Jürgen Schmidhuber, Christopher Potts, and Christopher D. Manning. 2024. Moeut: Mixture-of-experts universal transformers. In *Advances in Neural Information Processing Systems*, volume 37, pages 28589–28614. Curran Associates, Inc.

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. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1280–1297, Bangkok, Thailand. Association for Computational Linguistics.

Damai Dai, Li Dong, Shuming Ma, Bo Zheng, Zhifang Sui, Baobao Chang, and Furu Wei. 2022. StableMoE: Stable routing strategy for mixture of experts. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7085–7095, Dublin, Ireland. Association for Computational Linguistics.

Zihang Dai, Guokun Lai, Yiming Yang, and Quoc Le. 2020. Funnel-transformer: Filtering out sequential redundancy for efficient language processing. In *Advances in Neural Information Processing Systems*, volume 33, pages 4271–4282. Curran Associates, Inc.

Fahim Dalvi, Hassan Sajjad, Nadir Durrani, and Yonatan Belinkov. 2020. Analyzing redundancy in pretrained transformer models. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4908–4926, Online. Association for Computational Linguistics.

Giang Do, Hung Le, and Truyen Tran. 2025. SimSMoE: Toward efficient training mixture of experts via solving representational collapse. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pages 2012–2025, Albuquerque, New Mexico. Association for Computational Linguistics.

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.

Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, and 1 others. 2020. The pile: An 800gb dataset of diverse text for language modeling. *arXiv preprint arXiv:2101.00027*.

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.

Aaron Gokaslan and Vanya Cohen. 2019. Openwebtext corpus. `http://Skylion007.github.io/OpenWebTextCorpus`.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Thomas Hennigan, Eric Noland, Katherine Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karén Simonyan, Erich Elsen, and 3 others. 2022. An empirical analysis of compute-optimal large language model training. In *Advances in Neural Information Processing Systems*, volume 35, pages 30016–30030. Curran Associates, Inc.

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.

Shibo Jie, Yehui Tang, Kai Han, Yitong Li, Duyu Tang, Zhi-Hong Deng, and Yunhe Wang. 2025. Mixture of lookup experts. In *Forty-second International Conference on Machine Learning*.

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

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. 2020. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*.

Guokun Lai, Qizhe Xie, Hanxiao Liu, Yiming Yang, and Eduard Hovy. 2017. RACE: Large-scale ReAding comprehension dataset from examinations. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 785–794, Copenhagen, Denmark. Association for Computational Linguistics.

Dmitry Lepikhin, HyoukJoong Lee, Yuanzhong Xu, Dehao Chen, Orhan Firat, Yanping Huang, Maxim Krikun, Noam Shazeer, and Zhifeng Chen. 2021. {GS}hard: Scaling giant models with conditional computation and automatic sharding. In *International Conference on Learning Representations*.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In *International Conference on Learning Representations*.

Larry R Medsker, Lakhmi Jain, and 1 others. 2001. Recurrent neural networks. *Design and applications*, 5(64-67):2.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2017. Pointer sentinel mixture models. In *International Conference on Learning Representations*.

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 *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2381–2391, Brussels, Belgium. Association for Computational Linguistics.

Telmo Pires, António Vilarinho Lopes, Yannick Assogba, and Hendra Setiawan. 2023. One wide feedforward is all you need. In *Proceedings of the Eighth Conference on Machine Translation*, pages 1031–1044, Singapore. Association for Computational Linguistics.

Zihan Qiu, Zeyu Huang, Shuang Cheng, Yizhi Zhou, Zili Wang, Ivan Titov, and Jie Fu. 2025. Layerwise recurrent router for mixture-of-experts. In *The Thirteenth International Conference on Learning Representations*.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, and 1 others. 2019. Language models are unsupervised multitask learners. *OpenAI Blog*.

Stephen Roller, Sainbayar Sukhbaatar, arthur szlam, and Jason Weston. 2021. Hash layers for large sparse models. In *Advances in Neural Information Processing Systems*, volume 34, pages 17555–17566. Curran Associates, Inc.

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. In *International Conference on Learning Representations*.

Zhenpeng Su, Xing W, Zijia Lin, Yizhe Xiong, Minxuan Lv, Guangyuan Ma, Hui Chen, Songlin Hu, and Guiguang Ding. 2025. CartesianMoE: Boosting knowledge sharing among experts via Cartesian product routing in mixture-of-experts. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 10040–10055, Albuquerque, New Mexico. Association for Computational Linguistics.

Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(86):2579–2605.

Alex Warstadt, Alicia Parrish, Haokun Liu, Anhad Mohananey, Wei Peng, Sheng-Fu Wang, and Samuel R. Bowman. 2020. BLiMP: The benchmark of linguistic minimal pairs for English. *Transactions of the Association for Computational Linguistics*, 8:377–392.

Sang Michael Xie, Hieu Pham, Xuanyi Dong, Nan Du, Hanxiao Liu, Yifeng Lu, Percy S Liang, Quoc V Le, Tengyu Ma, and Adams Wei Yu. 2023. Doremi: Optimizing data mixtures speeds up language model pretraining. In *Advances in Neural Information Processing Systems*, volume 36, pages 69798–69818. Curran Associates, Inc.

Yuanhang Yang, Shiyi Qi, Wenchao Gu, Chaozheng Wang, Cuiyun Gao, and Zenglin Xu. 2024. XMoE: Sparse models with fine-grained and adaptive expert selection. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 11664–11674, Bangkok, Thailand. Association for Computational Linguistics.

Rongjie Yi, Liwei Guo, Shiyun Wei, Ao Zhou, Shangguang Wang, and Mengwei Xu. 2025. Edgemoe: Empowering sparse large language models on mobile devices. *IEEE Transactions on Mobile Computing*, 24(8):7059–7073.

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.

Geng Zhang, Han Yuxuan, Yuxuan Lou, Yiqi Zhang, Wangbo Zhao, and Yang You. 2026. MoNE: Replacing redundant experts with lightweight novices for structured pruning of moe. In *The Fourteenth International Conference on Learning Representations*.

Hao Zhao, Zihan Qiu, Huijia Wu, Zili Wang, Zhaofeng He, and Jie Fu. 2024. HyperMoE: Towards better mixture of experts via transferring among experts. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 10605–10618, Bangkok, Thailand. Association for Computational Linguistics.

Yanqi Zhou, Tao Lei, Hanxiao Liu, Nan Du, Yanping Huang, Vincent Zhao, Andrew M Dai, zhifeng Chen, Quoc V Le, and James Laudon. 2022. Mixture-of-experts with expert choice routing. In *Advances in Neural Information Processing Systems*, volume 35, pages 7103–7114. Curran Associates, Inc.

A Layer-wise Transformations

A defining feature of GMoE is that the Global Experts are shared across all layers. While this design substantially reduces model parameters and prevents inter-layer redundancy, it also raises a potential concern: sharing the same FFN experts across layers may blur functional layer boundaries and weaken the functional role of individual layers. GMoE is explicitly designed to mitigate this risk. Each layer is equipped with a dedicated Local Expert that remains layer-specific, and routing decisions are coupled across layers through logit propagation. In this section, we empirically examine whether these mechanisms are sufficient to ensure that each layer still performs a nontrivial transformation of its input, despite sharing Global Experts across layers.

To this end, we analyze layer-wise representation changes using the Layer Effect Index (LEI), which measures the dissimilarity between a layer's input and output representations. For layer l, LEI is defined as:

$$\mathrm{LEI}^{(l)} = 1 - \frac{1}{T}\sum_{t=1}^{T} \frac{\mathbf{x}_t^{(l)} \cdot \mathbf{y}_t^{(l)}}{\|\mathbf{x}_t^{(l)}\|_2 \, \|\mathbf{y}_t^{(l)}\|_2}. \quad (8)$$

An LEI value close to zero indicates near-identity behavior, whereas larger values indicate stronger representational updates.

We observe clear layer-wise differences in LEI for GMoE and the baselines. Despite sharing Global Experts across layers, GMoE consistently exhibits high LEI values throughout the network. In several layers, LEI exceeds 1.0, which occurs when the cosine similarity between input and output representations becomes negative, indicating strong reorientation in representation space. By contrast, for the conventional baselines that apply MoE layers to every other Transformer block, we observe a different pattern. MoE layers tend

to exhibit lower LEI values, while the remaining dense FFN layers show relatively higher LEI. Notably, although these dense layers are architecturally identical to those in Dense, Dense itself displays uniformly low LEI across layers. Quantitatively, GMoE achieves the highest average LEI (1.09), followed by Hash (1.02), GShard (1.00), Switch (0.95), and Dense (0.76). These results indicate that sharing Global Experts in GMoE does not weaken the functional role of individual layers; rather, each layer continues to contribute meaningful representational transformations.

B Experimental Setup Details

Hyperparameters	Small	Base	Large
Embedding & Hidden Size	512	768	1,024
FFN Inner Hidden Size	2,048	3,072	4,096
Number of Attention Heads	8	12	16
Number of Transformer Blocks	6	12	24
Number of MoE Layers	3	6	12
Sequence Length	1,024		
Batch Size	524K tokens		
Optimizer	AdamW		
Weight Decay / Betas	0.1 / (0.9, 0.95)		
Maximum Learning Rate	3e-4		
Learning Rate Scheduler	Cosine Decay		
Total Training Steps	95K		
Warm-up Ratio	0.1		
Gradient Clip Norm	1.0		
Capacity Factor	1.25		
Load-Balance Loss	0.01		

Table 6: Hyperparameter configurations for Small, Base, and Large models.

Table 6 presents the detailed hyperparameter configurations employed for pre-training the Small, Base, and Large model variants used in our experiments.

C Memory Efficiency Analysis

In this section, we analyze the memory efficiency of the proposed architecture, which is a critical factor for deploying LLMs in resource-constrained environments. To ensure a rigorous comparison, peak memory usage was measured on a single NVIDIA A100 GPU using bfloat16 precision with a fixed batch size of 64.

As shown in Table 7, GMoE maintains a memory footprint comparable to Dense across all scales,

Model	Peak Memory Usage (GB) ↓
Small Model	
Dense	1.15
Switch	1.68
GShard	1.69
Hash	1.69
GMoE	1.26
Base Model	
Dense	1.81
Switch	4.20
GShard	4.19
Hash	4.19
GMoE	1.98
Large Model	
Dense	3.58
Switch	12.03
GShard	12.03
Hash	12.02
GMoE	3.63

Table 7: Peak memory usage analysis across Small, Base, and Large models. Values represent the peak allocated GPU memory, excluding the reserved cache.

effectively mitigating the high memory cost typically associated with MoE architectures. The disparity in memory usage becomes particularly pronounced as the model scale increases. For the Large model, while the memory consumption of conventional MoE baselines surges to approximately 12.03 GB, GMoE remains highly efficient at 3.63 GB—closely tracking the 3.58 GB footprint of Dense. This result confirms that GMoE successfully decouples the model's effective capacity from its physical memory requirements, offering a practical solution for deploying high-capacity models within limited VRAM budgets.

D Additional Results

D.1 Matched Model Parameters

Table 8 further evaluates the models under matched model parameters. We downscale the baselines by reducing the FFN inner hidden size to 731 so that their model parameters align with those of GMoE, and perform this comparison for the Base model. Under these matched model parameters, GMoE achieves lower perplexity than the baselines, indicating more effective utilization of model parameters. These results provide additional evidence

Model	# Params	# Experts	The Pile	CC100	OpenWebText	WikiText-103	Avg ↓
Switch	204M	16	10.77	52.83	25.21	48.92	34.43
GShard	204M	16	10.91	52.32	25.66	51.43	35.08
Hash	204M	16	11.26	54.18	26.47	54.69	36.65
GMoE	204M	16	**10.39**	**50.16**	**24.15**	**48.58**	**33.32**

Table 8: Comparison of MoE architectures under matched model parameters using perplexity. Lower values indicate better performance. All baselines are scaled to match the model parameters of GMoE by reducing the FFN inner hidden size to 731.

Model	ARC-C	ARC-E	BLiMP	HellaSwag	OBQA	PIQA	RACE	Avg ↑
Switch	**20.48**	43.94	76.49	27.73	14.60	60.94	26.32	38.64
GShard	17.92	43.56	76.66	28.09	**15.00**	60.99	27.08	38.47
Hash	19.54	43.31	**77.56**	27.76	14.20	61.48	27.37	38.75
GMoE	19.97	**45.08**	77.49	**28.39**	**15.00**	**61.64**	**29.00**	**39.51**

Table 9: Comparison of MoE architectures under matched model parameters on downstream benchmarks using accuracy. Higher values indicate better performance.

that GMoE mitigates compounded redundancy and improves parameter efficiency.

Table 9 reports downstream benchmark performance under the same matched-parameter setting. Under this setting, GMoE consistently achieves either the top-1 or top-2 accuracy across all evaluated benchmarks. These results further demonstrate that the parameter efficiency of GMoE translates into effective downstream generalization, rather than being limited to language modeling performance.

D.2 Matched Activated Parameters

To further examine whether the performance gains of GMoE stem merely from increased per-token compute (i.e., higher activated parameters), we conducted a controlled comparison under a matched activated parameter setting. We selected two strong baselines, Switch-SE and GShard, and upscaled their FFN inner hidden size to 4,096. This adjustment aligns their activated parameters with those of GMoE-Max.

Model	# Params	# Activated Params	Avg ↓
Switch-SE	776M	181M	**29.22**
GShard	700M	181M	29.74
GMoE-Max	546M	181M	29.29

Table 10: Comparison of MoE architectures under matched activated parameters using average perplexity on the Base model. To match the activated parameters of GMoE-Max, the baselines are upscaled by increasing the FFN inner hidden size to 4,096.

As presented in Table 10, GMoE-Max achieves competitive perplexity compared to the upscaled baselines. Notably, GMoE-Max matches Switch-SE within a small margin while using only 0.70× the model parameters. Moreover, GMoE-Max attains lower perplexity than GShard with fewer parameters, using only 0.78× the model parameters. This suggests that even when the activated-parameter budget is equalized, the architectural advantages of Global Experts and logit propagation enable more efficient parameter utilization. Overall, these results indicate that the effectiveness of GMoE stems primarily from its architectural design rather than simply increased activated parameters.

E Analysis Metrics Details

In this section, we provide the formal definitions and mathematical formulations for the metrics used in Section 5.

For a set of MoE layers $\mathcal{L} = \{l_1, \ldots, l_M\}$, the routing path of token t is formally defined as:

$$\mathbf{r}_t = \left(e_t^{(l_1)}, e_t^{(l_2)}, \ldots, e_t^{(l_M)}\right), \tag{9}$$

where $e_t^{(l)} \in \{1, \ldots, N\}$ denotes the top-1 expert index at layer l. Let $\{\bar{\mathbf{r}}_k\}_{k=1}^{K}$ denote the set of unique paths observed in $\{\mathbf{r}_t\}_{t=1}^{T}$, and let $n(\bar{\mathbf{r}}_k)$ be the number of tokens assigned to $\bar{\mathbf{r}}_k$. This yields the empirical distribution:

$$p(\bar{\mathbf{r}}_k) = \frac{n(\bar{\mathbf{r}}_k)}{\sum_{k'=1}^{K} n(\bar{\mathbf{r}}_{k'})}, \tag{10}$$

and the corresponding path entropy:

$$H_{\text{path}} = -\sum_{k=1}^{K} p(\bar{\mathbf{r}}_k) \log_2 p(\bar{\mathbf{r}}_k), \tag{11}$$

where K is the number of unique paths.

To evaluate the semantic consistency of routing paths, we utilize three complementary metrics. Let $\mathcal{C}_k$ be the set of tokens belonging to the k-th path ($k = 1, \ldots, K$), with $n_k = |\mathcal{C}_k|$, and let $\boldsymbol{\mu}_k$ denote the centroid of cluster $\mathcal{C}_k$. The global centroid is denoted by $\boldsymbol{\mu}$.

kNN Purity. This metric measures the local label consistency of clusters. For a sample $\mathbf{x}_i$ with path label y_i, purity is the fraction of its k nearest neighbors sharing the same label:

$$\text{Purity} = \frac{1}{T}\sum_{i=1}^{T}\frac{1}{k}\sum_{\mathbf{x}_j \in \mathcal{N}_k(\mathbf{x}_i)} \mathbb{I}(y_i = y_j), \quad (12)$$

where $\mathcal{N}_k(\mathbf{x}_i)$ denotes the set of k nearest neighbors.

Inter/Intra Ratio. We define the intra-cluster distance $\mathcal{D}_{\text{intra}}$ and inter-cluster distance $\mathcal{D}_{\text{inter}}$ using cosine distance:

$$\mathcal{D}_{\text{intra}} = \frac{1}{K}\sum_{k=1}^{K}\frac{1}{n_k}\sum_{\mathbf{x}\in\mathcal{C}_k}\left(1 - \cos(\mathbf{x}, \boldsymbol{\mu}_k)\right), \quad (13)$$

$$\mathcal{D}_{\text{inter}} = \frac{2}{K(K-1)}\sum_{1\le i<j\le K}\left(1 - \cos(\boldsymbol{\mu}_i, \boldsymbol{\mu}_j)\right). \quad (14)$$

We report the ratio $\mathcal{D}_{\text{inter}}/\mathcal{D}_{\text{intra}}$ as a single summary statistic.

Calinski-Harabasz Index (CHI). CHI is defined as:

$$\text{CHI} = \frac{\text{Tr}(\mathbf{B}_K)}{\text{Tr}(\mathbf{W}_K)} \times \frac{T-K}{K-1}, \quad (15)$$

where T is the total number of samples, $\mathbf{B}_K = \sum_{k=1}^{K} n_k(\boldsymbol{\mu}_k - \boldsymbol{\mu})(\boldsymbol{\mu}_k - \boldsymbol{\mu})^T$ is the between-group dispersion matrix, and $\mathbf{W}_K = \sum_{k=1}^{K}\sum_{\mathbf{x}\in\mathcal{C}_k}(\mathbf{x} - \boldsymbol{\mu}_k)(\mathbf{x} - \boldsymbol{\mu}_k)^T$ is the within-group dispersion matrix.

F Load Balancing Analysis

To further examine the scalability of GMoE from the perspective of expert utilization, we report additional load balancing analysis across Small, Base, and Large model configurations. Since GMoE shares a set of global experts across layers, an important question is whether expert utilization remains well-balanced as the model scale increases. To study this, we quantify how uniformly routing probability is distributed across experts.

Model	Load Balance Entropy ↑
Small Model	
Switch	2.32
Switch-SE	2.22
GShard	2.21
GMoE	**2.56**
Base Model	
Switch	2.36
Switch-SE	2.48
GShard	2.27
GMoE	**2.60**
Large Model	
Switch	2.50
Switch-SE	2.62
GShard	2.58
GMoE	**2.69**

Table 11: Comparison of Load Balance Entropy across Small, Base, and Large models. Higher values indicate better expert utilization, and the theoretical maximum with 16 experts is approximately 2.77.

We quantify expert utilization using Load Balance Entropy. For each MoE layer l, we first compute the average routing probability of expert i across tokens:

$$\bar{p}_i^{(l)} = \frac{1}{T}\sum_{t=1}^{T} p_{t,i}^{(l)}, \quad (16)$$

where $p_{t,i}^{(l)}$ denotes the routing probability assigned to expert i for token t at layer l. We then define the layer-wise load balance entropy as

$$H^{(l)} = -\sum_{i=1}^{N}\bar{p}_i^{(l)}\ln\bar{p}_i^{(l)}. \quad (17)$$

Finally, we report the average across MoE layers:

$$H_{\text{LB}} = \frac{1}{|L|}\sum_{l\in L} H^{(l)}. \quad (18)$$

Higher values indicate better expert utilization. In our setting with 16 experts, the theoretical maximum for each layer is approximately $\ln(16) \approx 2.77$. For GMoE, this entropy is computed over the Global Experts only, excluding the always-on Local Expert.

Table 11 shows that GMoE consistently achieves higher Load Balance Entropy than the baselines across all model sizes. Notably, in the Large model setting, GMoE reaches an entropy of 2.69, which is

close to the theoretical maximum of approximately 2.77. These results suggest that GMoE maintains well-balanced expert utilization as the model scales within the evaluated range.

G Expert Count Scaling

In this section, we clarify a fundamental scaling difference between conventional MoE architectures and GMoE. In conventional MoE architectures, experts are instantiated independently at each layer. Therefore, if each layer contains $E_{\text{per-layer}}$ experts and the model has L layers, the total number of experts scales as

$$N_{\text{total}} = L \times E_{\text{per-layer}}. \tag{19}$$

In contrast, GMoE decomposes the expert module into layer-specific Local Experts and Global Experts shared across layers. Under this design, the total number of experts scales as

$$N_{\text{total}} = L \times E_{\text{local}} + E_{\text{global}}, \tag{20}$$

where E_{local} denotes the number of Local Experts assigned to each layer, and E_{global} denotes the number of Global Experts shared across all layers.

This design leads to a fundamentally different scaling behavior. In conventional MoE architectures, the total number of experts grows proportionally with both the number of layers and the number of experts per layer. In GMoE, by contrast, only the Local Experts scale with the number of layers, while the Global Experts are instantiated once and reused across layers. As a result, GMoE can increase expert capacity while requiring substantially fewer model parameters.

For example, when $L = 100$, $E_{\text{local}} = 1$, and $E_{\text{global}} = 1{,}000$, GMoE contains a total of $1{,}100$ experts. Under a conventional MoE design with $1{,}000$ experts at each layer, the total number of experts would instead be $100{,}000$. This example illustrates how sharing Global Experts across layers changes the scaling property of the architecture.