Expert-Aware Router: Coupling Routers and Experts in Mixture-of-Experts

Anonymous ACL submission

Abstract

Sparse Mixture-of-Experts (MoE) models scale Transformer capacity by activating only a small subset of experts for each token. However, standard MoE routers rely on standalone routing parameters that are weakly coupled with the experts they control, making routing decisions slow to reflect expert updates and **potentially reducing effective expert utilization** under highly sparse, fine-grained expert settings. Existing expert-aware routing methods strengthen this coupling by using expert-side activations, but they often require evaluating all experts before routing and therefore introduce substantial training overhead. We propose **Expert-Aware Router (EAR)**, a lightweight routing mechanism that derives each expert's routing representation directly from its gate projection matrix through a shared learnable probe vector. EAR explicitly couples routing scores with expert parameters while preserving the efficient linear routing form of standard MoE. We further introduce EAR-Hybrid, which combines expert-aware routing with a standard router to improve routing flexibility. Experiments on MoE models ranging from 1B to 16B parameters and trained on 100B to 500B tokens show that EAR-Hybrid improves downstream performance over standard MoE, while EAR improves expert load balance and intra-expert activation strength.

1 Introduction

Sparse Mixture of Experts (MoE) is an effective approach for efficiently scaling the parameter count of Transformers (Shazeer et al., 2017; Lepikhin et al., 2020; Fedus et al., 2022; Jiang et al., 2024a; Zhao et al., 2023). By leveraging sparse expert activation, it enables training models with substantially larger numbers of parameters under the same computational budget. Modern MoE architectures typically adopt highly sparse and fine-grained expert configurations, requiring the router to accurately and evenly select the Top-K routed experts from a massive expert pool (Dai et al., 2024; Yang et al., 2025; Jiang et al., 2024a; Olmo et al., 2025; Abdin et al., 2024; DeepSeek-AI, 2026; GLM-5-Team et al., 2026).

However, standard MoE still exhibits limited coupling between routers and experts (Lv et al., 2025b; Liu et al., 2026). First, during the forward pass, changes in experts cannot be immediately reflected in the router's index vectors. For a single expert update, its effect on routing can only be captured after the next backward propagation step. Second, updates to the router are also not synchronously reflected in the experts. In extreme cases, after router updates, expert neurons may gradually become inactive, leading to a reduction in effective expert capacity (Huang et al., 2026). To quantify this limitation, we conduct an empirical study on the rank consistency between router outputs and expert output norms in open-source models. The results show that standard MoE generally exhibits weak coupling between routers and experts, especially under high sparsity and fine-grained expert configurations (Yang et al., 2025; Dai et al., 2024; Google DeepMind, 2026; Abdin et al., 2024).

To mitigate the capacity mismatch between the router and experts, recent studies have explored different approaches to strengthen their coupling (Nguyen et al., 2025). AoE (Lv et al., 2025b) and RoutingFree (Liu et al., 2026) enable experts to participate in routing based on their intermediate activations, achieving tight expert–router coupling. However, these methods require activating parts of every expert during the routing stage, leading to a significant slowdown in training. In contrast, the ERC loss no longer performs an explicit pass over all experts before routing; instead, it introduces a loss-based soft constraint to align the routing distribution with expert capacity, and thus the coupling remains implicit (Lv et al., 2025a). Overall, these

approaches either sacrifice efficiency or fail to ensure explicit coupling

To address this tension, we propose the Expert-Aware Router (EAR), a lightweight routing mechanism that no longer learns independent routing vectors for each expert. Instead, it dynamically generates routing representations directly from expert parameters. Specifically, we introduce a learnable projection vector that maps each expert's gate projection matrix into a compact routing vector. These routing vectors are then concatenated to form the router matrix used in standard MoE routing. This design tightly couples routing decisions with expert parameters while avoiding additional architectural complexity. We further implement EAR in an expert-parallel setting, enabling efficient training.

To validate our method, we evaluate two variants: EAR-Hybrid, which augments a standard MoE router with the EAR mechanism, and EAR, which uses only the generated routing vectors as the router. We train models ranging from 1B to 16B parameters on 100B to 500B tokens. We find that EAR-Hybrid outperforms standard MoE in terms of the average performance across seven downstream benchmarks. In terms of load balancing, EAR improves both max_vio and min_vio during training. On one hand, a reduced number of highly routed experts leads to improved training stability and speed; on the other hand, fewer dead experts (Huang et al., 2026) indicate more effective utilization of parameters. In addition, at the intra-expert level, EAR produces higher activation strengths for neurons within each expert compared to standard MoE, further improving expert utilization.

2 Preliminaries

Mixture-of-Experts. A standard Transformer-based MoE layer replaces the dense feed-forward network (FFN) with N expert FFNs $\{E_j\}_{j=1}^{N}$, each activated only for a selected subset of tokens. Given an input hidden state $\mathbf{x} \in \mathbb{R}^d$ with hidden dimension d, the router assigns a score to each expert and selects the top-K experts (Lepikhin et al., 2020; Fedus et al., 2022). The layer output is

$$\mathbf{y} = \sum_{j \in \mathcal{T}(\mathbf{x})} s_j \cdot E_j(\mathbf{x}), \tag{1}$$

where $\mathcal{T}(\mathbf{x})$ denotes the selected expert set and s_j is the corresponding normalized gating weight. We summarize the main notations used in Appendix A.

SwiGLU Expert. Prior work often interprets FFNs as key–value memories (Geva et al., 2021). Modern MoE models (Jiang et al., 2024b; Yang et al., 2025; Dai et al., 2024) commonly replace the standard FFN in each expert E_j with a GLU-style gated FFN (Dauphin et al., 2017; Shazeer, 2020):

$$\mathbf{k}^j = \mathrm{Swish}\left(W^{\mathrm{gate},j}\mathbf{x}\right) \odot \left(W^{\mathrm{up},j}\mathbf{x}\right) \in \mathbb{R}^{d_{\mathrm{ff}}} \tag{2}$$

$$E_j(\mathbf{x}) = \sum_{i}^{d_{\mathrm{ff}}} \mathbf{k}_i^j \cdot W_i^{\mathrm{down},j}, \tag{3}$$

where $W^{\mathrm{gate},j}, W^{\mathrm{up},j} \in \mathbb{R}^{d_{\mathrm{ff}} \times d}$, $W^{\mathrm{down},j} \in \mathbb{R}^{d \times d_{\mathrm{ff}}}$ and $\mathrm{Swish}(x) = x \cdot \mathrm{sigmoid}(x)$. The same key–value memory interpretation naturally extends to SwiGLU: the relative magnitude of each element in $\mathbf{k}_j$ reflects the activation strength of the corresponding memory component.

Top-K Router. In the standard MoE router, expert scores are computed by an independent router matrix with activation functions (Yang et al., 2025):

$$\mathbf{u} = W^{\mathrm{router}}\mathbf{x} \tag{4}$$

$$\mathcal{T}(\mathbf{x}) = \mathrm{TopK}(\mathbf{u}), \tag{5}$$

$$\mathbf{t}_j = \begin{cases} \mathbf{u}_j, & j \in \mathcal{T}(\mathbf{x}), \\ 0, & \text{otherwise.} \end{cases} \tag{6}$$

where $W^{\mathrm{router}} \in \mathbb{R}^{N \times d}$. Whether expert j is selected depends solely on its router score $\mathbf{u}_j = \langle W_j^{\mathrm{router}}, \mathbf{x} \rangle$. The relationship between routers and experts (Equation 1) admits a similar key–value style interpretation to that of SwiGLU (Equation 3). This structural correspondence has led prior work to observe a high degree of similarity between the gate projection $W^{\mathrm{gate},j}$ and the corresponding router vector W_j^{router} (Lo et al., 2025). That is, for the evaluated open-source models, the similarity between the router and gate projection is typically significantly higher than that between the up projection and down projection.

3 Methodology

3.1 Rank Alignment between Router and Experts

Prior theoretical work views each expert as a component of a dense network, where a sparse top-k MoE approximates the dense output by selecting the k experts with the largest output norms (Su, 2025, 2024). Several studies further adopt this norm-based view and report strong results (Lo et al.,

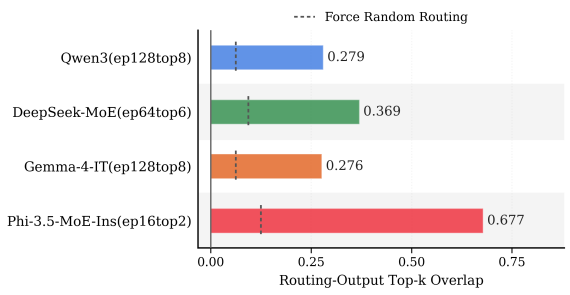

Figure 1: Top-k overlap between router-selected experts and output-norm top-k experts in open-source MoE models. Dashed lines show force-random routing.

2025; Nguyen et al., 2025; Lv et al., 2025b; Liu et al., 2026; Lv et al., 2025a). Motivated by this, we examine whether such a pattern also emerges in open-source MoE models where routers and experts are trained independently.

We rank experts by router logits and by output norms obtained from activating all experts, then compare their top-k sets. As shown in Figure 1, learned routers exceed force-random routing, reaching 0.276–0.677 overlap across four open-source MoE models. However, the overlap remains far from perfect: Phi-3.5-MoE-Ins (ep16top2), the least sparse setting here, obtains the highest overlap, while more sparse or fine-grained settings remain much lower. This suggests that higher sparsity and finer expert granularity make accurate routing harder. Appendix B gives details.

This gap motivates a parameter-level question: can routing scores be tied to expert parameters without computing all expert responses? We answer this by analyzing which SwiGLU projection best preserves the contribution ranking.

3.2 Emergent Similarity Between Routers and Gate Projections

In this section, we derive the similarity between the router and gate projection observed in prior work on open-source models (Lo et al., 2025).

Consider a dense MoE, whereas a sparse MoE reduces computation by selecting the top-k experts with the largest predicted outputs (Su, 2025):

$$\mathbf{y} = \sum_{j=1}^{n} E_j(\mathbf{x}), \quad \sum_{j\in\mathcal{T}} E_j(\mathbf{x}) \approx \sum_{j=1}^{n} E_j(\mathbf{x}). \quad (7)$$

That is, under the assumption that expert outputs are orthogonal, the approximation error decreases by retaining vectors with larger norms. Thus, the ideal router should preserve the ranking induced by $\|E_j(\mathbf{x})\|$. To simplify the analysis, consider each Expert with only one-neuron,

$$\mathbf{E}_j(\mathbf{x}) = \mathrm{Swish}(\underbrace{\mathbf{x}^\top W^{\mathrm{gate},j}}_{\mathbf{a}^j}) \cdot \underbrace{\mathbf{x}^\top W^{\mathrm{up},j}}_{\mathbf{b}^j} \cdot W^{\mathrm{down},j}. \quad (8)$$

The ranking of $|E_j(\mathbf{x})| = |Swish(\mathbf{a}^j)| \cdot |\mathbf{b}^j| \cdot |W^{down,j}|$ can be simplified by removing the input-dependent term, *i.e.*reducing it to ranking:

$$\sigma(\mathbf{a}^j) \cdot |\mathbf{a}^j| \cdot |\mathbf{b}^j|, \quad (9)$$

By analyzing each term in Equation 9, we can identify the source of the similarity between the gate projection and the router. For the up projection, the resulting term $|\mathbf{b}^j| = |\mathbf{x}^\top W_i^{\mathrm{up}}|$ is wrapped by an absolute value, and thus it does not guarantee an identical ranking. Therefore, directly using the up projection as a linear routing signal can produce the wrong ranking. However, the gate factor exhibits a stronger rank-preserving structure:

$$\sigma(\mathbf{a}^j) \cdot |\mathbf{a}^j|. \quad (10)$$

For positive gate activations, both $|a_i|$ and $\sigma(a_i)$ increase monotonically with a_i, thereby preserving the ordering of active responses. For negative activations, the sigmoid term suppresses the contribution, effectively reducing the importance of these dimensions in routing decisions. As a result, the signed gate pre-activation a_i serves as a more reliable linear proxy for expert response strength. In contrast, the down projection contributes only a token-independent scaling factor $|W_i^{\mathrm{down}}|$.

3.3 Expert-Aware Router

In this section, we propose the Expert-Aware Router (EAR). Instead of learning an independent router vector for each expert, we dynamically generate the router representation from the corresponding expert parameters, thereby coupling the router and expert more tightly.

We introduce a learnable vector $\boldsymbol{\alpha} \in \mathbb{R}^{d_{\mathrm{ff}}}$, which is used to project each expert's gate projection matrix into a compact routing vector:

$$\mathbf{r}^j = (W^{\mathrm{gate},j})^\top \boldsymbol{\alpha}. \quad (11)$$

Specifically, before each router forward pass, we compute routing vectors from the corresponding experts and concatenate them to construct the full router matrix:

$$\mathbf{W}^{\mathrm{router}} = [r^1, r^2, \cdots, r^N]. \quad (12)$$

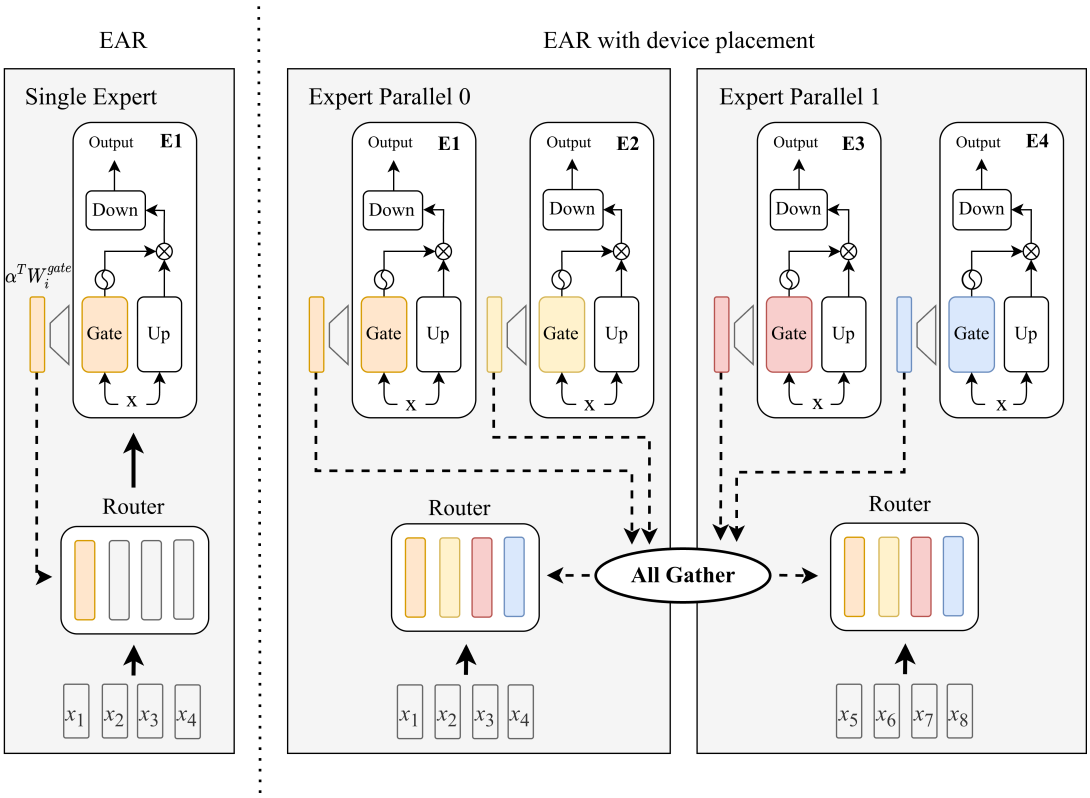

Figure 2: Unlike the standard decoupled router, EAR derives routing vectors from each expert's gate projection. Under expert parallelism, each shard constructs routing vectors for its local experts, and an All-Gather operation assembles the global router matrix.

The resulting router matrix is then used in the standard MoE routing procedure during training. Unlike prior approaches that couple routers and experts through additional architectural complexity, EAR remains highly lightweight, introducing only a simple router-weight generation step on top of standard MoE.

3.4 Update Propagation from Expert Gates to Routing Decisions in EAR

We analyze how an update to an expert gate projection propagates to routing decisions and expert computation. Consider an update step:

$$W^{\text{gate},j} \leftarrow W^{\text{gate},j} + \Delta W^{\text{gate},j}. \tag{13}$$

Standard MoE. In standard MoE (Eq. 1), routing logits are independent of expert parameters. Therefore, under the above update,

$$\Delta u_j = 0, \tag{14}$$

and the top-K selection $\mathcal{T}(\mathbf{x})$ remains unchanged in response to expert-side updates. From a gradient perspective, $W^{\text{gate},j}$ is optimized solely through the expert computation path:

$$\nabla_{W^{\text{gate},j}} \mathcal{L} = \nabla_{\text{expert}}. \tag{15}$$

EAR. In EAR, routing logits are derived from expert parameters via

$$u_j = \boldsymbol{\alpha}^\top W^{\text{gate},j} \mathbf{x}. \tag{16}$$

As a result, an update induces a direct change in routing scores:

$$\Delta u_j = \boldsymbol{\alpha}^\top (\Delta W^{\text{gate},j}) \mathbf{x}. \tag{17}$$

This implies that expert updates can immediately alter routing scores and potentially change the selected set $\mathcal{T}(\mathbf{x})$, thereby inducing coupled dynamics between expert computation and expert selection. From a gradient perspective, $W^{\text{gate},j}$ receives an additional learning signal through the routing pathway:

$$\nabla_{W^{\text{gate},j}} \mathcal{L} = \nabla_{\text{expert}} + \nabla_{\text{router}}. \tag{18}$$

Unlike standard MoE, expert updates in EAR simultaneously influence both expert computation and routing decisions, leading to a tightly coupled co-adaptation between expert specialization and expert selection.

3.5 Implementation Details

To enable efficient training, we implement expert parallelism for EAR in Megatron-LM (Shoeybi et al., 2020) as shown in Figure 2. Specifically, each device reduces the gate projection of its local experts into the corresponding router vectors according to Equation 11, and then performs an all-gather operation within the expert-parallel region to assemble the full router matrix from these router vectors. When implementing expert parallelism, we use a differentiable version of `all_gather` to preserve gradients during backpropagation. It is important to note that each router vector depends solely on the expert weights and is independent of the input hidden states. Consequently, this procedure can be effectively hidden at any point after the completion of the preceding backward pass. As a result, EAR introduces only negligible wall-clock overhead in theory. Moreover, since the parameters are frozen after training, EAR can be exported as a standard router for inference without any additional computation overhead.

To further improve training stability, we initialize the probe vector as $\boldsymbol{\alpha}_0 = \frac{1}{d_{\text{ff}}}\mathbf{1}$ instead of using zero-mean random initialization. Under this initialization, the initial score becomes

$$s_j^{(0)} = \boldsymbol{\alpha}_0^\top W_j^{\text{gate}}\mathbf{x} = \frac{1}{d_{\text{ff}}}\mathbf{1}^\top W_j^{\text{gate}}\mathbf{x}, \qquad (19)$$

thereby ensuring that the scale of the initial logits is well controlled.

4 Experiments

4.1 Experimental Setup

We resample a 500B- and a 100B-token dataset from the OLMo3 (Olmo et al., 2026) 6T-token mixture for training. We adopt two model scales following an architecture similar to ling-mini-2.0 (Tian et al., 2025), both featuring sparse and fine-grained expert designs: (1) a 1B-A0.1B model with 160 experts and top-8 routing; and (2) a 16B-A1.6B model with 256 experts and top-8 routing. For each model size, we tune learning rates and batch sizes via grid search, selecting configurations that minimize validation loss under the Muon optimizer (Liu et al., 2025). We apply a global auxiliary loss (Fedus et al., 2022) with a coefficient of 5×10^{-3} and use a context length of 8K and an expert parallelism degree of 8. Additional training details are provided in Appendix D. Evaluation benchmarks and aggregation details are described in Appendix D.1.

4.2 Baselines

Vanilla MoE Vanilla MoE uses an independent linear router with the standard auxiliary load-balancing loss. It is the decoupled-router baseline.

AoE AoE (Lv et al., 2025b) removes the router and ranks experts by expert-side activation norms. This makes routing directly depend on expert-internal signals, but requires all experts to produce routing signals before selection.

RoutingFree RoutingFreeMoE lets each expert activate itself based on its internal response (Liu et al., 2026). For token $\mathbf{x}_t$ and expert j, it computes the response norm $r_{t,j} = \|\mathbf{x}_t A_j^{\text{gate}}\|_2$, applies a shifted ReLU gate $g_{t,j}^{\text{RF}} = \text{ReLU}(r_{t,j} - b_j)$, and activates the expert if $f_{t,j}^{\text{RF}} = \mathbb{I}\{g_{t,j}^{\text{RF}} \geq \theta\}$. Thus, routing is determined by expert-side response strength rather than an independent router.

ERC loss. ERC loss keeps the standard router and adds an auxiliary router–expert coupling objective (Lv et al., 2025a). It uses the router embedding of expert i as a proxy token and measures its response on expert $M[i,j] = \|\tilde{R}[i]W_j^{\text{gate}}\|_2$. ERC penalizes off-diagonal responses that exceed a margin-scaled diagonal response, encouraging each proxy to activate its matched expert most strongly. Thus, ERC introduces coupling only through an auxiliary loss, while the inference-time router remains unchanged.

EAR-Hybrid. EAR-Hybrid adds the expert-aware router to the standard independent router, using $W_{\text{hyb}} = W_r + R_{\text{EAR}}$ with $\mathbf{r}_j^{\text{EAR}} = (W_j^{\text{gate}})^\top \boldsymbol{\alpha}$. The resulting scores are $\mathbf{s}^{\text{hyb}} = W_{\text{hyb}}\mathbf{x}$. This semi-coupled router preserves expert-derived scoring while keeping an independent component for direct load-balancing control.

4.3 Main Results

Table 2 reports downstream benchmark accuracy. For the ablation in Section 5.3, we also report EAR(Up) and EAR(Down), which replace the gate projection in EAR with the up projection and down projection, respectively.

Enhanced Performance Under different model scales and token budgets, EAR-Hybrid consistently achieves strong downstream accuracy. RoutingFree performs best in the 1B-scale, 100B-token setting, which is understandable because its decision rule directly uses the norm of the full gate

Method	Routing score $s_{t,j}$	Aux. loss	Computation	Comm.
MoE	$(\mathbf{w}_j^{\text{router}})^\top \mathbf{x}_t$	$\mathcal{L}_{\text{bal}}$	$O(TNd)$	/
AoE	$\lVert \tilde{\mathbf{h}}_j(\mathbf{x}_t) \rVert_2$	$\mathcal{L}_{\text{bal}}$	$O(TNdr + TNr)$	$O(TN)$
RoutingFree	$\lVert W_j^{\text{gate}} \mathbf{x}_t \rVert_2$	$\mathcal{L}_{\text{rf bal}}$	$O(TNdd_{\text{ff}} + TNd_{\text{ff}})$	$O(TN)$
ERC loss	$(\mathbf{w}_j^{\text{router}})^\top \mathbf{x}_t$	$\mathcal{L}_{\text{bal}} + \mathcal{L}_{\text{ERC}}$	$O(TNd + N^2dd_{\text{ff}})$	$O(1)$
EAR	$\boldsymbol{\alpha}^\top W_j^{\text{gate}} \mathbf{x}_t$	$\mathcal{L}_{\text{bal}}$	$O(Ndd_{\text{ff}} + TNd)$	$O(Nd)$
EAR-Hybrid	$(\mathbf{w}_j^{\text{router}})^\top \mathbf{x}_t + \boldsymbol{\alpha}^\top W_j^{\text{gate}} \mathbf{x}_t$	$\mathcal{L}_{\text{bal}}$	$O(Ndd_{\text{ff}} + TNd)$	$O(Nd)$

Table 1: Comparison of router–expert coupling methods. The table reports the per-token score used to rank expert j, asymptotic routing-score complexity for T tokens and N experts, and the additional EP routing communication beyond the standard sparse-MoE token dispatch.

Model	Iter. time ↓	LAMB.	HellaS.	Wino.	ARC-E	ARC-C	PIQA	MMLU	Avg.
1B-scale MoE, trained on 100B tokens.									
MoE	2.475	41.68	41.86	51.22	48.57	26.62	69.70	–	46.61
AoE	2.871	26.85	30.92	51.38	37.75	23.72	58.37	–	38.17
RoutingFree	6.960	**45.17**	**44.62**	50.90	**51.80**	28.32	**70.23**	–	**48.51**
MoE + ERC	2.473	40.87	42.04	51.85	48.53	26.37	69.48	–	46.52
EAR-Hybrid	**2.469**	42.97	41.59	**52.64**	48.48	**28.50**	69.21	–	47.23
Ablations on 1B-scale EAR, trained on 100B tokens.									
EAR	2.542	**41.98**	**41.88**	50.59	**50.08**	**27.65**	**69.53**	–	**46.95**
EAR(Up)	–	39.86	39.52	**51.46**	46.34	25.34	67.79	–	45.05
EAR(Down)	–	39.36	40.53	51.22	46.59	27.56	69.15	–	45.74
16B-A1.6B, trained on 100B tokens.									
MoE	–	63.19	**46.90**	56.99	68.31	40.02	75.19	–	58.43
MoE + ERC	–	**63.48**	46.63	55.64	**70.37**	38.57	73.45	–	58.02
EAR	–	62.04	45.33	54.70	65.61	39.25	73.56	–	56.75
EAR-Hybrid	–	62.92	46.84	**59.04**	67.00	**40.19**	**75.63**	–	**58.60**
16B-A1.6B, trained on 500B tokens.									
MoE	–	57.05	57.31	**57.77**	**64.18**	36.95	73.72	37.50	57.83
EAR	–	59.85	56.75	56.43	61.24	36.69	**75.30**	**38.00**	57.71
EAR-Hybrid	–	**60.94**	**57.40**	57.38	61.87	**37.29**	74.92	37.78	**58.30**

Table 2: Downstream evaluation results across the three controlled pretraining configurations. The best result in each block is shown in bold, and the second-best result is underlined.

response $W_j^{\text{gate}}\mathbf{x}$ without a low-rank approximation and therefore preserves more expert-response information. This richer signal, however, comes with substantially lower efficiency, as discussed in Section 5.2. Beyond this small-scale regime, EAR-Hybrid remains the most reliable performer, achieving the best average score in both 16B-scale settings. This suggests that semi-coupled routing is substantially more effective than both fully decoupled and fully coupled routing.

Rank Alignment We evaluate router–expert alignment on the 16B-A1.6B models by comparing router-induced rankings with token-conditioned expert output norms. As shown in Figure 3, EAR achieves higher alignment than both MoE and EAR-Hybrid. This confirms that coupling routing scores with expert gate projections effectively improves rank alignment. Notably, EAR-Hybrid also improves alignment under partial coupling, suggesting that the gain comes from better expert selection rather than simple logit rescaling.

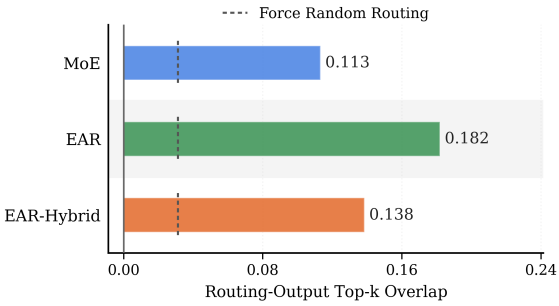

Figure 3: Router–expert ranking alignment metrics for the 16B-A1.6B model trained on 500B tokens.

Improved Load Balance Figure 4 shows a clear trade-off. The standard MoE router obtains a lower global auxiliary loss and different dead-expert recovery patterns, because the auxiliary objective can directly update its independent router matrix.

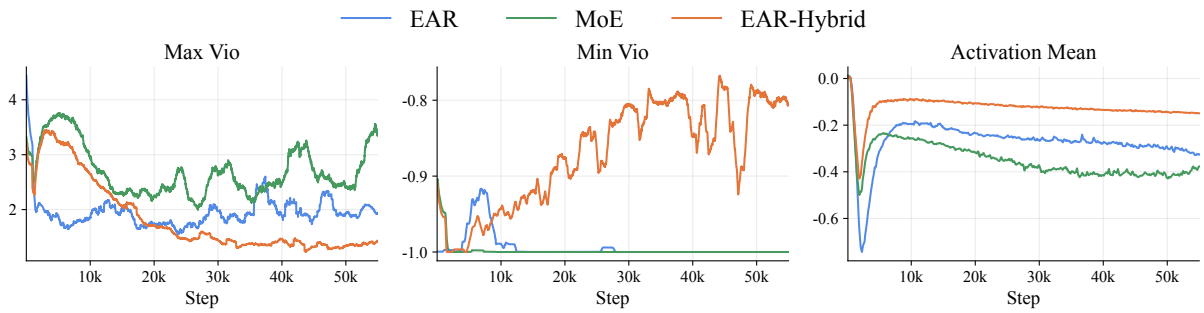

Figure 4: Load-balancing and expert activation behavior for the 16B-A1.6B models trained on 500B tokens. The left and middle panels show MaxVio and MinVio. The right panel shows the mean expert FFN activation.

In EAR, routing logits are derived from expert gate projections, making load-balancing gradients coupled with expert-side representations and less direct. In contrast, EAR better suppresses extreme overload, yielding lower $\mathrm{max_vio}$ under concentrated routing. Thus, MoE provides more direct aggregate load control, whereas EAR provides a stronger expert-aware routing bias that reduces severe overload. This complementarity motivates EAR-Hybrid, which combines independent router matrix with the expert-derived routing matrix.

Improved Expert Utilization As shown in the right panel of Figure 4, we measure the activation strength of neurons within each expert by computing the mean value of Equation 2, as a proxy for expert utilization. We observe that EAR yields a substantially higher average value of $\mathbf{k}$ (converging around -0.15) compared to standard MoE (converging around -0.4), indicating that, under the same number of activated experts, a larger fraction of neurons are effectively engaged, and thus the experts are more fully utilized.

5 Analysis

5.1 Gradient Flow Ablation

We ablate the two trainable components in the expert-aware routing score:

$$s_j = \boldsymbol{\alpha}^\top W_j^{\mathrm{gate}} \mathbf{x}. \tag{20}$$

The full EAR model allows routing-score gradients to update both the shared probe vector $\boldsymbol{\alpha}$ and the expert gate projection W_j^{gate}. We compare it with two restricted variants.

DETACH GATE detaches W_j^{gate} in the routing score, so routing-score gradients update only $\boldsymbol{\alpha}$. FREEZE PROBE freezes $\boldsymbol{\alpha}$, so routing-score gradients update only W_j^{gate}. Notably, W_j^{gate} remains trainable in all variants.

Figure 5 shows that the two gradient paths play different roles. EAR achieves the best load balance. DETACH GATE is less stable: it partially controls overload, but still shows large spikes and weaker recovery of under-used experts. FREEZE PROBE gives the weakest load balance, with high $\mathrm{max_vio}$ and $\mathrm{min_vio}$ staying close to -1, indicating nearly inactive experts. As shown by the loss curves in Figure 5, detaching the gate projections increases language-modeling loss, whereas freezing the probe substantially lowers it. We attribute this to highly imbalanced routing: a few experts receive most tokens and become strongly optimized, which reduces loss but harms expert utilization. Overall, EAR provides the best trade-off: the probe vector stabilizes expert allocation, while routing gradients to the gate projections preserve the expert-aware optimization signal.

5.2 Efficiency Analysis

We analyze the overhead of coupling routing scores with expert parameters. As shown in Table 1, MoE and ERC use independent linear routers and require no extra expert parallelism (EP) communication. EAR only employs a lightweight router generation mechanism. After weight generation, it preserves the same linear scoring form, incurring only a token-independent router construction cost and an $O(Nd)$ All-Gather under expert parallelism (EP). EAR-Hybrid shares the same overhead. In contrast, AoE and RoutingFree compute token-dependent expert responses before routing, so their costs scale with both T and N and require extra EP score aggregation.

Table 2 shows the same trend in iteration time. EAR remains close to MoE, while AoE and RoutingFree are much slower. The slight speed advantage of EAR-Hybrid mainly comes from better load balance under EP=8, not lower routing cost.

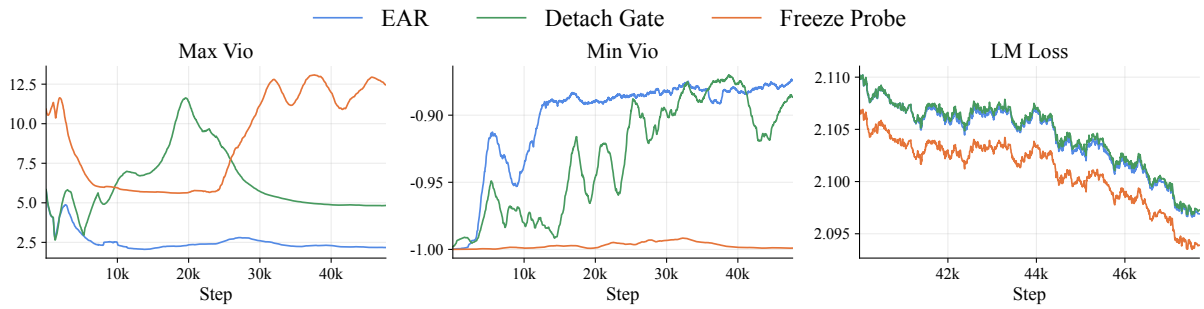

Figure 5: Gradient-flow ablation for expert-aware routing. EAR allows routing-score gradients to update both the probe vector and the expert gate projections. DETACH GATE stops routing-score gradients, while FREEZE PROBE keeps the probe vector fixed. The left and middle panels report expert-level overload and under-utilization through maxvio and min_vio, while the right panel reports language-modeling loss.

5.3 Component Ablations

We ablate the source projection used to generate routing vectors:

$$
\begin{aligned}
s_j^{(q)} &= \boldsymbol{\alpha}^\top A_j^{(q)} \mathbf{x}, \\
A_j^{(q)} &\in \left\{ W_j^{\text{gate}}, W_j^{\text{up}}, (W_j^{\text{down}})^\top \right\}.
\end{aligned}
\tag{21}
$$

This directly tests the theoretical claim in Section 3.2: gate projections should preserve expert-importance order better than up or down projections. As shown in Table 2, using the gate projection consistently outperforms the up- and down-projection variants.

6 Related Work

Sparse Mixture-of-Experts. Sparse MoE layers enable large parameter counts with moderate per-token computation (Shazeer et al., 2017; Lepikhin et al., 2020; Fedus et al., 2022). Recent surveys and system reports show that MoE has become a practical scaling strategy for modern LLMs (Zhao et al., 2023; Lo et al., 2025; Wei et al., 2024). Recent large language models further demonstrate the practical value of sparse expert architectures (Jiang et al., 2024a; Dai et al., 2024; DeepSeek-AI et al., 2025; Yang et al., 2025). EAR is orthogonal to these architectures: it changes how experts are scored, while leaving the sparse expert computation pattern intact.

Router Design and Load Balancing. Most MoE systems rely on learned top-K routers plus auxiliary load-balancing objectives (Fedus et al., 2022; Zoph et al., 2022). Other studies explore expert-choice routing (Zhou et al., 2022), auxiliary-loss-free or adaptive load balancing (Wang et al., 2024; Wen et al., 2025), token-adaptive expert counts (Zeng et al., 2024), router architecture variants (Harvey et al., 2025), and dynamically generated routers (Do et al., 2023). These methods improve routing behavior, but usually keep the router parameterization separate from expert internals.

Coupling Routers and Experts. Recent work argues that router–expert separation can lead to suboptimal expert selection. AoE lets experts evaluate themselves through internal activation norms (Lv et al., 2025b), competition-based routing selects experts according to neural response (Pham et al., 2024; Nguyen et al., 2025), and ERC loss encourages router embeddings to match expert capabilities (Lv et al., 2025a). In contrast, our method constructs router weights directly from expert gate projections, thereby enabling explicit parameter coupling during training while retaining the standard linear routing formulation at inference.

7 Conclusion

We presented EAR, an expert-aware router that derives routing vectors from expert gate projections rather than learning router parameters independently from the experts they select. Across controlled pretraining runs and rank-alignment diagnostics, EAR improves the agreement between router rankings and token-conditioned expert output norms while preserving the standard linear-router inference form. The load-balancing analysis shows that this coupling introduces a trade-off: expert-aware routing improves selection quality and suppresses severe overload, while independent router parameters provide more direct aggregate load control. EAR-Hybrid combines these strengths by adding an independent router component to the expert-derived routing matrix.

Limitations

EAR is currently designed for SwiGLU-style MoE experts and may not directly apply to other expert architectures. Our experiments are limited to controlled pretraining settings at two MoE scales, so broader model sizes, data mixtures, sparsity levels, expert counts, and downstream tasks remain to be evaluated.

Ethics Statement

We follow the required rules for artifact use and reporting. Our use of existing artifacts, including models, tokenizers, datasets, and evaluation benchmarks, is consistent with their intended research use. For artifacts created in this work, including code and experimental outputs, the intended use is research on MoE routing, and this use is compatible with the access conditions of the original artifacts. AI tools were used only for manuscript polishing, and all AI-assisted code generation was manually reviewed by the authors. This work does not introduce a new dataset, user-facing application, or human-subject study.

References

Marah Abdin, Jyoti Aneja, Hany Awadalla, Ahmed Awadallah, Ammar Ahmad Awan, Nguyen Bach, Amit Bahree, Arash Bakhtiari, Jianmin Bao, Harkirat Behl, Alon Benhaim, Misha Bilenko, Johan Bjorck, Sébastien Bubeck, Martin Cai, Qin Cai, Vishrav Chaudhary, Dong Chen, Dongdong Chen, and 110 others. 2024. Phi-3 technical report: A highly capable language model locally on your phone. *Preprint*, arXiv:2404.14219.

Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. 2020. PIQA: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 7432–7439.

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

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. *arXiv preprint*.

Yann N. Dauphin, Angela Fan, Michael Auli, and David Grangier. 2017. Language modeling with gated convolutional networks. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 933–941. PMLR.

DeepSeek-AI. 2026. DeepSeek-V4: Towards highly efficient million-token context intelligence.

DeepSeek-AI, Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Daya Guo, Dejian Yang, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, and 181 others. 2025. DeepSeek-V3 Technical Report. *arXiv preprint*.

Giang Do, Khiem Le, Quang Pham, TrungTin Nguyen, Thanh-Nam Doan, Bint T. Nguyen, Chenghao Liu, Savitha Ramasamy, Xiaoli Li, and Steven Hoi. 2023. HyperRouter: Towards Efficient Training and Inference of Sparse Mixture of Experts. *arXiv preprint*.

William Fedus, Barret Zoph, and Noam Shazeer. 2022. Switch Transformers: Scaling to Trillion Parameter Models with Simple and Efficient Sparsity. *arXiv preprint*.

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

GLM-5-Team, Aohan Zeng, Xin Lv, Zhenyu Hou, Zhengxiao Du, Qinkai Zheng, Bin Chen, Da Yin, Chendi Ge, Chenghua Huang, Chengxing Xie, Chenzheng Zhu, Congfeng Yin, Cunxiang Wang, Gengzheng Pan, Hao Zeng, Haoke Zhang, Haoran Wang, Huilong Chen, and 167 others. 2026. GLM-5: from Vibe Coding to Agentic Engineering. *arXiv preprint*.

Google DeepMind. 2026. Gemma 4: Open models by Google DeepMind (gemma-4-26b-a4b-it). `https://huggingface.co/google/gemma-4-26B-A4B-it`. Hugging Face model repository.

Daniel Fidel Harvey, George Weale, and Berk Yilmaz. 2025. Optimizing MoE Routers: Design, Implementation, and Evaluation in Transformer Models. *arXiv preprint*.

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

Ailin Huang, Ang Li, Aobo Kong, and 1 others. 2026. Step 3.5 flash: Open frontier-level intelligence with 11b active parameters. *arXiv preprint*.

Albert Q. Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, Gianna Lengyel, Guillaume Bour, Guillaume Lample, Lélio Renard Lavaud, Lucile Saulnier, Marie-Anne Lachaux, Pierre Stock, Sandeep Subramanian, Sophia Yang, and 7 others. 2024a. Mixtral of Experts. *arXiv preprint*.

Albert Q. Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, Gianna Lengyel, Guillaume Bour, Guillaume Lample, Lélio Renard Lavaud, Lucile Saulnier, Marie-Anne Lachaux, Pierre Stock, Sandeep Subramanian, Sophia Yang, and 7 others. 2024b. Mixtral of experts. *Preprint*, arXiv:2401.04088.

Dmitry Lepikhin, HyoukJoong Lee, Yuanzhong Xu, Dehao Chen, Orhan Firat, Yanping Huang, Maxim Krikun, Noam Shazeer, and Zhifeng Chen. 2020. GShard: Scaling Giant Models with Conditional Computation and Automatic Sharding. *arXiv preprint*.

Jingyuan Liu, Jianlin Su, Xingcheng Yao, Zhejun Jiang, Guokun Lai, Yulun Du, Yidao Qin, Weixin Xu, Enzhe Lu, Junjie Yan, Yanru Chen, Huabin Zheng, Yibo Liu, Shaowei Liu, Bohong Yin, Weiran He, Han Zhu, Yuzhi Wang, Jianzhou Wang, and 9 others. 2025. Muon is scalable for llm training. *Preprint*, arXiv:2502.16982.

Yilun Liu, Jinru Han, Sikuan Yan, Volker Tresp, and Yunpu Ma. 2026. Routing-free mixture-of-experts. *Preprint*, arXiv:2604.00801.

Ka Man Lo, Zeyu Huang, Zihan Qiu, Zili Wang, and Jie Fu. 2025. A closer look into mixture-of-experts in large language models. *Preprint*, arXiv:2406.18219.

Ang Lv, Jin Ma, Yiyuan Ma, and Siyuan Qiao. 2025a. Coupling Experts and Routers in Mixture-of-Experts via an Auxiliary Loss. *arXiv preprint*.

Ang Lv, Ruobing Xie, Yining Qian, Songhao Wu, Xingwu Sun, Zhanhui Kang, Di Wang, and Rui Yan. 2025b. Autonomy-of-Experts Models. *arXiv preprint*.

Nam V. Nguyen, Huy Nguyen, Quang Pham, Van Nguyen, Savitha Ramasamy, and Nhat Ho. 2025. CompeteSMoE – Statistically Guaranteed Mixture of Experts Training via Competition. *arXiv preprint*.

Team Olmo, :, Allyson Ettinger, Amanda Bertsch, Bailey Kuehl, David Graham, David Heineman, Dirk Groeneveld, Faeze Brahman, Finbarr Timbers, Hamish Ivison, Jacob Morrison, Jake Poznanski, Kyle Lo, Luca Soldaini, Matt Jordan, Mayee Chen, Michael Noukhovitch, Nathan Lambert, and 50 others. 2026. Olmo 3. *Preprint*, arXiv:2512.13961.

Team Olmo, Allyson Ettinger, Amanda Bertsch, Bailey Kuehl, David Graham, David Heineman, Dirk Groeneveld, Faeze Brahman, Finbarr Timbers, Hamish Ivison, Jacob Morrison, Jake Poznanski, Kyle Lo, Luca Soldaini, Matt Jordan, Mayee Chen, Michael Noukhovitch, Nathan Lambert, Pete Walsh, and 49 others. 2025. Olmo 3. *Preprint*, arXiv:2512.13961.

OpenAI. 2025. gpt-oss-120b and gpt-oss-20b model card.

Denis Paperno, German 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 *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 1525–1534, Berlin, Germany. Association for Computational Linguistics.

Quang Pham, Giang Do, Huy Nguyen, TrungTin Nguyen, Chenghao Liu, Mina Sartipi, Binh T. Nguyen, Savitha Ramasamy, Xiaoli Li, Steven Hoi, and Nhat Ho. 2024. CompeteSMoE – Effective Training of Sparse Mixture of Experts via Competition. *arXiv preprint*.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2020. WinoGrande: An adversarial winograd schema challenge at scale. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 8732–8740.

Noam Shazeer. 2020. Glu variants improve transformer. *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. *Preprint*, arXiv:1701.06538.

Mohammad Shoeybi, Mostofa Patwary, Raul Puri, Patrick LeGresley, Jared Casper, and Bryan Catanzaro. 2020. Megatron-lm: Training multi-billion parameter language models using model parallelism. *Preprint*, arXiv:1909.08053.

Jianlin Su. 2024. The road to low-rank approximation (iii): Cr.

Jianlin Su. 2025. Moe's travelogue: 1. starting from the geometric meaning.

Changxin Tian, Kunlong Chen, Jia Liu, Ziqi Liu, Zhiqiang Zhang, and Jun Zhou. 2025. Towards greater leverage: Scaling laws for efficient mixture-of-experts language models. *Preprint*, arXiv:2507.17702.

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

Tianwen Wei, Bo Zhu, Liang Zhao, Cheng Cheng, Biye Li, Weiwei Lü, Peng Cheng, Jianhao Zhang, Xiaoyu Zhang, Liang Zeng, Xiaokun Wang, Yutuan Ma, Rui Hu, Shuicheng Yan, Han Fang, and Yahui Zhou. 2024. Skywork-MoE: A Deep Dive into Training Techniques for Mixture-of-Experts Language Models. *arXiv preprint*.

Jialin Wen, Xiaojun Li, Junping Yao, Xinyan Kong, and Peng Cheng. 2025. Adaptive-expert-weight-based load balance scheme for dynamic routing of MoE. *Frontiers in Neurorobotics*, 19.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, and 41 others. 2025. Qwen3 Technical Report. *arXiv preprint*.

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.

Zihao Zeng, Yibo Miao, Hongcheng Gao, Hao Zhang, and Zhijie Deng. 2024. AdaMoE: Token-Adaptive Routing with Null Experts for Mixture-of-Experts Language Models. *arXiv preprint*.

Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, Yifan Du, Chen Yang, Yushuo Chen, Zhipeng Chen, Jinhao Jiang, Ruiyang Ren, Yifan Li, Xinyu Tang, Zikang Liu, and 3 others. 2023. A survey of large language models. *Preprint*, arXiv:2303.18223.

Yanqi Zhou, Tao Lei, Hanxiao Liu, Nan Du, Yanping Huang, Vincent Zhao, Andrew Dai, Zhifeng Chen, Quoc Le, and James Laudon. 2022. Mixture-of-Experts with Expert Choice Routing. *arXiv preprint*.

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*.

A Notation

Unless otherwise specified, subscripts usually denote tensor indices (e.g., vector elements or matrix rows), while superscripts are used as labels or identifiers (e.g., expert index j or projection type).

Notation	Description
d	Transformer hidden dimension
d_{ff}	FFN intermediate dimension
N	Number of experts
$\mathbf{x} \in \mathbb{R}^d$	Input hidden state
E_j	Expert j
$\mathcal{T}(\mathbf{x})$	Top-K selected expert set
s_j	Gating weight of expert j
$\mathbf{u}$	Router score vector
$\mathbf{u}_j$	Router score of expert j
W^{router}	Router projection matrix
$W^{\text{gate},j}$	Gate projection matrix of expert j
$W^{\text{up},j}$	Up projection matrix of expert j
$W^{\text{down},j}$	Down projection matrix of expert j
$\mathbf{k}^j$	Intermediate SwiGLU activation of expert j
$\mathbf{k}_i^j$	i-th intermediate activation in expert j
$W_i^{\text{down},j}$	i-th row of $W^{\text{down},j}$
$\odot$	Element-wise multiplication
$\langle \cdot, \cdot \rangle$	Inner product

Table 3: Main notations used in this paper.

We track the load balancing metrics within a global batch during training using `max_vio` and `min_vio`:

$$\begin{aligned} \max_{\text{v}}\text{io} &= \max_i \left(N f_i - 1\right), \\ \min_{\text{v}}\text{io} &= \min_i \left(N f_i - 1\right). \end{aligned} \tag{22}$$

Notably, $\text{min_vio} = -1$ indicates that there exists at least one dead expert that receives no routed tokens within the batch.

B Rank Alignment Details

This appendix describes the router–expert rank-alignment diagnostic. The main figures report top-K overlap, which measures whether the router selects the same expert set as the output-norm oracle. We also define correlation and NDCG variants for completeness.

Evaluation inputs. We construct a small diagnostic token set containing heterogeneous text patterns, including English sentences, Chinese sentences, mathematical expressions, and code snippets. This set is not used for training. For each model, we run a forward pass in evaluation mode and record the hidden state $\mathbf{x}_{\ell,t}$ entering every MoE layer ℓ for every diagnostic token t.

During this diagnostic forward pass, we additionally compute *full expert activations*: for each token and each MoE layer, all experts are evaluated under `no_grad` only for measurement. This does not change the actual sparse forward computation or the reported model outputs. For a layer with N experts, we record two vectors:

$$\mathbf{s}_{\ell,t} = \mathrm{Router}_\ell(\mathbf{x}_{\ell,t}) \in \mathbb{R}^N, \tag{23}$$

and

$$\mathbf{n}_{\ell,t} = \left[\|E_{\ell,j}(\mathbf{x}_{\ell,t})\|_2\right]_{j=1}^N. \tag{24}$$

Here, $\mathbf{s}_{\ell,t}$ denotes router logits and $\mathbf{n}_{\ell,t}$ denotes expert output norms. For standard MoE, $\mathbf{s}_{\ell,t} = W_{r,\ell}\mathbf{x}_{\ell,t}$. For EAR, $\mathbf{s}_{\ell,t}$ is computed from Eq. 20. For EAR-Hybrid, it is computed from the mixed router matrix in Section 4.2.

Pearson correlation. Pearson correlation measures value-level calibration between router logits and expert output norms:

$$\mathrm{Pearson}_{\ell,t} = \frac{\tilde{\mathbf{s}}_{\ell,t}^\top \tilde{\mathbf{n}}_{\ell,t}}{\|\tilde{\mathbf{s}}_{\ell,t}\|_2 \|\tilde{\mathbf{n}}_{\ell,t}\|_2 + \varepsilon}, \tag{25}$$

where $\tilde{\mathbf{s}}_{\ell,t} = \mathbf{s}_{\ell,t} - \mathrm{mean}(\mathbf{s}_{\ell,t})$ and $\tilde{\mathbf{n}}_{\ell,t} = \mathbf{n}_{\ell,t} - \mathrm{mean}(\mathbf{n}_{\ell,t})$.

Spearman correlation. Spearman correlation measures rank-level agreement. Let $\mathrm{rank}(\mathbf{s}_{\ell,t})$ and $\mathrm{rank}(\mathbf{n}_{\ell,t})$ be the expert ranks induced by router logits and expert output norms, respectively. We compute

$$\mathrm{Spearman}_{\ell,t} = \mathrm{Pearson}\left(\mathrm{rank}(\mathbf{s}_{\ell,t}), \mathrm{rank}(\mathbf{n}_{\ell,t})\right). \tag{26}$$

This metric is more relevant than Pearson for routing, because top-K expert selection depends on ordering rather than score magnitude.

Top-K overlap. Top-K overlap measures whether the router selects the same expert set as the norm-induced oracle. Let

$$\mathcal{T}_{\ell,t}^s = \mathrm{TopK}(\mathbf{s}_{\ell,t}), \qquad \mathcal{T}_{\ell,t}^n = \mathrm{TopK}(\mathbf{n}_{\ell,t}). \tag{27}$$

The overlap score is

$$\mathrm{Overlap@}\,K_{\ell,t} = \frac{|\mathcal{T}_{\ell,t}^s \cap \mathcal{T}_{\ell,t}^n|}{K}. \tag{28}$$

A random router has expected overlap K/N, which is used as the dashed random baseline in Figure 1.

NDCG@K. NDCG@K evaluates whether experts with large output norms are placed near the top of the router ranking. Let $\pi_{\ell,t}^s$ be the expert order obtained by sorting $\mathbf{s}_{\ell,t}$ in descending order, and let $\pi_{\ell,t}^n$ be the order obtained by sorting $\mathbf{n}_{\ell,t}$ in descending order. We define

$$\mathrm{DCG@}\,K_{\ell,t} = \sum_{r=1}^{K} \frac{n_{\ell,t,\pi_{\ell,t}^s(r)}}{\log_2(r+1)}, \tag{29}$$

and

$$\mathrm{IDCG@}\,K_{\ell,t} = \sum_{r=1}^{K} \frac{n_{\ell,t,\pi_{\ell,t}^n(r)}}{\log_2(r+1)}. \tag{30}$$

The normalized score is

$$\mathrm{NDCG@}\,K_{\ell,t} = \frac{\mathrm{DCG@}\,K_{\ell,t}}{\mathrm{IDCG@}\,K_{\ell,t} + \varepsilon}. \tag{31}$$

Unlike top-K overlap, NDCG@K gives larger credit when high-norm experts appear earlier in the router ranking.

Aggregation. All metrics are first computed for each token at each MoE layer. We then average over tokens within each layer:

$$m_\ell = \frac{1}{|\mathcal{D}|} \sum_{t \in \mathcal{D}} m_{\ell,t}, \tag{32}$$

where m denotes one of Pearson, Spearman, Overlap@K, or NDCG@K. The final model-level metric is the mean over MoE layers:

$$m = \frac{1}{L} \sum_{\ell=1}^{L} m_\ell. \tag{33}$$

Thus, the reported number is a layer-balanced average rather than a raw average over all token–layer pairs. This avoids over-weighting models or layers that produce more valid diagnostic tokens after tokenization.

C Orthogonal Loss Ablation

This appendix reports an additional ablation on orthogonal regularization. The motivation is that load balancing encourages experts to receive similar numbers of tokens, but it does not directly require different experts to learn distinct routing directions. Orthogonal regularization instead penalizes similarity among expert routing vectors.

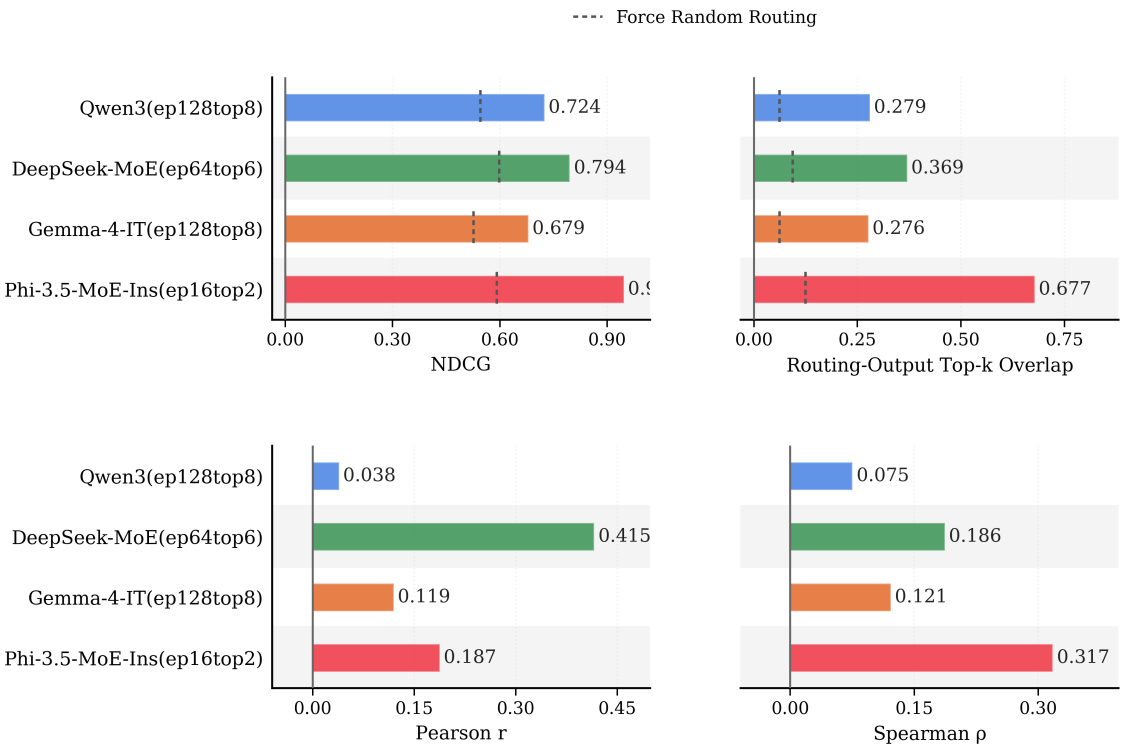

Figure 6: Full router–expert alignment diagnostics for open-source MoE models.

Orthogonal loss. For each MoE layer ℓ, let $R_\ell \in \mathbb{R}^{N\times d}$ denote the routing matrix whose j-th row corresponds to expert j. The definition of R_ℓ depends on the routing method:

$$R_\ell = \begin{cases} W_{r,\ell}, & \text{MoE}, \\ R_\ell^{\text{EAR}}, & \text{EAR}, \\ W_{r,\ell} + R_\ell^{\text{EAR}}, & \text{EAR-Hybrid}, \end{cases} \tag{34}$$

where the expert-aware matrix is

$$R_\ell^{\text{EAR}} = \begin{bmatrix} ((W_{\ell,1}^{\text{gate}})^\top \boldsymbol{\alpha}_\ell)^\top \\ \vdots \\ ((W_{\ell,N}^{\text{gate}})^\top \boldsymbol{\alpha}_\ell)^\top \end{bmatrix}. \tag{35}$$

We first normalize each row:

$$\hat{\mathbf{r}}_i = \frac{\mathbf{r}_i}{\|\mathbf{r}_i\|_2 + \varepsilon}, \qquad \hat{\mathbf{R}} = [\hat{\mathbf{r}}_1; \ldots; \hat{\mathbf{r}}_N]. \tag{36}$$

We then compute the Gram matrix

$$\mathbf{G} = \hat{\mathbf{R}}\hat{\mathbf{R}}^\top, \tag{37}$$

and penalize its deviation from the identity matrix:

The orthogonal loss is

$$\mathcal{L}_{\text{orth}} = \frac{1}{L}\sum_{\ell=1}^{L} \frac{\left\|\tilde{R}_\ell \tilde{R}_\ell^\top - I_N\right\|_{\text{F}}^2}{N^2}. \tag{38}$$

Since each row of $\tilde{R}_\ell$ is normalized, the diagonal of $\tilde{R}_\ell \tilde{R}_\ell^\top$ is close to one. Therefore, this loss mainly penalizes off-diagonal cosine similarity between different expert routing directions.

The final training objective is

$$\mathcal{L} = \mathcal{L}_{\text{LM}} + \lambda_{\text{aux}}\mathcal{L}_{\text{aux}} + \lambda_{\text{orth}}\mathcal{L}_{\text{orth}}. \tag{39}$$

We apply the same λ_{orth} across MoE, EAR, and EAR-Hybrid for a controlled comparison.

Why it differs across methods. For standard MoE, $\mathcal{L}_{\text{orth}}$ only regularizes the free router matrix W_r. It can diversify router embeddings, but it does not directly constrain expert parameters. For EAR, the routing vectors are derived from W_j^{gate}, causing the same orthogonal loss to back-propagate into the expert gate projections. Thus, orthogonal regularization becomes an expert-response regularizer rather than a router-only constraint. For EAR-Hybrid, the loss is applied to the combined matrix $W_r + R_\ell^{\text{EAR}}$, encouraging the independent router component and the expert-derived component to form diverse final routing directions.

Results Table 4 reports the final-step metrics for the 1B-scale runs. We compare each routing method with and without orthogonal regularization using the same training setup.

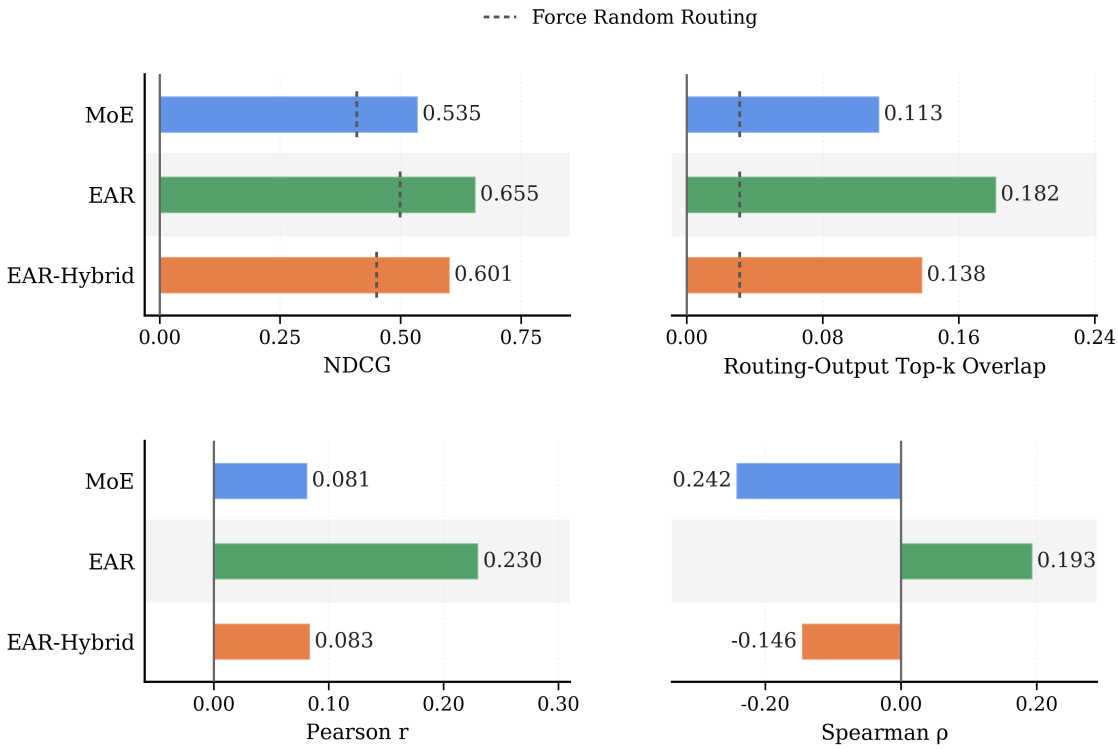

Figure 7: Full router–expert alignment diagnostics for the controlled 16B-A1.6B models trained on 500B tokens.

Method	LM loss ↓	max_vio ↓	min_vio ↑
MoE	2.1628	0.733	-0.726
MoE + Orth.	2.1636	0.881	-0.715
EAR	2.1662	2.078	-0.854
EAR + Orth.	2.1775	1.503	-0.875
EAR-Hybrid	2.1627	0.848	-0.617
EAR-Hybrid + Orth.	**2.1609**	1.013	**-0.504**

Table 4: Orthogonal-loss ablation at the final logged training step (step 47682). We report language-modeling loss and load-balancing violations. For min_vio, values closer to zero indicate fewer severely under-used experts.

Orthogonal regularization has method-dependent effects. For standard MoE, it leaves language-modeling loss almost unchanged, slightly improves under-utilization, but increases the largest overload. For EAR, orthogonal regularization reduces the largest overload from 2.078 to 1.503, suggesting that it can diversify expert-derived routing directions; however, this comes with worse language-modeling loss and slightly worse under-utilization. The most favorable trade-off appears in EAR-Hybrid: adding the orthogonal loss improves LM loss from 2.1627 to 2.1609 and improves min_vio from -0.617 to -0.504, while only moderately increasing max_vio. This supports the hybrid design: the independent router component absorbs direct load-balancing pressure, while the expert-derived component can benefit from diversity regularization without fully constraining expert responses.

D Model and Training Configurations

This appendix summarizes the pretraining configurations used in our controlled experiments. We report the three configurations used for the 100B-token small-scale run, the 500B-token large-scale run, and the 100B-token large-scale run. The effective token budget is computed as

$$T = I_{\text{train}} \times B_{\text{global}} \times S, \tag{40}$$

where I_{train} is the number of training iterations, B_{global} is the global batch size, and S is the sequence length.

Optimization and stabilization details. We use Muon to improve optimization stability in this

Configuration item	Small-100B	Large-100B	Large-500B
Model architecture			
Token budget	100B	100B	500B
Layers	16	20	20
Hidden size d	1024	2048	2048
MoE FFN size	128	512	512
Shared FFN size	–	512	512
Experts	160	256	256
Active experts K	8	8	8
MoE layers	16/16	19/20	19/20
Attention heads	8	16	16
KV groups	1	4	4
Training hyperparameters			
Sequence length	8192	8192	8192
Global batch size	256	1024	1024
Optimizer	Muon	Muon	Muon
LR schedule	constant	cosine	constant
Peak LR	8.6×10^{-4}	3×10^{-3}	3×10^{-3}
Minimum LR	8.6×10^{-4}	3×10^{-4}	3×10^{-3}
Global auxiliary loss	5×10^{-3}	5×10^{-3}	5×10^{-3}
TP / PP / EP / ETP	1 / 1 / 8 / 1	1 / 1 / 8 / 1	1 / 1 / 8 / 1

Table 5: Pretraining configurations for the three controlled runs. TP, PP, EP, and ETP denote tensor, pipeline, expert, and expert-tensor parallel size.

sparse setting, following recent large-scale MoE training recipes that report stable convergence with Muon-style updates (Huang et al., 2026; DeepSeek-AI, 2026). Because SwiGLU MoE experts can develop heavy-tailed intermediate activations, we apply a soft clamp to SwiGLU activations and the final logits, a stabilization practice used in recent MoE systems (OpenAI, 2025; Huang et al., 2026; DeepSeek-AI, 2026).

Shared implementation settings. All runs use SwiGLU experts, RMSNorm with $\epsilon = 10^{-6}$, RoPE with base 10^6, QK layer normalization, grouped-query attention, untied input/output embeddings, no linear bias, BF16 training, FlashAttention, zero attention/hidden dropout, softmax router scores, all-to-all token dispatch, router fusion, grouped GEMM, permute fusion, cross-entropy fusion, distributed checkpoints, asynchronous checkpoint saving, and manual garbage collection every 1,000 iterations. The small run uses vocabulary size 248,320 and KV dimension 128. The two large runs use vocabulary size 151,936, KV dimension 128, rotary percentage 0.5, a shared expert intermediate size of 512, and a router top-K scaling factor of 2.5. All runs use seed 1234, 2,000 warmup iterations, Adam-style Muon moments $(\beta_1, \beta_2) = (0.9, 0.95)$, $\epsilon = 10^{-8}$, weight decay 0.1, gradient clipping at 1.0, Muon momentum 0.95, 5 Newton–Schulz steps, spectral scaling, and blockwise tensor-parallel Muon updates.

Data and evaluation. All three configurations use the OLMo3 pretraining corpus. The 1B-scale 100B-token and 16B-scale 500B-token runs use the Qwen3.5 tokenizer, while the 16B-scale 100B-token run uses the Qwen3-1.7B tokenizer and is initialized from the corresponding 16B-scale checkpoint. All runs use a 999/1/0 train/validation/test split, evaluate validation perplexity every 200 iterations with 10 evaluation iterations, and report zero-shot benchmark results on LAMBADA, HellaSwag, WinoGrande, ARC-Easy, ARC-Challenge, and PIQA with sequence length 8192. For large-model evaluations where available, we additionally report MMLU. Benchmark evaluations are run every 1,000 iterations for the 1B-scale configuration and every 500 iterations for the two 16B-scale configurations.

D.1 Evaluation Details

We report zero-shot downstream accuracy on LAMBADA (Paperno et al., 2016), HellaSwag (Zellers et al., 2019), WinoGrande (Sakaguchi et al., 2020), ARC-Easy and ARC-Challenge (Clark et al., 2018), and PIQA (Bisk et al., 2020). For large-model evaluations where available, we further include MMLU (Hendrycks et al., 2021). The aggregate score is the unweighted average over the reported benchmarks.