Low-Rank Routing for Enhancing Expert Specialization in Mixture-of-Experts Pretraining

Anonymous ACL submission

Abstract

Sparse Mixture-of-Experts (MoE) architectures have become a widely used approach for scaling large language models through sparse expert activation. However, mainstream MoE models primarily rely on load balancing to prevent expert collapse, without explicitly encouraging expert specialization, which can hinder pretraining performance. While recent work improves expert specialization through architectural modifications or orthogonality regularization, these methods require model redesign or create optimization conflicts with the language modeling objective. To address these issues, we propose Low-rank Routing (LRouting), a routing strategy that incorporates the rank statistics of expert-assigned hidden states into Top-k routing to promote expert specialization. We evaluate LRouting in MoE pretraining across multiple model scales. Experiments on downstream benchmarks show that LRouting consistently outperforms competitive baselines. Further analyses demonstrate that LRouting strengthens expert specialization while maintaining balanced expert utilization.

1 Introduction

Sparse Mixture-of-Experts (MoE) architectures have become a compelling paradigm for scaling modern large language models, as they increase total model capacity while activating only a small subset of experts for each input (Lepikhin et al., 2021; Fedus et al., 2022; Du et al., 2022). To improve parameter utilization, existing MoE models primarily focus on load balancing, which prevents expert collapse and ensures balanced expert utilization (Shazeer et al., 2017; Du et al., 2022). However, load balancing does not explicitly guide different experts toward distinct representation patterns, which may limit expert specialization and weaken pretraining performance (Guo et al., 2026; Omi et al., 2025; Hu et al., 2026).

Recent studies have attempted to improve expert specialization from different perspectives. One line of work modifies the MoE architecture itself: DeepSeekMoE introduces fine-grained and shared experts (Dai et al., 2024), while HMoE assigns heterogeneous capacities to different experts (Wang et al., 2025a). Another line of work incorporates auxiliary regularization losses to explicitly promote expert orthogonality (Guo et al., 2026; Omi et al., 2025; Hu et al., 2026). However, architecture-level modifications may require redesigning the MoE structure, and regularization losses may introduce potential optimization conflicts with the primary language modeling objective.

Drawing inspiration from recent work investigating MoE specialization through the geometric properties of hidden states, we depart from prior approaches relying on architectural adjustments or auxiliary regularization. Our core insight is that expert specialization can be strengthened by leveraging the low-rank structure of expert-specific hidden-state representations. This perspective is grounded in our empirical observation that experts processing lower-rank hidden states exhibit less co-activation with other experts, as validated in Figure 1. Our further theoretical analysis reveals that subspaces spanned by such low-rank representations tend to overlap less with those of other experts. Combined with existing evidence that expert diversity alleviates redundant co-activation and fosters natural specialization, these findings motivate using hidden-state rank as a routing signal to promote targeted expert specialization.

Building upon these observations, we propose Low-rank Routing (LRouting), a routing strategy that monitors rank statistics of expert-assigned hidden states and injects expert-specific bonuses into routing scores. This bonus incentivizes tokens to be routed toward experts whose associated hidden states form low-rank structures, thereby amplifying expert specialization without architec-

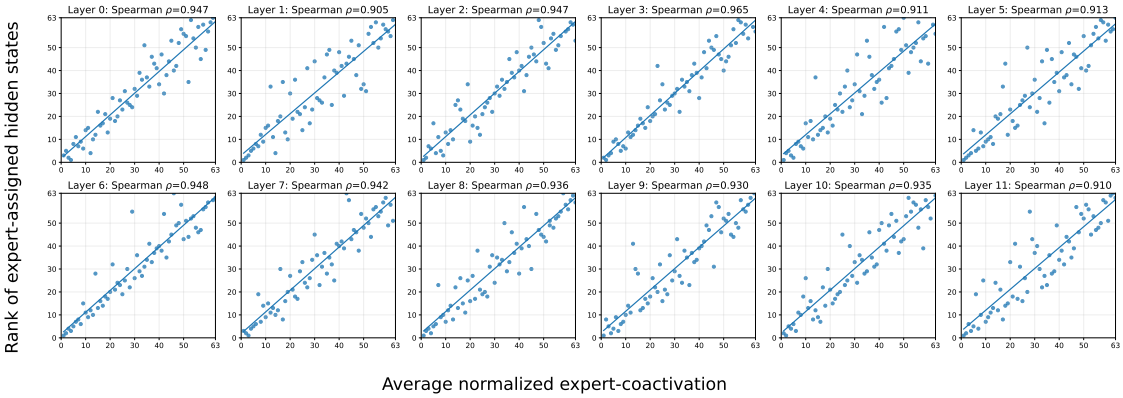


Figure 1: Spearman's rank correlation between average normalized expert co-activation and the Gram-matrix rank of expert-assigned hidden states in a 2B DeepSeek-style MoE model pretrained with auxiliary-loss-free load balancing. Each subplot corresponds to one of the 12 MoE layers, and each point represents one of the 64 routed experts.

tural modifications or extra specialization-oriented losses. To validate the efficacy of LRouting, we perform extensive pre-training experiments on DeepSeek-style MoE architectures across multiple model scales. Evaluations on downstream benchmarks demonstrate that LRouting consistently surpasses baseline methods. Further analyses verify that our approach improves expert specialization while preserving balanced expert utilization.

Contributions. Our contributions are threefold: 1) We study expert specialization in MoE pretraining from the perspective of low-rank hidden-state geometry, supported by both empirical observations and theoretical analysis; 2) We propose LRouting, a routing strategy that leverages the rank statistics of expert-assigned hidden states to promote expert specialization; 3) We validate LRouting across multiple MoE model scales, showing improved downstream performance and stronger expert specialization over competitive baselines.

Notations. We use lowercase boldface letters to denote vectors and uppercase boldface letters to denote matrices. For any positive integer n, we denote the set $\{1, 2, \ldots, n\}$ by $[n]$. For a vector or matrix, $\|\cdot\|$ denotes its norm. For a matrix $\mathbf{A}$, $\mathbf{A}^\top$ denotes its transpose, $\mathrm{rank}(\mathbf{A})$ denotes its rank, and $\mathrm{tr}(\mathbf{A})$ denotes its trace.

2 Related Work

MoE routing and expert specialization. Sparse Mixture-of-Experts (MoE) increases model capacity with conditional computation by routing each token to a small subset of feed-forward experts. Sparsely-gated MoE, GShard, and Switch Transformer established top-k routing with auxiliary load-balancing losses for stable scaling (Shazeer et al., 2017; Lepikhin et al., 2021; Fedus et al., 2022). However, balanced utilization alone does not guarantee expert specialization. One line modifies the MoE architecture: DeepSeekMoE introduces fine-grained and shared experts to reduce routed-expert redundancy, and HMoE assigns heterogeneous expert capacities (Dai et al., 2024; Wang et al., 2025a). Another line directly regularizes specialization. AES-MoE adds orthogonality and routing-variance losses, and recent intra-/cross-layer regularization encourages complementary expert activations and coherent expert pathways across layers (Guo et al., 2026; Hu et al., 2026). However, these approaches often require architectural redesign or extra objectives that interact with language-modeling optimization.

Low-rank geometry in LLM training. Recent analyses suggest that MoE routing patterns are closely tied to hidden-state geometry rather than only to human-interpretable domains (Wang et al., 2026; Ahrac et al., 2026; Liu, 2026). Along this line, L2R (Yang et al., 2026) performs expert assignment in a low-rank latent routing space with Lipschitz-controlled scoring. Low-rank structures have also been explored in LLM training, for example, through cross-layer activation residuals for activation reconstruction (Kong et al., 2025). LRouting is complementary to these works: it does not impose an explicit low-rank architecture, but instead uses the rank statistics of expert-assigned hidden states as a routing bonus, thereby promoting specialization while preserving the standard MoE backbone and load-balancing mechanism.

3 Preliminaries

In this section, we introduce the MoE pretraining preliminaries used throughout the paper, including top-k routing and load-balancing mechanisms.

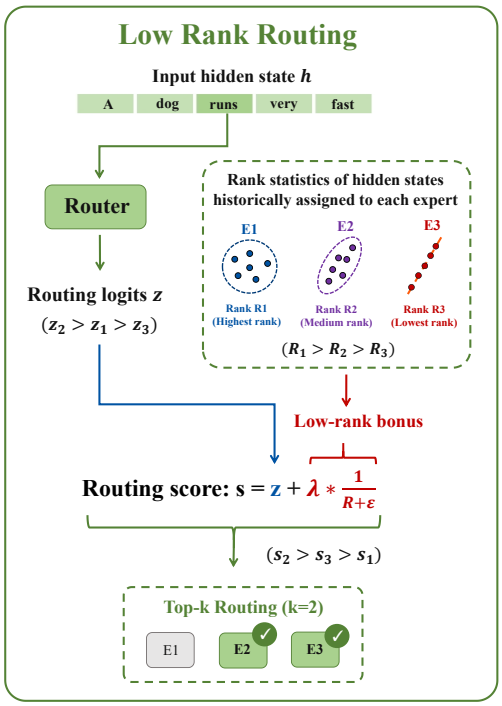


Figure 2: Overview of LRouting. LRouting jointly considers routing logits and hidden-state rank in Top-k routing, favoring experts whose historically assigned hidden states exhibit lower rank.

3.1 Top-k routing MoE

We first introduce the sparsely activated MoE architecture commonly used in modern language models. Consider a standard MoE layer with N experts $\mathcal{E} = \{E_1, E_2, \ldots, E_N\}$ and n input tokens. Each expert E_j is implemented as a feed-forward network (FFN). Given an input token with a d-dimensional hidden state $\mathbf{h} \in \mathbb{R}^d$, a router computes the routing logits as $\mathbf{z} = \mathbf{W}_g\mathbf{h}$, where $\mathbf{W}_g \in \mathbb{R}^{N\times d}$ is a learnable parameter. The router then selects the top-k experts according to these logits, denoted by $\mathcal{I}_k(\mathbf{h}) = \mathrm{TopK}(\mathbf{z}, k)$. Only the selected experts are activated for this token, and their mixing weights are obtained by applying a softmax over the selected logits:

$$g_j(\mathbf{h}) = \begin{cases} \dfrac{\exp(z_j)}{\sum_{j'\in\mathcal{I}_k(\mathbf{h})}\exp(z_{j'})}, & j \in \mathcal{I}_k(\mathbf{h}), \\ 0, & j \notin \mathcal{I}_k(\mathbf{h}). \end{cases} \tag{1}$$

The final output of $\mathbf{h}$ is thus the sparsely weighted sum of the output of its k experts:

$$\mathbf{y} = \sum_{j\in\mathcal{I}_k(\mathbf{h})} g_j(\mathbf{h})E_j(\mathbf{h}). \tag{2}$$

3.2 Load Balancing in MoE

A key challenge in Top-k routing MoE is load imbalance (Shazeer et al., 2017; Wang et al., 2024). Without explicit balancing constraints, the router may concentrate tokens on a small subset of experts, causing the remaining experts to receive few updates and become under-utilized. This phenomenon, often referred to as expert starvation, reduces the effective capacity of the MoE model and can lead to inefficient training. To mitigate this issue, existing MoE models commonly adopt load-balancing mechanisms.

One widely used approach is to add an auxiliary load-balancing loss to the training objective (Shazeer et al., 2017; Fedus et al., 2022). This loss encourages both the actual token assignments and the router probability mass to be close to uniform: $\mathcal{L}_{\text{aux}} := N\sum_{j=1}^{N} f_j P_j$, where f_j denotes the fraction of tokens routed to expert E_j, and P_j denotes the average routing probability assigned to E_j over a batch. Based on this auxiliary loss, the final pretraining objective is written as $\mathcal{L} = \mathcal{L}_{\text{lm}} + \alpha\mathcal{L}_{\text{aux}}$, where $\mathcal{L}_{\text{lm}}$ and $\mathcal{L}_{\text{aux}}$ denote the language modeling loss and the load-balancing loss, respectively. The hyperparameter α controls the strength of load balancing.

While load-balancing mechanisms regulate the overall utilization of experts, balanced utilization alone does not determine what kinds of hidden states each expert should process, and thus limits expert specialization. Since hidden states are representation vectors of tokens in a high-dimensional space, differences in expert specialization can be reflected by the geometric structure of the hidden-state sets assigned to different experts. This motivates us to study expert-assigned hidden-state geometry and use it to design a routing strategy that enhances expert specialization.

4 Methodology

In this section, we first present both empirical and theoretical motivations for utilizing the low-rank structure of expert-assigned hidden states in enhancing expert specialization, and then introduce our proposed expert-specialized routing strategy based on these motivations.

4.1 Motivation

Empirical Observation. In the following, we conduct an empirical study to investigate whether the geometric structure of expert-assigned hidden

states is associated with expert specialization during MoE pretraining.

Below, we analyze two expert-level quantities within each MoE layer. First, following recent works (Wang et al., 2025b; Zhang et al., 2025), we study co-activation between experts to quantify the degree of expert specialization. For layer l and expert j, let $\mathcal{T}_{l,j}$ denote the set of token indices routed to this expert in this layer. Based on it, we define the normalized co-activation between two experts j and j' as the Jaccard similarity between their assigned token sets: $\mathrm{CoAct}_l(j, j') := \frac{|\mathcal{T}_{l,j} \cap \mathcal{T}_{l,j'}|}{|\mathcal{T}_{l,j} \cup \mathcal{T}_{l,j'}|}$. We then compute the average normalized expert co-activation for each expert: $\mathrm{AvgCoAct}_{l,j} := \frac{1}{N-1} \sum_{j \neq j'} \mathrm{CoAct}_l(j, j')$. A smaller value indicates that expert j shares fewer routed tokens with other experts, suggesting lower routing redundancy and therefore stronger expert specialization.

Second, we characterize the geometric structure of hidden states assigned to each expert. For layer l and expert j, let $\mathbf{H}_{l,j} \in \mathbb{R}^{n_{l,j} \times d}$ denote the matrix of token hidden states routed to expert j, where each row corresponds to one assigned token representation and $n_{l,j}$ is the number of tokens routed to this expert. We define the corresponding Gram matrix as $\mathbf{G}_{l,j} = \mathbf{H}_{l,j}^\top \mathbf{H}_{l,j}$. The rank of $\mathbf{G}_{l,j}$ measures the dimensionality of the subspace spanned by the hidden states assigned to expert j. A lower rank suggests that the hidden states routed to an expert lie in a lower-dimensional subspace and are more concentrated along a small number of dominant directions, indicating stronger directional similarity among the assigned hidden states.

To study the relationship between expert specialization and the low-rank structure of expert-assigned hidden states, we train a 2B-parameter DeepSeek-style MoE model with 12 MoE layers and 64 routed experts on 2B tokens, using an auxiliary-loss-free load-balancing strategy to maintain balanced expert utilization (Wang et al., 2024). During pretraining, we measure the specialization degree of each expert using the average normalized expert co-activation score $\mathrm{AvgCoAct}_{l,j}$. We also construct the Gram matrix $\mathbf{G}_{l,j}^{(t)}$ from $\mathbf{H}_{l,j}$ routed to expert j at iteration t, and maintain an exponential moving average of its rank, denoted by $\bar{r}(\mathbf{G}_{l,j})$. We then compute the Spearman's rank correlation between $\mathrm{AvgCoAct}_{l,j}$ and $\bar{r}(\mathbf{G}_{l,j})$. As shown in Figure 1, layers 0-11 exhibit consistently strong positive Spearman correlations, with ρ ranging from 0.905 to 0.965. These results provide empirical evidence that expert specialization is closely associated with the low-rank structure of expert-assigned hidden states.

Theoretical Motivation. We further provide a theoretical interpretation of why low-rank hidden-state structure can serve as an informative signal for expert-specialized routing. Let $S = \mathrm{span}(\mathbf{H})$ denote the subspace spanned by the hidden-state matrix $\mathbf{H}$. The rank of $\mathbf{H}$ determines the subspace dimension, i.e., $\mathrm{rank}(\mathbf{H}) = \dim(S)$. For two subspaces S and T, let $\mathbf{P}_S$ and $\mathbf{P}_T$ denote their orthogonal projection matrices, respectively. We measure the overlap between these subspaces by $\mathrm{tr}(\mathbf{P}_S \mathbf{P}_T)$, which equals the sum of the squared cosines of their principal angles. A smaller value indicates that the two subspaces are more nearly orthogonal.

The following theorem shows that a lower-dimensional hidden-state subspace has smaller expected overlap with a random subspace, providing a geometric basis for stronger expert specialization.

Theorem 4.1. *Let $S \subset \mathbb{R}^d$ be a fixed subspace spanned by a rank-r set of expert-assigned hidden states, with $\dim(S) = r$. Let $T \subset \mathbb{R}^d$ be a random q-dimensional subspace constructed by sampling q independent Gaussian vectors in $\mathbb{R}^d$ and orthonormalizing them. Then the expected projection overlap between S and T is*

$$\mathbb{E}\left[\mathrm{tr}(\mathbf{P}_S \mathbf{P}_T)\right] = rq/d. \tag{3}$$

Remark 4.2. *The theorem shows that when $r \ll d$, the expected overlap $\frac{rq}{d}$ is small, meaning that the fixed subspace is nearly orthogonal in expectation to the random subspace.*

This theoretical insight provides a geometric interpretation of the empirical observation above: experts whose assigned hidden states form lower-rank structures are expected to be more distinguishable from other experts, as indicated by lower overlap between expert-assigned hidden-state subspaces. Due to space limitations, we defer the proof of the theorem to Appendix A.

4.2 Routing with Low-Rank Hidden-State Structure

Based on the above motivations, we formally introduce Low-rank Routing (LRouting), a strategy for enhancing expert specialization in MoE pretraining. Figure 2 provides an overview of LRouting.

Recall that in standard Top-k routing, the router computes a logit $z_{i,j}^{(t,l)}$ for token $i \in [n]$ and expert $j \in [N]$ at layer l and iteration t, and selects the k

experts with the largest routing scores. To promote expert specialization, we augment the original routing logit with an expert-specific low-rank bonus term before performing Top-k selection.

Formally, let $\mathbf{H}_{l,j}^{(t)} \in \mathbb{R}^{n_{l,j}^{(t)} \times d}$ denote the hidden states assigned to expert j in layer l at iteration t, where $n_{l,j}^{(t)}$ is the number of tokens routed to expert j. We note that directly estimating the rank in the original hidden size can be computationally expensive. To reduce this cost, we first project the hidden states into a lower-dimensional random subspace. For each layer l, we initialize a fixed layer-wise random projection matrix $\mathbf{U}_l \in \mathbb{R}^{d \times d'}$. The projected hidden-state matrix is defined as $\widetilde{\mathbf{H}}_{l,j}^{(t)} = \mathbf{H}_{l,j}^{(t)} \mathbf{U}_l$. The corresponding projected Gram matrix is then constructed as

$$\widetilde{\mathbf{G}}_{l,j}^{(t)} = (\widetilde{\mathbf{H}}_{l,j}^{(t)})^\top \widetilde{\mathbf{H}}_{l,j}^{(t)}. \tag{4}$$

By the Johnson–Lindenstrauss lemma (Dasgupta and Gupta, 2003), random projections approximately preserve pairwise distances among hidden states with high probability. Hence, we use random projection as a computationally efficient proxy for rank estimation in the original hidden space.

To characterize the effective dimensionality of expert-assigned hidden states, we use stable rank as a smooth surrogate for exact rank, since exact rank is sensitive to small singular values. The stable rank of the projected Gram matrix is defined as $\mathrm{srank}(\widetilde{\mathbf{G}}_{l,j}^{(t)}) := \frac{\|\widetilde{\mathbf{G}}_{l,j}^{(t)}\|_F^2}{\|\widetilde{\mathbf{G}}_{l,j}^{(t)}\|_2^2}$, where $\|\cdot\|_2$ denotes the spectral norm and $\|\cdot\|_F$ denotes the Frobenius norm. The spectral norm can be efficiently estimated using power iteration. A lower stable rank indicates that the hidden states routed to expert j are concentrated in fewer dominant directions.

To reduce the variance of batch-level rank estimates, we maintain an exponential moving average (EMA) of the projected stable rank for each expert:

$$R_{l,j}^{(t)} = \beta R_{l,j}^{(t-1)} + (1-\beta)\mathrm{srank}(\widetilde{\mathbf{G}}_{l,j}^{(t)}), \tag{5}$$

where $\beta \in [0,1)$ is the EMA coefficient. The EMA statistic provides a smoother estimate of the historical information associated with each expert.

Therefore, at iteration t, we use the EMA statistic $R_{l,j}^{(t-1)}$ to compute an expert-specific low-rank routing bonus term. Given the original router logit $z_{i,j}^{(t,l)}$ for token i and expert j, we define the modified routing score as

$$s_{i,j}^{(t,l)} := z_{i,j}^{(t,l)} + \lambda \cdot \frac{1}{R_{l,j}^{(t-1)} + \epsilon}, \tag{6}$$

Algorithm 1 LRouting for Layer l

Require: Input hidden states $\mathbf{H} = \{\mathbf{h}_i\}_{i \in [n]}$, gating weights $\mathbf{W}_g^{(t,l)}$, low-rank hyperparameter λ, rank statistics $\{R_{l,1}^{(t-1)}, \ldots, R_{l,N}^{(t-1)}\}$

Ensure: Top-k routing result $\mathcal{I}_k^{(t,l)}(\mathbf{h}_i), i \in [n]$

Compute routing logits: $\mathbf{z}^{(t,l)} = \mathbf{W}_g^{(t,l)} \cdot \mathbf{H}$
Compute routing score for each token i:
$s_{i,j}^{(t,l)} = z_{i,j}^{(t,l)} + \lambda \cdot \frac{1}{R_{l,j}^{(t-1)}+\epsilon}, \forall j \in [N]$
Select top-k experts for each token i:
$\mathcal{I}_k^{(t,l)}(\mathbf{h}_i) \leftarrow \mathrm{TopK}(\mathbf{s}_i^{(t,l)}, k)$
for $j \in [N]$ **do**
  Construct $\widetilde{\mathbf{G}}_{l,j}^{(t)}$ defined in Eq. (4)
  Update rank statistics:
  $R_{l,j}^{(t)} \leftarrow \beta R_{l,j}^{(t-1)} + (1-\beta)\mathrm{srank}(\widetilde{\mathbf{G}}_{l,j}^{(t)})$
end for
return $\mathcal{I}_k^{(t,l)}(\mathbf{h})$

Method	Per-layer Routing FLOPs
Aux-Loss-Free	$2ndN$
LRouting-SR	$2ndN + 2ndd' + 2\big(nk + (\tau+2)N\big)d'^2$
LRouting-PR	$2ndN + 2ndd' + 2(nk+N)d'^2$

Table 1: Per-layer routing FLOPs of different methods. τ denotes the number of power iterations.

where λ is a hyper-parameter controlling the strength of the low-rank bonus term and ϵ is a small constant for numerical stability. The Top-k experts for token i are then selected according to the modified scores: $\mathcal{I}_k^{(t,l)}(\mathbf{h}_i) = \mathrm{TopK}(\mathbf{s}_i^{(t,l)}, k)$.

After routing the current batch, we collect the hidden states assigned to each expert, compute the projected stable rank, and update the EMA statistic for use in subsequent training iterations. The overall workflow of LRouting is presented in Alg. 1.

Efficient variant with participation ratio. As an optional efficient variant, we consider the participation ratio (PR) (Gao et al., 2017) as a lightweight alternative to stable rank. The participation ratio is defined as $\mathrm{PR}(\widetilde{\mathbf{G}}_{l,j}^{(t)}) = \frac{\mathrm{tr}(\widetilde{\mathbf{G}}_{l,j}^{(t)})^2}{\|\widetilde{\mathbf{G}}_{l,j}^{(t)}\|_F^2}$. Unlike stable rank, this variant reduces the overhead of rank estimation by avoiding spectral norm computation.

Analysis on computation cost. Table 1 compares the per-layer routing FLOPs of our proposed LRouting variants, LRouting-SR (Stable Rank) and LRouting-PR (Participation Ratio), against the Aux-Loss-Free load-balancing strategy. All methods share the same router-logit cost $2ndN$. LRout-

ing adds only a one-time random projection cost $2ndd'$ and rank estimation on low-dimensional projected Gram matrices. Since $d' \ll d$, the additional overhead remains moderate.

Discussion. LRouting has several practical benefits: 1) **it introduces no explicit regularization losses**, thereby reducing potential optimization conflicts with the primary language modeling objective; 2) **it uses only historical expert-assignment statistics before the current routing step**, introducing no look-ahead information and thus respecting the causal constraint of autoregressive language modeling; 3) **the low-rank bonus term is expert-level rather than token-specific.** During inference, we simply reuse the final bonus values and add them to the routing logits, without recomputing rank statistics. This ensures training-inference consistency with negligible inference overhead.

5 Experiments

5.1 Experimental Setup

Training settings. We implement all experiments in the Megatron-LM framework (Shoeybi et al., 2019). We conduct experiments on a DeepSeek-style MoE architecture (Dai et al., 2024), where all Transformer layers are equipped with MoE FFN modules. Each MoE layer contains two shared experts. We consider two model configurations with different parameter scales and training budgets. We train the 2B MoE model on 50B tokens, and the larger 8B MoE model on 100B tokens. The model configurations are provided in Table 4 of Appendix B. For our proposed method, we set $\beta = 0.9$, $d' = 256$, and $\lambda = 0.1$, with auxiliary-loss-free load balancing (Wang et al., 2024) to maintain balanced expert utilization.

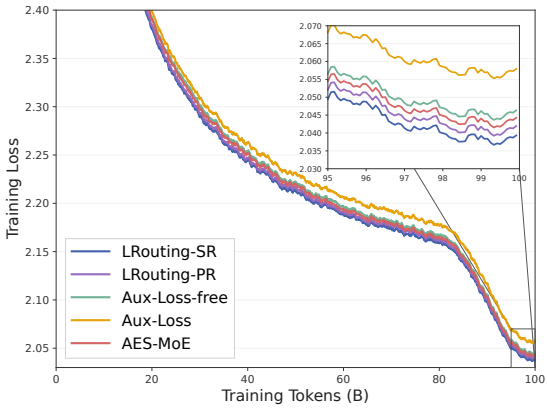
Baselines. We compare our proposed LRouting method with three baselines: 1) Auxiliary load-balancing loss (Fedus et al., 2022); 2) Aux-loss-free load balancing (Wang et al., 2024); and 3) AES-MoE (Guo et al., 2026), short for *Advancing Expert Specialization for Better MoE*, which introduces an orthogonality loss and a routing-logit variance loss to promote expert specialization.

Evaluation settings. We evaluate the checkpoint from the final pretraining iteration on both validation perplexity and downstream performance. For validation perplexity, we use the validation sets of C-Eval (Huang et al., 2023), MMLU (Hendrycks et al., 2020), BigBench (Suzgun et al., 2023), WPLC (Ge et al., 2021), and LAMBADA (Paperno et al., 2016). For downstream evaluation, we report results on C-Eval, MMLU, BBH, WPLC, LAMBADA, GPQA (Rein et al., 2023), and PIQA (Bisk et al., 2020).

Due to the page limit, more detailed experimental settings are provided in the Appendix B.

5.2 Evaluations

Figure 3 compares the pretraining loss curves of our proposed LRouting methods with the baselines during pretraining of the 8B model on 100B tokens. The curves are smoothed for visualization. Both LRouting-SR and LRouting-PR consistently achieve lower training loss than the baseline methods throughout pretraining. Table 2 and Table 3 report downstream performance and validation perplexity evaluated on the final checkpoint. For downstream evaluation, LRouting achieves the best average accuracy at both scales: LRouting-PR reaches 29.65% on the 2B model, while LRouting-SR reaches 35.15% on the 8B model. Compared with AES-MoE, the strongest baseline, these results correspond to relative improvements of 1.02% and 4.06%, respectively. These results show that LRouting consistently improves downstream performance across model scales and benchmarks. Moreover, LRouting achieves the lowest average validation perplexity among all methods across several datasets and both model scales.



8B model pretrained on 100B tokens.

Figure 3: Training loss curves of LRouting methods and baselines during pretraining.

These results show that LRouting improves MoE pretraining performance by enhancing expert specialization, leveraging the low-rank structure of expert-assigned hidden states to adjust routing scores while avoiding the additional regularization losses used in AES-MoE and thereby preserving the primary language modeling objective.

Model Scale	Method	C-Eval (5-shot)	MMLU (5-shot)	BBH (3-shot)	WPLC (0-shot)	LAMBADA (0-shot)	GPQA (5-shot)	PIQA (10-shot)	Avg.	Diff.
Base (2B)	Aux-Loss-Free	25.46	25.52	23.51	5.53	46.28	25.22	49.45	28.71	0.00
	Aux-Loss	27.36	25.36	24.78	5.72	45.61	20.98	49.45	28.47	-0.24
	AES-MoE	27.06	26.20	24.42	5.41	46.80	25.89	49.67	29.35	+0.64
	LRouting-PR	27.72	25.81	**25.11**	**6.11**	46.41	**26.56**	49.83	**29.65**	**+0.94**
	LRouting-SR	**28.03**	**26.35**	23.50	5.60	**46.84**	25.89	**50.48**	29.53	+0.82
Large (8B)	Aux-Loss-Free	28.23	28.33	31.28	12.48	58.57	25.44	49.61	33.42	0.00
	Aux-Loss	26.75	29.84	30.43	12.56	58.26	25.00	50.10	33.28	-0.14
	AES-MoE	28.26	29.91	30.89	12.50	58.37	26.56	49.94	33.78	+0.36
	LRouting-PR	29.95	29.02	**31.37**	12.46	59.15	**26.78**	50.43	34.17	+0.75
	LRouting-SR	**31.42**	**32.97**	30.98	**13.24**	**59.77**	26.33	**51.36**	**35.15**	**+1.73**

Table 2: Downstream evaluation results. Parenthesized values after benchmark names indicate the number of evaluation shots. Avg. denotes the arithmetic mean over all listed benchmarks, and Diff. denotes the difference in Avg. relative to Aux-Loss-Free.

Model Scale	Method	Training Loss ↓	Spec. Experts	C-Eval (5) PPL ↓	MMLU (5) PPL ↓	BigBench (3) PPL ↓	WPLC (0) PPL ↓	LAMBADA (0) PPL ↓	Val PPL Avg. ↓
Base (2B)	Aux-Loss-Free	2.230	✗	4.230	4.061	8.443	80.05	9.563	21.27 (0.00)
	Aux-Loss	2.238	✗	4.190	4.081	8.758	83.39	9.512	21.99 (+0.72)
	AES-MoE	2.229	✓	4.168	4.054	8.437	78.30	9.195	20.83 (-0.44)
	LRouting-PR	**2.225**	✓	**4.128**	4.067	8.438	**76.84**	**8.391**	**20.37 (-0.90)**
	LRouting-SR	**2.224**	✓	4.140	**4.051**	**7.335**	78.87	9.110	20.70 (-0.57)
Large (8B)	Aux-Loss-Free	2.025	✗	4.187	4.051	8.978	52.04	6.127	15.08 (0.00)
	Aux-Loss	2.036	✗	4.141	3.973	7.863	41.86	5.218	12.61 (-2.47)
	AES-MoE	2.023	✓	4.069	3.974	7.524	39.22	5.094	11.98 (-3.10)
	LRouting-PR	**2.020**	✓	**4.060**	3.985	8.069	38.33	5.117	11.91 (-3.17)
	LRouting-SR	**2.018**	✓	4.069	**3.946**	**7.017**	**38.31**	**5.081**	**11.68 (-3.40)**

Table 3: Training loss and validation perplexity results. Lower values indicate better performance. Parenthesized numbers after benchmark names indicate the number of evaluation shots. Val PPL Avg. denotes the mean validation perplexity over five benchmarks, with the parenthesized value showing the difference relative to Aux-Loss-Free. Spec. Experts indicates whether the method explicitly encourages expert specialization.

5.3 Analysis on Expert Specialization

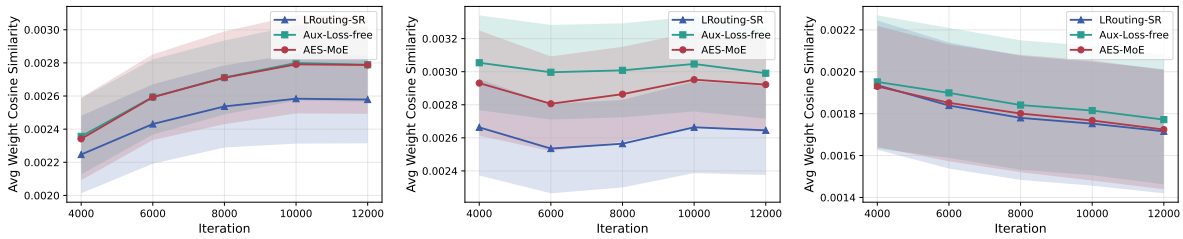
To analyze expert specialization, we measure the average cosine similarity between expert weights within the same MoE layer (Qiu et al., 2025; Lo et al., 2025; Wu et al., 2022). Specifically, for each expert, we flatten its FFN parameters into a weight vector and compute its average cosine similarity to all other experts in the same layer. A lower value indicates lower expert redundancy and thus stronger specialization (Zheng et al., 2025). We conduct this analysis on the 2B MoE model trained on 50B tokens and select Layers 3, 6, and 11 as representative MoE layers. Figure 4 shows the evolution of inter-expert weight cosine similarity throughout pretraining, where the curves report the mean across experts and the shaded regions indicate the variation in per-expert average cosine similarities. Across Layers 3, 6, and 11, LRouting consistently achieves lower similarity than the baselines, demonstrating that our method improves expert specialization throughout pretraining.

Following Liu et al. (2024), we then analyze domain expertise at Layer 6 of LRouting-SR on the 2B model using the relative gap to uniform expert load. We take the final training checkpoint and measure the token-to-expert routing distribution on downstream benchmarks. For expert j, we define this gap as $\frac{p_j - 1/N}{1/N}$, where p_j is the normalized token load and N is the number of experts. Figure 5 shows that on different benchmarks, the router assigns tokens to different groups of experts instead of using the entire expert pool uniformly. This suggests that LRouting-SR learns benchmark-conditioned expert usage patterns by assigning different domains to specialized experts.

5.4 Analysis on Load Balancing

To evaluate load balancing throughout pretraining, we follow (Wang et al., 2024) and adopt maximal violation (MaxVio), which measures the largest relative deviation of expert load from the ideally balanced load:

$$\mathrm{MaxVio} = \frac{\max_j \mathrm{Load}_j - \overline{\mathrm{Load}}}{\overline{\mathrm{Load}}}, \qquad (7)$$



Layer 3 Layer 6 Layer 11

Figure 4: Average inter-expert weight cosine similarity at different pretraining iterations.

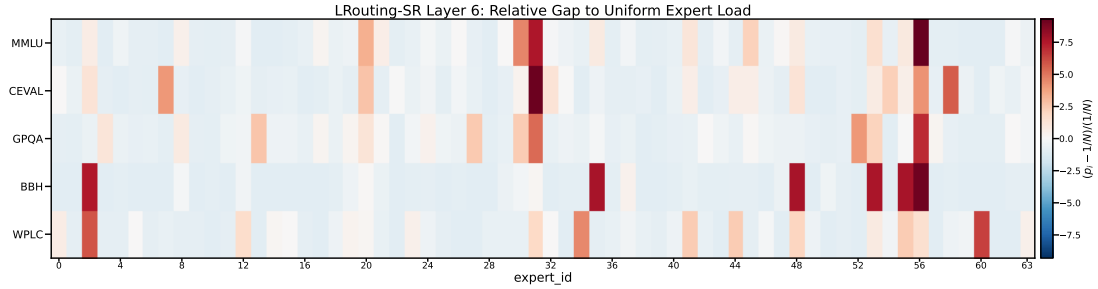


Figure 5: Relative gap to uniform expert load of LRouting-SR at Layer 6 on the 2B model across five benchmarks. Red indicates experts receiving more tokens than the uniform load, while blue indicates receiving fewer tokens.

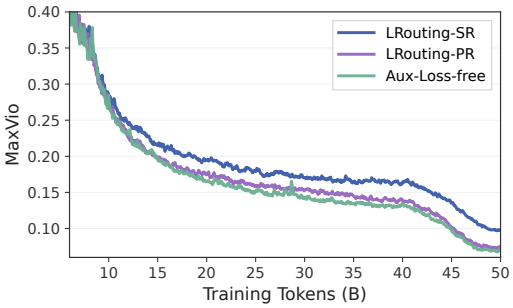


Figure 6: MaxVio curves of two LRouting methods and Aux-Loss-Free during pretraining on the 2B model.

where Load_j denotes the number of tokens assigned to expert j, and $\overline{\mathrm{Load}}$ denotes the average load per expert under perfect load balance. In our experiments, MaxVio is computed for each training batch and then averaged across all layers. A larger MaxVio indicates more severe load imbalance.

We compare the smoothed MaxVio curves of LRouting and the Aux-Loss-Free baseline during pretraining on the 2B model. As shown in Figure 6, both LRouting variants remain stable below 50%, and LRouting-SR only increases MaxVio by about 5% relative to the Aux-Loss-Free baseline. These results show that LRouting improves expert specialization while preserving load balance.

5.5 Ablation Analysis on λ

Below we investigate the effect of the hyperparameter λ used in our proposed method. We train the 2B model for 50B tokens using LRouting-SR with

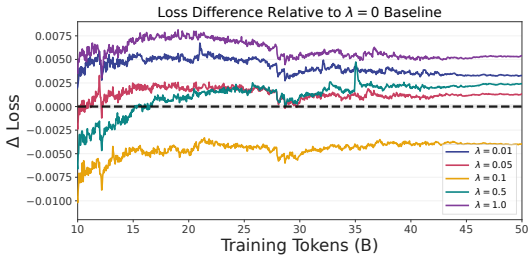


Figure 7: Ablation study on the low-rank bonus hyperparameter λ in LRouting-SR on the 2B model.

$\lambda \in \{0.01, 0.05, 0.1, 0.5, 1.0\}$. The baseline sets $\lambda = 0$, which disables the low-rank bonus, and we report the training loss difference relative to this baseline. As shown in Figure 7, $\lambda = 0.1$ achieves the best result, indicating that LRouting benefits from balancing the original routing logits with the proposed low-rank bonus.

6 Conclusion

In this paper, we investigate the relationship between hidden-state geometry and expert specialization in MoE pretraining. Motivated by our empirical and theoretical observations, we propose LRouting, a novel routing strategy that promotes expert specialization. Extensive experiments show that LRouting improves downstream performance over strong baselines. Further analyses demonstrate that LRouting strengthens expert specialization while maintaining balanced expert utilization.

Limitations

One limitation of LRouting is that it requires estimating rank statistics from the Gram matrix of expert-assigned hidden states during pretraining. Developing more efficient approximation algorithms for hidden-state rank estimation could further reduce the training overhead.

References

Sagi Ahrac, Noya Hochwald, and Mor Geva. 2026. Routers learn the geometry of their experts: Geometric coupling in sparse mixture-of-experts. *arXiv preprint arXiv:2605.12476*.

Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. 2020. PIQA: reasoning about physical commonsense in natural language. In *AAAI*, pages 7432–7439. AAAI Press.

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 *ACL (1)*, pages 1280–1297. Association for Computational Linguistics.

Sanjoy Dasgupta and Anupam Gupta. 2003. An elementary proof of a theorem of johnson and lindenstrauss. *Random Structures & Algorithms*, 22(1):60–65.

Nan Du, Yanping Huang, Andrew M. Dai, Simon Tong, Dmitry Lepikhin, Yuanzhong Xu, Maxim Krikun, Yanqi Zhou, Adams Wei Yu, Orhan Firat, Barret Zoph, Liam Fedus, Maarten P. Bosma, Zongwei Zhou, Tao Wang, Yu Emma Wang, Kellie Webster, Marie Pellat, Kevin Robinson, and 8 others. 2022. Glam: Efficient scaling of language models with mixture-of-experts. In *ICML*, Proceedings of Machine Learning Research, pages 5547–5569. PMLR.

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

Trevor Gale, Deepak Narayanan, Cliff Young, and Matei Zaharia. 2023. Megablocks: Efficient sparse training with mixture-of-experts. *Proceedings of Machine Learning and Systems*, 5:288–304.

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*.

Peiran Gao, Eric Trautmann, Byron Yu, Gopal Santhanam, Stephen Ryu, Krishna Shenoy, and Surya Ganguli. 2017. A theory of multineuronal dimensionality, dynamics and measurement. *BioRxiv*, page 214262.

Huibin Ge, Chenxi Sun, Deyi Xiong, and Qun Liu. 2021. Chinese WPLC: A chinese dataset for evaluating pretrained language models on word prediction given long-range context. In *EMNLP (1)*, pages 3770–3778. Association for Computational Linguistics.

Hongcan Guo, Haolang Lu, Guoshun Nan, Bolun Chu, Jialin Zhuang, Yuan Yang, Wenhao Che, Xinye Cao, Sicong Leng, Qimei Cui, and 1 others. 2026. Advancing expert specialization for better moe. *Advances in Neural Information Processing Systems*, 38:48767–48809.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2020. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*.

Rizhen Hu, Yuan Cao, Boao Kong, Mou Sun, and Kun Yuan. 2026. Synergistic intra- and cross-layer regularization losses for moe expert specialization. *CoRR*, abs/2602.14159.

Yuzhen Huang, Yuzhuo Bai, Zhihao Zhu, Junlei Zhang, Jinghan Zhang, Tangjun Su, Junteng Liu, Chuancheng Lv, Yikai Zhang, Yao Fu, and 1 others. 2023. C-eval: A multi-level multi-discipline chinese evaluation suite for foundation models. *Advances in neural information processing systems*, 36:62991–63010.

Boao Kong, Junzhu Liang, Yuxi Liu, Renjia Deng, and Kun Yuan. 2025. Cr-net: Scaling parameter-efficient training with cross-layer low-rank structure. *arXiv preprint arXiv:2509.18993*.

Dmitry Lepikhin, HyoukJoong Lee, Yuanzhong Xu, Dehao Chen, Orhan Firat, Yanping Huang, Maxim Krikun, Noam Shazeer, and Zhifeng Chen. 2021. Gshard: Scaling giant models with conditional computation and automatic sharding. In *ICLR*. OpenReview.net.

Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, and 1 others. 2024. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*.

Feilong Liu. 2026. Mixture-of-experts as soft clustering: A dual jacobian-pca spectral geometry perspective. *arXiv preprint arXiv:2601.11616*.

Ka Man Lo, Zeyu Huang, Zihan Qiu, Zili Wang, and Jie Fu. 2025. A closer look into mixture-of-experts in large language models. In *NAACL (Findings)*, Findings of ACL, pages 4427–4447. Association for Computational Linguistics.

Nabil Omi, Siddhartha Sen, and Ali Farhadi. 2025. Load balancing mixture of experts with similarity preserving routers. *CoRR*, abs/2506.14038.

Denis Paperno, Germán Kruszewski, Angeliki Lazaridou, Quan Ngoc Pham, Raffaella Bernardi, Sandro Pezzelle, Marco Baroni, Gemma Boleda, and Raquel Fernández. 2016. The LAMBADA dataset: Word prediction requiring a broad discourse context. In *ACL (1)*. The Association for Computer 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 *ICLR*. OpenReview.net.

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

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. 2023. GPQA: A graduate-level google-proof q&a benchmark. *CoRR*, abs/2311.12022.

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

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*.

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

Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. 2024. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063.

Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc V. Le, Ed H. Chi, Denny Zhou, and Jason Wei. 2023. Challenging big-bench tasks and whether chain-of-thought can solve them. In *ACL (Findings)*, Findings of ACL, pages 13003–13051. Association for Computational Linguistics.

An Wang, Xingwu Sun, Ruobing Xie, Shuaipeng Li, Jiaqi Zhu, Zhen Yang, Pinxue Zhao, Weidong Han, Zhanhui Kang, Di Wang, Naoaki Okazaki, and Cheng-Zhong Xu. 2025a. Hmoe: Heterogeneous mixture of experts for language modeling. In *EMNLP*, pages 21943–21957. Association for Computational Linguistics.

Lean Wang, Huazuo Gao, Chenggang Zhao, Xu Sun, and Damai Dai. 2024. Auxiliary-loss-free load balancing strategy for mixture-of-experts. *CoRR*, abs/2408.15664.

Xi Wang, Soufiane Hayou, and Eric T. Nalisnick. 2026. The myth of expert specialization in moes: Why routing reflects geometry, not necessarily domain expertise. *CoRR*, abs/2604.09780.

Yun Wang, Lingyun Yang, Senhao Yu, Yixiao Wang, Ruixing Li, Zhixiang Wei, James Yen, and Zhengwei Qi. 2025b. Buddymoe: Exploiting expert redundancy to accelerate memory-constrained mixture-of-experts inference. *arXiv preprint arXiv:2511.10054*.

Lemeng Wu, Mengchen Liu, Yinpeng Chen, Dongdong Chen, Xiyang Dai, and Lu Yuan. 2022. Residual mixture of experts. *arXiv preprint arXiv:2204.09636*.

Minghao Yang, Ren Togo, Guang Li, Takahiro Ogawa, and Miki Haseyama. 2026. L2r: Low-rank and lipschitz-controlled routing for mixture-of-experts. *arXiv preprint arXiv:2601.21349*.

Biao Zhang and Rico Sennrich. 2019. Root mean square layer normalization. *Advances in neural information processing systems*, 32.

Mohan Zhang, Pingzhi Li, Jie Peng, Mufan Qiu, and Tianlong Chen. 2025. Advancing moe efficiency: A collaboration-constrained routing (c2r) strategy for better expert parallelism design. 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 6815–6825.

Chen Zheng, Yuhang Cai, Deyi Liu, Jin Ma, Yiyuan Ma, Yuan Yang, Jing Liu, Yutao Zeng, Xun Zhou, and Siyuan Qiao. 2025. Gatepro: Parameter-free expert selection optimization for mixture-of-experts models. *CoRR*, abs/2510.13079.

Barret Zoph, Irwan Bello, Sameer Kumar, Nan Du, Yanping Huang, Jeff Dean, Noam Shazeer, and William Fedus. 2022. St-moe: Designing stable and transferable sparse expert models. *arXiv preprint arXiv:2202.08906*.

A Proof of Theorem 4.1.

Let $S \subset \mathbb{R}^d$ be a fixed r-dimensional subspace, and let $\mathbf{P}_S$ denote its orthogonal projection matrix. Since $\mathbf{P}_S$ is an orthogonal projector onto an r-dimensional subspace, its eigenvalues consist of r ones and $d-r$ zeros. Therefore, $\mathrm{tr}(\mathbf{P}_S) = r$.

Let T be a random q-dimensional subspace, where $q \leq d$, constructed by sampling a random matrix $\mathbf{A} \in \mathbb{R}^{d\times q}$ with independent standard Gaussian columns and orthonormalizing them into $\mathbf{Q} = [\mathbf{q}_1, \ldots, \mathbf{q}_q]$, where $\mathbf{Q}^\top\mathbf{Q} = \mathbf{I}_q$. Since the columns of $\mathbf{Q}$ form an orthonormal basis of T, the corresponding projection matrix onto T is

$$\mathbf{P}_T = \mathbf{Q}\mathbf{Q}^\top = \sum_{j=1}^{q} \mathbf{q}_j\mathbf{q}_j^\top. \tag{8}$$

We measure the projection overlap between the fixed subspace S and the random subspace T by $\mathrm{tr}(\mathbf{P}_S\mathbf{P}_T)$. Substituting the decomposition of $\mathbf{P}_T$, we obtain

$$\begin{aligned}\mathrm{tr}(\mathbf{P}_S\mathbf{P}_T) &= \mathrm{tr}\left(\mathbf{P}_S \sum_{j=1}^{q} \mathbf{q}_j\mathbf{q}_j^\top\right) \\ &\overset{(a)}{=} \sum_{j=1}^{q} \mathrm{tr}\left(\mathbf{P}_S\mathbf{q}_j\mathbf{q}_j^\top\right),\end{aligned} \tag{9}$$

where (a) follows from the linearity of the trace operator.

Next, because the Gaussian distribution is rotationally invariant, the orthonormalized vectors $\mathbf{q}_j$ are marginally distributed as isotropic random unit directions in $\mathbb{R}^d$. Hence, for each j,

$$\mathbb{E}\left[\mathbf{q}_j\mathbf{q}_j^\top\right] = \frac{1}{d}\mathbf{I}_d. \tag{10}$$

We now compute the expected contribution of one random direction $\mathbf{q}_j$ to the projection overlap:

$$\begin{aligned}\mathbb{E}\left[\mathrm{tr}\left(\mathbf{P}_S\mathbf{q}_j\mathbf{q}_j^\top\right)\right] &\overset{(b)}{=} \mathrm{tr}\left(\mathbf{P}_S\mathbb{E}\left[\mathbf{q}_j\mathbf{q}_j^\top\right]\right) \\ &\overset{(c)}{=} \mathrm{tr}\left(\mathbf{P}_S\frac{1}{d}\mathbf{I}_d\right) \\ &= \frac{1}{d}\mathrm{tr}(\mathbf{P}_S) = \frac{r}{d},\end{aligned} \tag{11}$$

where (b) follows from the linearity of both expectation and trace, allowing the expectation to be moved inside the trace while keeping the deterministic matrix $\mathbf{P}_S$ fixed. Step (c) follows from Eq. (10).

Finally, taking expectation over the total overlap gives

$$\begin{aligned}\mathbb{E}\left[\mathrm{tr}(\mathbf{P}_S\mathbf{P}_T)\right] &= \mathbb{E}\left[\sum_{j=1}^{q} \mathrm{tr}\left(\mathbf{P}_S\mathbf{q}_j\mathbf{q}_j^\top\right)\right] \\ &\overset{(d)}{=} \sum_{j=1}^{q} \mathbb{E}\left[\mathrm{tr}\left(\mathbf{P}_S\mathbf{q}_j\mathbf{q}_j^\top\right)\right] \\ &= \sum_{j=1}^{q} \frac{r}{d} = \frac{rq}{d}.\end{aligned} \tag{12}$$

Here, (d) follows from the linearity of expectation, and the second equality substitutes the single-direction expectation derived above.

Therefore, the expected projection overlap between a fixed r-dimensional subspace S and a random q-dimensional subspace T is

$$\mathbb{E}\left[\mathrm{tr}(\mathbf{P}_S\mathbf{P}_T)\right] = \frac{rq}{d}. \tag{13}$$

When $r \ll d$ and q is not too large, this expected overlap is small. This shows that a low-dimensional subspace has small expected overlap with an unrelated random subspace. In our context, this supports the intuition that the subspace spanned by low-rank expert-assigned hidden states has small expected overlap with unrelated random directions.

Interpretation of the overlap term. For two subspaces S and T, the quantity $\mathrm{tr}(\mathbf{P}_S\mathbf{P}_T)$ measures their total squared projection overlap. Equivalently, if $\theta_1, \ldots, \theta_{\min(r,q)}$ are the principal angles between S and T, then

$$\mathrm{tr}(\mathbf{P}_S\mathbf{P}_T) = \sum_{j=1}^{\min(r,q)} \cos^2(\theta_j).$$

Thus, a smaller value of $\mathrm{tr}(\mathbf{P}_S\mathbf{P}_T)$ indicates that the two subspaces are closer to being orthogonal.

B Detailed Experimental Setup

This section provides additional details on the experimental setup used in our main experiments, including model architectures, baselines, and training hyperparameters.

Model architecture. Table 4 provides the detailed architectural configurations of the Base and Large MoE models used in our experiments. Both models follow a DeepSeek-style MoE Transformer

Configuration	Base	Large
Total Params	2B	8B
Activated Params	0.81B	1.29B
Sparsity Ratio	3.96	5.82
Hidden Size	1280	1920
MoE Layers	12	14
Attention Heads	16	24
MoE Intermediate Size	894	640
Shared Experts	2	2
Routed Experts	64	128
Top-k	6	8
Vocabulary Size	153376	153376

Table 4: Detailed model configurations for the Base and Large MoE models.

architecture (Dai et al., 2024), where all Transformer layers are equipped with MoE FFN modules. Each MoE layer contains both shared experts and routed experts. We use RMSNorm (Zhang and Sennrich, 2019), RoPE positional embeddings (Su et al., 2024), and SwiGLU activation (Shazeer, 2020) in the feed-forward modules. All methods use the router z-loss by default to improve training stability (Zoph et al., 2022). To avoid the effect of token dropping, we adopt a dropless MoE routing strategy for all methods (Gale et al., 2023). All models are pretrained on an internal large-scale text corpus of 100B tokens, constructed from publicly available datasets such as C4 (Raffel et al., 2020), The Pile (Gao et al., 2020), and other open-domain pretraining corpora.

Baselines. We compare our proposed LRouting method with three baselines: 1) Auxiliary load-balancing loss (Fedus et al., 2022), which adds an explicit load-balancing objective; 2) Aux-loss-free load balancing (Wang et al., 2024), which balances expert utilization through routing-bias adjustment; and 3) AES-MoE (Guo et al., 2026), short for *Advancing Expert Specialization for Better MoE*, which introduces an orthogonality loss and a routing-logit variance loss to promote expert specialization.

Hyperparameter settings. We use a global batch size of 1024 and a maximum sequence length of 4096 tokens. For the auxiliary load-balancing loss baseline, we set the auxiliary loss coefficient to $\alpha = 10^{-2}$. For aux-loss-free load balancing, we set the routing bias update rate to 0.001. The router z-loss is applied to all methods with coefficient 10^{-6}. For our LRouting method, we set the EMA coefficient to $\beta = 0.9$, the random projection dimension to $d' = 256$, and the low-rank coefficient to $\lambda = 0.1$. We use the AdamW optimizer with $\beta_1 = 0.9$, $\beta_2 = 0.95$, $\epsilon = 10^{-8}$, and a weight decay coefficient of 0.1. The learning rate follows a WSD cosine schedule, with a starting learning rate of 1×10^{-3}.