MoRA: MoE Pruning via Router Bias Learning and Expert Approximation

Anonymous authors
Paper under double-blind review

Abstract

Mixture-of-Experts (MoE) models enable parameter scaling with limited per-token computation by activating only a small subset of experts for each token, but deploying them still requires loading the complete expert pool into memory. Structured expert pruning can effectively reduce the memory usage by removing experts. However, existing pruning methods either use expert ranking criteria that are not well aligned with model performance or rely on effective expert subset searching that is computationally expensive. Moreover, these methods typically overlook the routing-behavior redundancy among the retained experts. In this paper, we propose MoE Pruning via Router Bias Learning and Expert Approximation (MoRA), a framework for structured MoE expert pruning. We introduce a learnable router bias for each expert and optimize these biases by minimizing the language-modeling loss and a routing-diversity regularizer. The learned router biases sharpen the routing probability distributions to identify experts critical to model performance while encouraging the selection of experts with diverse routing preferences. In addition, we introduce an expert approximation mechanism as a post-pruning enhancement. It leverages the remaining experts to approximate the outputs of pruned experts by affine transformation, further improving the performance of the pruned model. We evaluate MoRA on Qwen3-30B-A3B, DeepSeek-V2-Lite, and Moonlight-16B-A3B, removing 25% and 50% of the routed experts in each MoE layer. Extensive experiments on nine zero-shot benchmarks show that MoRA outperforms state-of-the-art pruning algorithms. Our code will be released.

1 Introduction

Mixture-of-Experts (MoE) models activate only a small subset of experts for each token, enabling model scaling with limited per-token computation (Shazeer et al., 2017; Fedus et al., 2022). Although sparse activation reduces inference computation, deploying MoE models still requires loading the full expert pool, resulting in substantial memory usage. Structured expert pruning offers a direct way to reduce this cost by removing less important experts while retaining the sparse routing structure (Lee et al., 2025; Xie et al., 2024; Zhang et al., 2026). Hence, the core challenge is to determine which experts can be removed while preserving the predictive behavior of the model.

Early expert pruning methods rank experts using aggregate routing statistics, such as activation frequency and average gate values (Koishekenov et al., 2023; Xie et al., 2024). However, these expert ranking criteria may not directly align with the model predictive performance. For example, frequency-based pruning may assign higher scores to experts that frequently process common function words with less semantic information (Dong et al., 2026), as illustrated in Figure 1 (a). Recent methods select an expert subset by minimizing layer-wise output reconstruction error on a calibration dataset under a prescribed retention budget (Lu et al., 2024), as illustrated in Figure 1 (b). However, these methods require evaluating candidate expert subsets through a set-level search, whose cost grows rapidly with the number of experts. Moreover, the above-mentioned methods generally do not consider the diversity in the routing behavior of retained experts. Consequently, they may retain experts with highly similar routing patterns, leading to redundancy within the retained set.

To address these issues, we propose MoE pruning via Router bias learning and expert Approximation (MoRA), a framework for structured MoE expert pruning. As shown in Figure 1 (c), we first

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

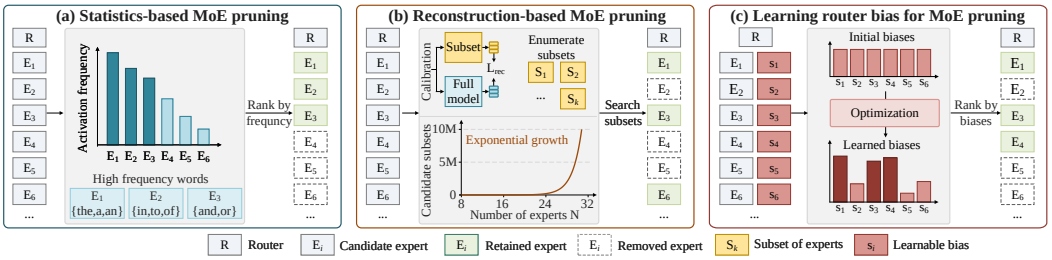


Figure 1: Comparison of three ways to select retained MoE experts. (a) Statistic-based pruning assigns each expert a fixed score based on routing statistics and retains the highest-scoring experts. (b) Reconstruction-based pruning evaluates candidate expert subsets using layer-output reconstruction error and selects the subset with the lowest reconstruction error. (c) MoRA adds a router bias to each expert to sharpen the routing probability distribution. These biases are optimized to establish an expert ranking for subsequent selection.

introduce a learnable router bias for each expert. These experts are originally pretrained under the load-balancing constraints. The biases can sharpen routing probability distributions over experts and serve as scores for selecting experts. To learn effective biases, we optimize them with the language-modeling (LM) loss and a routing-diversity regularizer. The former encourages higher biases for experts critical to predictive performance, while the latter discourages experts with similar routing preferences from receiving high biases simultaneously. Herein, similar routing preference means that experts tend to receive similar routing weights for the same tokens.

We also introduce an expert approximation mechanism in our MoRA as a post-pruning enhancement. Specifically, we assign a retained expert as a proxy for each pruned expert and apply an affine transformation to approximate the output of the pruned expert with the retained experts, thereby mitigating the performance degradation caused by pruning. We evaluate MoRA on Qwen3-30B-A3B Yang et al. (2025), DeepSeek-V2-Lite Liu et al. (2024), and Moonlight-16B-A3B Liu et al. (2025) under two deployment budgets that remove 25% and 50% of the routed experts in each MoE layer. We evaluate zero-shot downstream performance on nine benchmarks. Further analysis reveals that the association between the learned router biases and predictive performance is stronger in models with more experts and in deeper MoE layers.

Our main contributions are as follows:

- We propose MoRA, a structured MoE expert pruning framework that learns router biases using the LM loss and a routing-diversity regularizer. The learned biases serve as scores for selecting experts with high predictive importance and diverse routing preferences.
- We propose expert approximation as a post-pruning enhancement, which uses lightweight affine transformations to approximate the outputs of pruned experts with the retained experts. It effectively mitigates the performance degradation caused by pruning.
- We evaluate MoRA on Qwen3-30B-A3B, DeepSeek-V2-Lite, and Moonlight-16B-A3B at 25% and 50% routed-expert pruning. Results on nine zero-shot benchmarks show that MoRA outperforms other pruning baselines while better preserving the predictive performance.

2 Related Work

Sparse MoE models. Sparse MoE models increase model capacity while limiting per-token computation by activating only a subset of experts (Shazeer et al., 2017). Large-scale systems such as GShard (Lepikhin et al., 2020) and Switch Transformers (Fedus et al., 2022) established routing and training recipes for this regime. Despite sparse activation, storing all expert parameters remains memory-intensive. This motivates expert pruning to reduce memory requirements while preserving predictive performance.

MoE Pruning. Weight-level pruning methods, such as SparseGPT, Wanda, and MoE-Pruner, can reduce MoE parameters by pruning individual weights within experts (Frantar & Alistarh, 2023;

Sun et al., 2024; Xie et al., 2024). However, these unstructured pruning methods directly reduce storage requirements and typically require specialized hardware support to achieve inference speedup. Expert-level pruning instead removes entire experts, directly reducing storage requirements. Early methods rank experts using activation frequency and gating scores (Koishekenov et al., 2023). NAEE (Lu et al., 2024) selects expert subsets that minimize the reconstruction loss between the original and pruned layer outputs. However, enumerating the subsets leads to expensive computation as the number of experts and the pruning budget increase. STUN (Lee et al., 2025) avoids exhaustive subset evaluation through expert clustering and parameter-space approximations. Complementary approaches compress MoE models through expert merging (Li et al., 2024; Chen et al., 2025), or replacement with lightweight modules (Zhang et al., 2026; Hao et al., 2026). However, modifying or replacing expert functions can introduce additional approximation errors.

3 MoRA

In this section, we propose MoE pruning via Router bias learning and expert approximation (MoRA), a framework for structured MoE pruning. We introduce a learnable router bias for each expert to sharpen the routing probability distributions, and optimize these biases using the LM loss and a routing-diversity regularizer. The resulting biases serve as expert selection scores that balance predictive importance and routing diversity.

Then, MoRA performs expert approximation based on affine transformation as a post-pruning enhancement. Specifically, MoRA estimates the directed proxy fidelity between retained and removed experts. It fits a channel-wise affine transformation from each candidate retained-expert output to the corresponding removed-expert output and evaluates the resulting approximation quality. For each removed expert, MoRA selects the retained expert with the highest proxy fidelity as its proxy. MoRA then reuses the fitted affine map to approximate the output of pruned expert during inference.

3.1 Preliminary

Consider an MoE layer ℓ with E_ℓ routed experts. For a token representation h, the router selects a top-k_ℓ expert set $\mathcal{T}_\ell(h)$. The routed-expert output is:

$$y_\ell(h) = \sum_{e \in \mathcal{T}_\ell(h)} p_{\ell,e}(h) f_{\ell,e}(h), \tag{1}$$

where $f_{\ell,e}$ is the function implemented by expert e. $p_{\ell,e}(h)$ denotes the mixing weight applied to selected expert e.

3.2 MoE pruning: router bias learning

The key challenge of MoE pruning is to retain an expert set that preserves the performance of the original MoE model. A desirable retained set should satisfy two requirements. First, it should preserve as much of the next-token prediction capability as possible. Second, the retained experts should exhibit diverse routing preferences, allowing a limited number of retained experts to cover complementary routing behaviors.

Unlike methods pruning based on statistics or reconstruction, we propose a learning-based expert pruning mechanism to efficiently identify experts that are critical to model performance. During pretraining, sparse MoE models commonly regulate expert traffic through auxiliary load-balancing objectives (Yang et al., 2025; Liu et al., 2024) or expert-level correction biases (Liu et al., 2025) to mitigate routing collapse and encourage more balanced expert utilization. Building on this, we introduce an additional learnable router bias for each expert to sharpen routing probability distributions. These biases are optimized with the LM loss and a routing-diversity regularizer. The learned biases therefore can serve as retention scores that balance predictive performance and routing diversity.

3.2.1 Learnable Router Biases

MoRA augments the frozen pretrained router with one learnable router bias for each routed expert. For expert e in MoE layer ℓ, let $s_{\ell,e} \in \mathbb{R}$ denote its router bias. Given the hidden state h of a token,

we modify the corresponding pretrained logit as:

$$r'_{\ell,e}(h) = \frac{r_{\ell,e}(h) + \alpha_\ell s_{\ell,e}}{\tau}, \tag{2}$$

where $r_{\ell,e}(h)$ is the original router logit, $\alpha_\ell > 0$ controls the strength of the bias, and $\tau > 0$ is a calibration temperature. The original logit $r_{\ell,e}(h)$ remains dependent on the current token, whereas $s_{\ell,e}$ provides a global expert preference across all tokens. For a softmax router, let $p_s(e \mid h)$ denote the normalized routing probability induced by the adjusted logits $r'_{\ell,e}(h)$ before top-k selection. For any two experts e and e', their routing odds satisfy:

$$\frac{p_s(e \mid h)}{p_s(e' \mid h)} = \exp\left(\frac{r_{\ell,e}(h) - r_{\ell,e'}(h) + \alpha_\ell \left(s_{\ell,e} - s_{\ell,e'}\right)}{\tau}\right). \tag{3}$$

The learned bias therefore adjusts the router probability distributions. A larger $s_{\ell,e}$ leads to a higher routing probability for expert e relative to other experts.

In contrast, existing statistic-based pruning methods commonly estimate expert importance using activation frequency or average routing probability (Koishekenov et al., 2023; Xie et al., 2024). These methods can result in a high importance score when the expert is associated with common and easily predictable tokens. However, such experts may not be critical to the model's predictive performance since these easily predictable tokens generally produce smaller language-modeling losses and weaker gradient signals during training.

3.2.2 Optimizing the Biases for Predictive Utility and Routing Diversity

MoRA learns the router biases using the LM loss on the calibration corpus. The gradient of LM loss aggregates the effect of each expert on predictive performance across calibration tokens.

Specifically, consider a softmax router with normalized selected weights, and a locally fixed top-k set $\mathcal{T}_\ell(h)$. For a selected expert e, the contribution through token representation h to the gradient of LM loss with respect to its bias is:

$$G_{\ell,e}(h) = \frac{\alpha_\ell}{\tau} p_{\ell,e}(h) \left\langle g_\ell(h), f_{\ell,e}(h) - y_\ell(h) \right\rangle, \tag{4}$$

where $p_{\ell,e}(h)$ is the routing weight derived from the adjusted logits $r'_\ell(h)$, $f_{\ell,e}(h)$ is the output of expert e, and $y_\ell(h)$ is the routed-expert mixture defined in Equation 1. The vector $g_\ell(h) = \partial \mathcal{L}_{\mathrm{LM}} / \partial y_\ell(h)$ is computed from the full batch LM objective. The detailed derivation of the gradient of LM loss with respect to the router biases is provided in Appendix A.

A negative inner product indicates that moving this routed output toward expert e's output locally decreases the LM loss, yielding a gradient contribution that favors increasing its bias under gradient descent. A positive inner product gives the opposite contribution. The actual bias update depends on the contributions across all tokens, the diversity regularizer, and the optimizer.

Experts with similar routing preferences may receive high biases simultaneously, limiting the diversity of routing preferences represented in the retained set under a fixed expert budget. To address this, we introduce a routing-diversity regularizer based on the geometry of the pretrained router. The regularizer suppresses the expert with the lower bias in each similar pair, encouraging the retained set to cover more distinct routing preferences. Specifically, we measure the similarity between different experts using the cosine similarity of their corresponding pretrained router rows:

$$C_{\ell,e,e'} = \left[\max\left(0, \cos\left(w_{\ell,e}, w_{\ell,e'}\right)\right)\right]^2. \tag{5}$$

Here, $w_{\ell,e}$ denotes the pretrained router row associated with expert e in layer ℓ. For each pair with different biases, we apply the similarity penalty only to the lower-bias expert. We define the diversity loss as:

$$\mathcal{L}_{\mathrm{Div}} = \sum_{\ell \in \mathcal{M}} \sum_{e \neq e'} C_{\ell,e,e'} min(\sigma(s_{\ell,e}), \sigma(s_{\ell,e'})), \tag{6}$$

where $\mathcal{M}$ denotes the set of MoE layer indices. For a similar expert pair satisfying $\sigma(s_{\ell,e}) < \sigma(s_{\ell,e'})$, minimizing this loss suppresses $\sigma(s_{\ell,e})$ without pushing down $\sigma(s_{\ell,e'})$ through the same pair. This encourages priority separation among experts with overlapping routing preferences.

We optimize the learnable biases $s = \{s_{\ell,e}\}$ using a joint objective that combines the LM loss with the diversity regularization:

$$\mathcal{L}_{\text{bias}} = \mathcal{L}_{\text{LM}} + \lambda_{\text{d}} \mathcal{L}_{\text{Div}}, \tag{7}$$

where $\lambda_{\text{d}} \geq 0$ controls the strength of the diversity regularization.

3.2.3 Expert Selection from Learned Router Biases

After calibration, we rank the routed experts in each MoE layer according to their learned router biases to jointly consider the predictive performance and routing diversity. Let $\mathcal{E}_\ell = \{1, \ldots, E_\ell\}$ denote the original expert set in layer ℓ, and let B_ℓ denote the number of experts to retain. The retained and removed sets are:

$$S_\ell = \text{Top}_{B_\ell}\left(s_{\ell,1}, \ldots, s_{\ell,E_\ell}\right), \qquad \overline{S}_\ell = \mathcal{E} - S_\ell, \tag{8}$$

where Top_{B_ℓ} returns the indices of the B_ℓ largest values. Experts in $\overline{S}_\ell$ are physically removed. The learned biases are then discarded and introduce no inference overhead.

3.3 Post-pruning Enhancement: Expert Approximation

Although expert pruning reduces physical storage, removing experts discards their output contributions and may degrade model performance. To mitigate this degradation, MoRA fits a channel-wise affine map from each retained candidate to each removed target and evaluates its output-approximation fidelity. Each removed expert is assigned the retained candidate with the highest fidelity, and the same fitted affine map is then used to transform the selected proxy output during inference.

3.3.1 Directed Proxy Fidelity Estimation

We divide the calibration corpus into two non-overlapping subsets: a fit split $\mathcal{D}_{\text{fit}}$ and a scoring split $\mathcal{D}_{\text{score}}$. The fit split is used to estimate the pairwise affine maps, while the scoring split is used to evaluate their approximation fidelity. For a removed target expert $i \in \overline{S}_\ell$ and a retained candidate expert $j \in S_\ell$, MoRA predicts the target output using:

$$\widehat{f}_{\ell,i\leftarrow j}(h) = \widehat{a}_{\ell,i,j} \odot f_{\ell,j}(h) + \widehat{b}_{\ell,i,j}, \tag{9}$$

where $\widehat{a}_{\ell,i,j}$ and $\widehat{b}_{\ell,i,j}$ are the fitted channel-wise scale and bias. These parameters are estimated on $\mathcal{D}_{\text{fit}}$ using weighted ridge regression.

We derive target-conditioned token weights $\omega_{s,t,i}$ from the pretrained router, assigning larger weights to tokens that are more relevant to target expert i. Then we evaluate the affine predictor on $\mathcal{D}_{\text{score}}$ using the weighted coefficient of determination:

$$R^2_{\ell,i,j} = 1 - \frac{\sum_{(s,t)\in\mathcal{D}_{\text{score}}} \omega_{s,t,i} \left\| f_{\ell,i}(h_{s,t}) - \widehat{f}_{\ell,i\leftarrow j}(h_{s,t}) \right\|_2^2}{\sum_{(s,t)\in\mathcal{D}_{\text{score}}} \omega_{s,t,i} \left\| f_{\ell,i}(h_{s,t}) - \mu^{\text{score}}_{\ell,i} \right\|_2^2}, \qquad q_{\ell,i,j} = \max\left(0, R^2_{\ell,i,j}\right), \tag{10}$$

where $\mu^{\text{score}}_{\ell,i}$ is the weighted mean output of target expert i on the scoring split. A larger $q_{\ell,i,j}$ indicates that the affine-transformed output of retained expert j more accurately approximates the output variation of removed expert i. The relation is directional because the affine map predicts i from j, and both the predictor and token weights are conditioned on target expert i.

Each removed expert is assigned the retained candidate with the highest proxy fidelity:

$$j_\ell(i) = \arg\max_{j\in S_\ell} q_{\ell,i,j}. \tag{11}$$

The computation of the target-conditioned weights and the closed-form weighted ridge solution are provided in Appendix B.

3.3.2 Expert Approximation based on Affine Transformation

After the proxy assignment is fixed, MoRA performs the expert approximation based on affine transformation. For a removed expert i and its assigned retained proxy $j_\ell(i)$, MoRA replaces the

removed expert output with $\widehat{f}_{\ell,i\leftarrow j_\ell(i)}(h)$ from Equation 9. To limit the effect of unreliable affine approximations, we enable this proxy only when the selected proxy fidelity satisfies $q_{\ell,i,j_\ell(i)} > q_{\min}$, where $q_{\min}$ is a prescribed threshold.

No additional optimization is required after proxy assignment. The exported model stores only the retained experts, the target-proxy mappings, and the fitted channel-wise affine parameters. The removed expert parameters and proxy fidelity scores are discarded. Each pruned expert requires only $2d_o$ affine parameters, where d_o is the expert output dimension, corresponding to one scale and one bias per output channel. This is far fewer than the number of parameters in the original expert.

4 Experiments

4.1 Experimental Setup

Models and pruning budgets. We evaluate Qwen3-30B-A3B (Yang et al., 2025), DeepSeek-V2-Lite (Liu et al., 2024), and Moonlight-16B-A3B (Liu et al., 2025). Qwen3 has 48 MoE layers with 128 routed experts per layer and activates eight experts per token. DeepSeek-V2-Lite and Moonlight each have 26 MoE layers with 64 routed experts, two shared experts, and six activated routed experts per token. We remove either 25% or 50% of the routed experts in every MoE layer, retaining 96/64 experts for Qwen3 and 48/32 experts for the other two models. Shared experts and all non-expert parameters are preserved, and the original routing top-k is unchanged.

Baselines. **Activation Count** ranks routed experts according to how often they are selected by the pretrained router on the calibration corpus and retains the highest-ranked experts in each layer. **Average Gate Mass** ranks experts by their normalized top-k routing weights averaged over all calibration tokens, incorporating both selection frequency and routing confidence (Koishekenov et al., 2023; Xie et al., 2024). **MoNE** combines expert access frequency with output fluctuation to identify important experts and replaces less important experts with lightweight approximations (Zhang et al., 2026). **MC-SMoE** groups experts according to the similarity of their router-logit patterns and constructs representative experts through activation-frequency-weighted merging (Li et al., 2024). **HC-SMoE** hierarchically clusters experts using their output features and merges the experts within each cluster (Chen et al., 2025).

Implementation Details. Following the common practice in existing MoE pruning methods (Ling et al., 2024; Zhang et al., 2026; Lu et al., 2024), we conduct experiments using C4 (Raffel et al., 2020) as the calibration dataset. The router biases are learned on C4 with 1000 iterations for Qwen3-30B-A3B, DeepSeek-V2-Lite and Moonlight-16B-A3B. For post-pruning enhancement, we divide the sampled C4 calibration corpus into two non-overlapping subsets of equal size: Each subset contains 64 sequences of 1024 tokens. We set the hyperparameters $\lambda_{\mathrm{d}} = 0.8$ and $q_{\min} = 0.4$. The sensitivity analysis of these hyperparameters is provided in Appendix C.

Evaluation benchmarks. Following the previous MoE pruning methods (Zhang et al., 2026; Lu et al., 2024; Ling et al., 2024; An et al., 2024), we use lm-evaluation-harness (Gao et al., 2021) for zero-shot evaluation on BoolQ (Clark et al., 2019), RTE (Wang et al., 2019), HellaSwag (Zellers et al., 2019), WinoGrande (Sakaguchi et al., 2021), PIQA (Bisk et al., 2020), OpenBookQA (Mihaylov et al., 2018), ARC-Easy (Clark et al., 2018), ARC-Challenge (Clark et al., 2018), and MMLU. HellaSwag, PIQA, OpenBookQA, ARC-Easy, and ARC-Challenge use `acc_norm`; the other tasks use `acc`. Avg. is the mean of these nine percentages.

4.2 Main Results

Comparison with baselines. Tables 1, 2, and 3 compare MoRA with five pruning baselines on nine zero-shot benchmarks. MoRA achieves the highest average score across all three architectures at both 25% and 50% pruning ratios. Compared with the strongest baseline in each setting, MoRA improves the average score by 1.14, 0.43, and 1.01 percentage points at 25% pruning, and by 6.43, 0.56, and 0.97 points at 50% pruning, on Qwen3-30B-A3B, DeepSeek-V2-Lite, and Moonlight-16B-A3B, respectively. We attribute the weaker performance of HC-SMoE on Qwen3 to a potential mismatch between its clustering criterion, based on average output similarity, and the high degree of expert specialization in Qwen3. Merging experts under this aggregate criterion may reduce the diversity of expert outputs.

Table 1: Zero-shot results (%) on Qwen3-30B-A3B with C4 as the calibration dataset. Bold marks the best result among MoRA and the baselines for each column at each pruning ratio.

Pruning ratio	Method	ARC-c	ARC-e	BoolQ	HellaSwag	MMLU	OBQA	PIQA	RTE	WinoGrande	Avg.
0%	Qwen3-30B-A3B	56.31	78.87	88.47	77.66	77.81	45.00	80.58	82.67	70.56	73.10
25%	Activation Count	55.48	76.83	87.35	76.44	72.83	43.00	79.76	79.06	68.98	71.08
	Average Gate Mass	55.74	77.12	87.47	76.56	72.71	43.60	80.12	79.12	69.61	71.34
	MoNE	54.61	76.68	86.79	74.69	**72.87**	**44.80**	78.84	**80.87**	69.85	71.11
	MC-SMoE	52.52	**80.86**	87.38	77.42	70.62	43.80	79.82	79.23	70.59	71.36
	HC-SMoE	47.18	71.84	83.55	61.98	64.35	38.00	72.14	80.50	65.27	64.98
	MoRA	**57.42**	80.18	**88.81**	**79.28**	72.32	**44.80**	**81.18**	76.17	**72.38**	**72.50**
50%	Activation Count	36.26	52.06	82.23	65.57	52.29	37.60	69.91	58.48	63.06	57.50
	Average Gate Mass	36.26	54.84	82.66	66.76	52.98	36.40	72.03	58.84	62.04	58.09
	MoNE	44.11	63.22	**82.81**	55.97	50.40	35.80	73.94	64.98	63.38	59.40
	MC-SMoE	23.80	32.37	75.23	62.49	22.95	33.20	60.06	55.60	62.75	47.61
	HC-SMoE	30.03	47.39	72.29	35.61	41.33	27.40	56.47	58.12	53.43	46.90
	MoRA	**46.25**	**69.07**	82.51	**75.20**	**53.14**	**41.80**	**80.47**	**75.09**	**68.98**	**65.83**

Table 2: Zero-shot results (%) on DeepSeek-V2-Lite with C4 as the calibration dataset. Bold marks the best result among MoRA and the baselines for each column at each pruning ratio.

Pruning ratio	Method	ARC-c	ARC-e	BoolQ	HellaSwag	MMLU	OBQA	PIQA	RTE	WinoGrande	Avg.
0%	DeepSeek-V2-Lite	49.23	76.60	79.94	77.92	54.97	43.80	80.03	60.65	71.27	66.05
25%	Activation Count	45.14	73.53	71.10	73.38	44.58	42.80	79.60	59.57	70.25	62.22
	Average Gate Mass	45.82	74.41	70.52	75.26	46.39	41.80	79.71	60.29	69.22	62.60
	MoNE	45.59	72.81	**73.68**	76.28	**47.74**	43.20	79.71	60.37	69.88	63.25
	MC-SMoE	39.08	66.16	69.20	67.59	36.20	37.40	76.99	57.04	69.30	57.66
	HC-SMoE	44.88	72.10	73.37	74.49	47.29	40.60	78.78	**61.37**	**71.03**	62.66
	MoRA	**45.99**	**77.19**	72.26	**77.24**	47.73	**44.40**	**80.20**	58.12	70.01	**63.68**
50%	Activation Count	28.33	41.29	54.07	54.97	28.03	33.40	65.72	51.63	58.96	46.27
	Average Gate Mass	35.24	56.27	50.58	67.01	23.46	36.80	76.06	49.82	63.62	50.98
	MoNE	34.73	55.98	**67.92**	67.89	22.93	36.40	76.55	53.07	**67.25**	53.64
	MC-SMoE	28.16	43.98	57.19	48.56	22.96	29.60	66.00	55.23	60.77	45.83
	HC-SMoE	33.69	60.57	66.51	60.75	**34.32**	34.56	71.58	**57.04**	65.56	53.84
	MoRA	**38.65**	**63.17**	58.96	**71.47**	23.25	**42.20**	**77.80**	51.63	62.51	**54.40**

Table 3: Zero-shot results (%) on Moonlight-16B-A3B with C4 as the calibration dataset. Bold marks the best result among MoRA and the baselines for each column at each pruning ratio.

Pruning ratio	Method	ARC-c	ARC-e	BoolQ	HellaSwag	MMLU	OBQA	PIQA	RTE	WinoGrande	Avg.
0%	Moonlight-16B-A3B	58.19	82.58	80.15	78.37	67.31	45.60	80.96	66.06	71.43	70.07
25%	Activation Count	55.12	80.01	78.17	76.96	41.45	45.40	81.18	60.73	70.48	65.50
	Average Gate Mass	**55.24**	**80.51**	78.32	**77.11**	48.29	**46.80**	**81.50**	60.01	70.43	66.47
	MoNE	53.63	78.67	77.56	77.00	53.35	45.20	80.23	59.01	**71.43**	66.23
	MC-SMoE	48.55	76.73	78.10	75.27	39.74	44.80	80.31	57.40	70.25	63.46
	HC-SMoE	41.72	70.45	70.34	53.93	53.80	36.40	68.99	59.57	56.59	56.87
	MoRA	53.15	79.12	**79.57**	74.76	**56.79**	43.00	80.79	**70.76**	69.38	**67.48**
50%	Activation Count	**37.88**	55.39	58.87	70.22	23.10	37.40	76.55	53.79	68.67	53.54
	Average Gate Mass	37.23	56.26	62.53	**71.46**	23.14	37.40	76.78	55.40	**68.80**	54.33
	MoNE	34.98	57.95	68.38	67.10	22.98	**38.00**	**77.67**	56.84	65.64	54.39
	MC-SMoE	26.71	43.65	60.49	46.37	23.18	29.80	64.69	52.71	53.35	44.55
	HC-SMoE	26.02	50.63	61.31	34.96	**29.93**	27.80	62.73	53.43	54.06	44.54
	MoRA	36.43	**65.32**	**70.92**	61.47	28.66	35.40	77.31	**61.73**	61.01	**55.36**

Compression efficiency. As shown in Table 4, MoRA preserves most of the original performance at 25% expert pruning, with average score drops of only 0.60, 2.37, and 2.59 percentage points on Qwen3-30B-A3B, DeepSeek-V2-Lite, and Moonlight-16B-A3B, respectively. The affine transformations require less than 0.08% of the retained parameter count across all evaluated settings. Their additional inference FLOPs are bounded by 0.0005% of those of the original model, enabling post-pruning enhancement with negligible parameter and computational overhead.

Table 4: Compression efficiency of MoRA. We report the model parameter counts and storage, and the additional overhead of the affine transformations.

Model	Pruning ratio	Retained experts	Model params. (B)	Model storage (GiB)	Affine params. (B)	Affine storage (GiB)
Qwen3-30B-A3B	0%	128	30.532	56.871	0	0
	25%	96	23.284	43.371	0.0063	0.012
	50%	64	16.037	29.871	0.0126	0.023
DeepSeek-V2-Lite	0%	64	15.706	29.256	0	0
	25%	48	12.108	22.552	0.0017	0.0032
	50%	32	8.509	15.849	0.0034	0.0063
Moonlight-16B-A3B	0%	64	15.960	29.728	0	0
	25%	48	12.361	23.025	0.0017	0.0032
	50%	32	8.763	16.322	0.0034	0.0063

4.3 Ablation Study

Effect of the key components. We evaluate the effect of the key components of MoRA, containing two objectives and expert approximation. As shown in Table 5, jointly optimizing the LM and diversity losses achieves average scores of 72.17 and 65.30 at 25% and 50% pruning, respectively. Removing the diversity loss reduces these scores by 4.84 and 7.22 points, while removing the LM loss causes drops of 4.69 and 10.73 points. The larger degradation without the LM loss at 50% pruning suggests that predictive utility becomes more important under a tighter retention budget. The diversity objective discourages overlapping routing preferences but does not assess the contribution to next-token prediction of each expert. Consequently, selecting experts solely for diversity may discard predictively useful experts, with fewer retained experts available to compensate for their removal at higher pruning ratios.

Table 5: Component ablation on Qwen3-30B-A3B. Avg. denotes the mean score across nine benchmarks.

Components			Avg.	
$\mathcal{L}_{\text{LM}}$	$\mathcal{L}_{\text{Div}}$	Expert Approximation	25%	50%
✓	–	–	67.33	58.08
–	✓	–	67.48	54.57
✓	✓	–	72.17	65.30
✓	✓	✓	**72.50**	**65.83**

Pruning with learned biases versus adjusted routing probabilities.

Table 6 shows that computing gate mass from the bias-adjusted router improves average performance over the original gate mass in all six settings. However, ranking experts by adjusted gate mass still underperforms direct bias-based selection, particularly at 50% pruning on Qwen3 and DeepSeek-V2-Lite, where the gaps reach 6.49 and 2.64 percentage points, respectively. These results indicate that aggregating routing probabilities after bias calibration remains less effective than directly ranking experts by the learned biases, particularly at high pruning ratios. This suggests that expert utilization, even under the calibrated router, does not fully capture predictive importance.

Table 6: Mean scores (%) across nine zero-shot benchmarks. Average Gate Mass uses the pretrained router, whereas r' Gate Mass uses the bias-adjusted routing probabilities. We report the average score of MoRA after pruning without expert approximation.

Model	Pruning ratio	Average Gate Mass	r' Gate Mass	MoRA w/o Expert Approximation
Qwen3-30B-A3B	25%	71.34	71.55	**72.17**
	50%	58.09	58.81	**65.30**
DeepSeek-V2-Lite	25%	62.60	62.79	**63.25**
	50%	50.98	51.38	**54.02**
Moonlight-16B-A3B	25%	66.47	67.07	**67.19**
	50%	54.33	54.58	**54.82**

4.4 Discussion

4.4.1 Learned Router Biases and Expert Deletion Sensitivity

We investigate whether learned router biases reflect the model's sensitivity to expert removal. On Qwen3-30B-A3B, we use a randomly sampled 10% subset of the C4 calibration tokens and measure the increase in next-token prediction loss after masking one expert at a time. All pretrained parameters remain fixed, and the router recomputes top-k selection over the remaining experts.

Within each layer, we analyze the top 50% of experts ranked by learned bias and, separately, by activation frequency. Figure 2 presents scatter plots and Spearman correlations between each score and the loss increase caused by expert deletion. Learned biases exhibit a clear positive correlation with deletion sensitivity, with stronger correlations in deeper MoE layers, whereas activation frequency shows a less consistent relationship. Additional results for DeepSeek-V2-Lite and Moonlight-16B-A3B are provided in Appendix F.1.

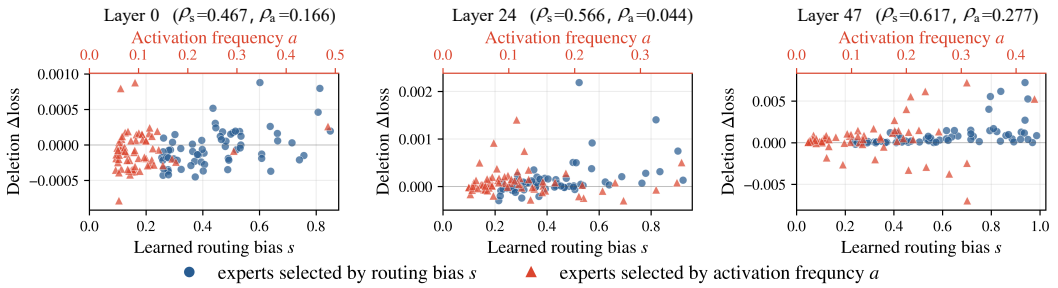
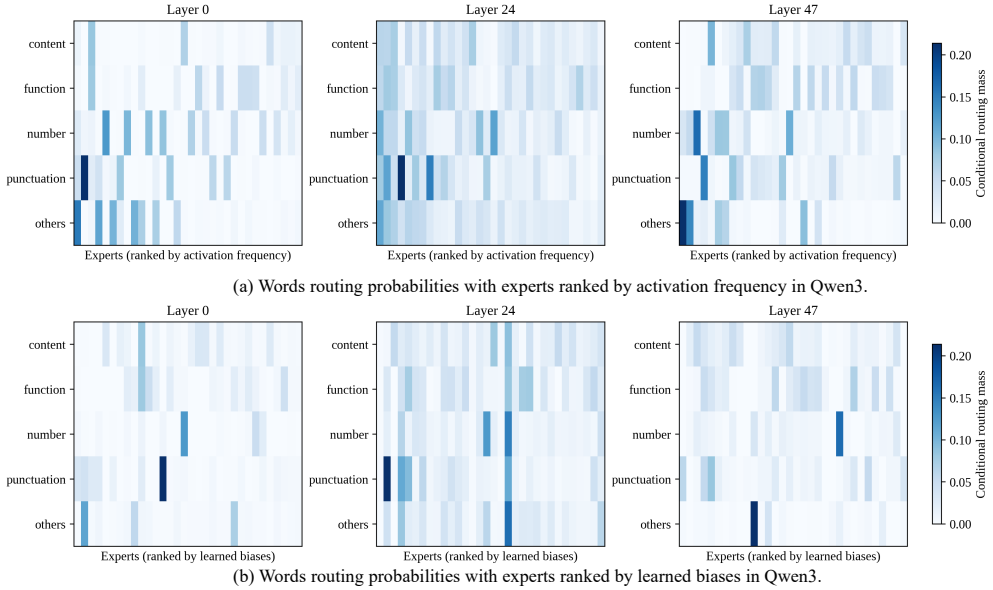


Figure 2: Relationship between expert scores and deletion-induced loss increases on Qwen3-30B-A3B. Each analysis uses the top 50% of experts within a layer ranked by the corresponding score. Spearman correlations for learned biases and activation frequency are denoted by ρ_s and ρ_a, respectively.

4.4.2 Token-category routing analysis

We analyze token-category routing patterns in Qwen3-30B-A3B using 128 samples from C4. Specifically, we record expert assignments under the pretrained router and visualize token-category routing probabilities with experts ordered by learned bias or activation frequency. Figure 3 shows that content words tend to be routed to experts with higher learned biases, whereas numbers and punctuation tend to be routed to experts with higher activation frequencies. Annotation and measurement details are provided in Appendix E. Additional results for DeepSeek-V2-Lite and Moonlight-16B-A3B are provided in Appendix F.2.



(a) Words routing probabilities with experts ranked by activation frequency in Qwen3.

(b) Words routing probabilities with experts ranked by learned biases in Qwen3.

Figure 3: Heatmap visualization of token-category routing probabilities in Qwen3 at Layers 0, 24, and 47. Rows correspond to word categories, and darker cells indicate larger routing probabilities. Experts are ordered from left to right by decreasing activation frequency (a) and learned biases (b).

5 Conclusion

In this paper, we propose MoRA, a framework for structured MoE expert pruning. We introduce a learnable router bias for each expert that sharpens the routing probability distributions. Experts that are critical to the predictive performance tend to receive higher learned biases and are therefore prioritized for retention. The retained experts are then used to approximate the outputs of the pruned experts through channel-wise affine transformations. Extensive experiments across three MoE architectures demonstrate that MoRA outperforms the evaluated baselines and preserves predictive performance with minimal additional parameter overhead.

AI USE STATEMENT

Generative AI tools were used to assist with drafting, polishing sections of the manuscript and assisting with programming. All analyses, editorial decisions, and conclusions in the paper were made independently by the authors, who take full responsibility for the methodology, results, and conclusions presented in this paper.

REPRODUCIBILITY STATEMENT

Section 4.1 describes the evaluated models, pruning budgets, calibration data, implementation settings, and evaluation protocol. Appendix A provides the derivation of the gradient of LM loss with respect to router biases, while Appendix B details the target-conditioned token weights, affine fitting procedure, and proxy assignment. We provide the token-category analysis protocol in Appendix E and additional experiments on hyperparameter sensitivity and calibration-corpus robustness in Appendices C and D. We will release the code to facilitate reproduction of our results.

REFERENCES

Yongqi An, Xu Zhao, Tao Yu, Ming Tang, and Jinqiao Wang. Fluctuation-based adaptive structured pruning for large language models. In *Proceedings of the AAAI conference on artificial intelligence*, volume 38, pp. 10865–10873, 2024.

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

I-Chun Chen, Hsu-Shen Liu, Wei-Fang Sun, Chen-Hao Chao, Yen-Chang Hsu, and Chun-Yi Lee. Retraining-free merging of sparse moe via hierarchical clustering. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267, pp. 8594–8620. PMLR, 2025.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. Boolq: Exploring the surprising difficulty of natural yes/no questions. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 2924–2936, 2019.

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

Zican Dong, Han Peng, Peiyu Liu, Xin Zhao, Dong Wu, Feng Xiao, and Zhifeng Wang. Domain-specific pruning of large mixture-of-experts models with few-shot demonstrations. *Advances in Neural Information Processing Systems*, 38:103552–103577, 2026.

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

Elias Frantar and Dan Alistarh. Sparsegpt: Massive language models can be accurately pruned in one-shot. In *International conference on machine learning*, pp. 10323–10337. PMLR, 2023.

Leo Gao, Jonathan Tow, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Kyle McDonell, Niklas Muennighoff, et al. A framework for few-shot language model evaluation. *Zenodo*, 2021.

Jiawei Hao, Zhiwei Hao, Jianyuan Guo, Li Shen, Yong Luo, Han Hu, and Dan Zeng. Lightmoe: Reducing mixture-of-experts redundancy through expert replacing. *arXiv preprint arXiv:2603.12645*, 2026.

Yeskendir Koishekenov, Alexandre Berard, and Vassilina Nikoulina. Memory-efficient nllb-200: Language-specific expert pruning of a massively multilingual machine translation model. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 3567–3585, 2023.

Jaeseong Lee, Seung-won Hwang, Aurick Qiao, Daniel F Campos, Zhewei Yao, and Yuxiong He. Stun: Structured-then-unstructured pruning for scalable moe pruning. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 13660–13676, 2025.

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

Pingzhi Li, Zhenyu Zhang, Prateek Yadav, Yi-Lin Sung, Yu Cheng, Mohit Bansal, and Tianlong Chen. Merge, then compress: Demystify efficient smoe with hints from its routing policy. In *International Conference on Learning Representations*, volume 2024, pp. 14234–14256, 2024.

Gui Ling, Ziyang Wang, Yuliang Yan, and Qingwen Liu. Slimgpt: Layer-wise structured pruning for large language models. *Advances in Neural Information Processing Systems*, 37:107112–107137, 2024.

Aixin Liu, Bei Feng, Bin Wang, Bingxuan Wang, Bo Liu, Chenggang Zhao, Chengqi Dengr, Chong Ruan, Damai Dai, Daya Guo, et al. Deepseek-v2: A strong, economical, and efficient mixture-of-experts language model. *arXiv preprint arXiv:2405.04434*, 2024.

Jingyuan Liu, Jianlin Su, Xingcheng Yao, Zhejun Jiang, Guokun Lai, Yulun Du, Yidao Qin, Weixin Xu, Enzhe Lu, Junjie Yan, et al. Muon is scalable for llm training. *arXiv preprint arXiv:2502.16982*, 2025.

Xudong Lu, Qi Liu, Yuhui Xu, Aojun Zhou, Siyuan Huang, Bo Zhang, Junchi Yan, and Hongsheng Li. Not all experts are equal: Efficient expert pruning and skipping for mixture-of-experts large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 6159–6172, 2024.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. Can a suit of armor conduct electricity? a new dataset for open book question answering. In *Proceedings of the 2018 conference on empirical methods in natural language processing*, pp. 2381–2391, 2018.

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

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

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

Mingjie Sun, Zhuang Liu, Anna Bair, and Zico Kolter. A simple and effective pruning approach for large language models. In *International Conference on Learning Representations*, volume 2024, pp. 4942–4964, 2024.

Yury Tokpanov, Paolo Glorioso, Quentin Anthony, and Beren Millidge. Zyda-2: a 5 trillion token high-quality dataset. *arXiv preprint arXiv:2411.06068*, 2024.

Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. Superglue: A stickier benchmark for general-purpose language understanding systems. *Advances in neural information processing systems*, 32, 2019.

Yanyue Xie, Zhi Zhang, Ding Zhou, Cong Xie, Ziang Song, Xin Liu, Yanzhi Wang, Xue Lin, and An Xu. Moe-pruner: Pruning mixture-of-experts large language model using the hints from its router. *arXiv preprint arXiv:2410.12013*, 2024.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. Hellaswag: Can a machine really finish your sentence? In *Proceedings of the 57th annual meeting of the association for computational linguistics*, pp. 4791–4800, 2019.

Geng Zhang, Han Yuxuan, Yuxuan Lou, Yiqi Zhang, Wangbo Zhao, and Yang You. Mone: Replacing redundant experts with lightweight novices for structured pruning of moe. In *International Conference on Learning Representations*, volume 2026, pp. 39983–40009, 2026.

APPENDIX OVERVIEW

- Section A: Derivation of the Gradient of LM Loss with Respect to Router Biases.
- Section B: Details of Directed Proxy Fidelity and Assignment.
- Section C: Analysis of Hyperparameters.
- Section D: Robustness to the Calibration Corpus.
- Section E: Details of Token-Category Routing Analysis.
- Section F: Additional Expert Importance and Routing Analyses.

A DERIVATION OF THE GRADIENT OF LM LOSS WITH RESPECT TO ROUTER BIASES

This section derives how the LM loss guides router biases learning, showing that each token's contribution to the gradient of LM loss depends on whether shifting the routed output toward an expert's output locally decreases the loss. Throughout the derivation, we condition on a locally fixed top-k expert set $\mathcal{T}_\ell(h)$ and hold α_ℓ and τ constant. When differentiating with respect to $s_{\ell,e}$, the input h and expert outputs $f_{\ell,j}(h)$ are fixed because they do not depend on the router biases of the same layer. The routed output $y_\ell(h)$ excludes any shared-expert contribution.

Softmax routing with selected-weight renormalization. We first consider a softmax router whose selected weights sum to one, with no additional output scaling. Given the adjusted logits $r'_{\ell,j}(h) = (r_{\ell,j}(h) + \alpha_\ell s_{\ell,j})/\tau$, the selected routing weights and routed output are:

$$p_{\ell,j}(h) = \frac{\exp(r'_{\ell,j}(h))}{\sum_{m\in\mathcal{T}_\ell(h)} \exp(r'_{\ell,m}(h))}, \qquad y_\ell(h) = \sum_{j\in\mathcal{T}_\ell(h)} p_{\ell,j}(h) f_{\ell,j}(h), \tag{12}$$

where the expression for $p_{\ell,j}(h)$ applies to selected experts, and we set $p_{\ell,j}(h) = 0$ otherwise. Applying the softmax Jacobian and the chain rule gives, for $j, e \in \mathcal{T}_\ell(h)$:

$$\frac{\partial p_{\ell,j}(h)}{\partial s_{\ell,e}} = \frac{\alpha_\ell}{\tau} p_{\ell,j}(h) \left(\delta_{je} - p_{\ell,e}(h)\right), \tag{13}$$

where δ_{je} is the Kronecker delta. The corresponding output derivative is:

$$\frac{\partial y_\ell(h)}{\partial s_{\ell,e}} = \sum_{j\in\mathcal{T}_\ell(h)} f_{\ell,j}(h) \frac{\partial p_{\ell,j}(h)}{\partial s_{\ell,e}} = \frac{\alpha_\ell}{\tau} p_{\ell,e}(h) \left(f_{\ell,e}(h) - y_\ell(h)\right). \tag{14}$$

Let $g_\ell(h) = \partial \mathcal{L}_{\mathrm{LM}}/\partial y_\ell(h)$ denote the gradient of LM loss, computed over the full calibration batch, with respect to this routed output. It includes all downstream paths through which this representation affects the loss. The contribution through token representation h to the gradient of LM loss with respect to the shared bias is:

$$G_{\ell,e}(h) := \left\langle g_\ell(h), \frac{\partial y_\ell(h)}{\partial s_{\ell,e}} \right\rangle = \frac{\alpha_\ell}{\tau} p_{\ell,e}(h) \left\langle g_\ell(h), f_{\ell,e}(h) - y_\ell(h) \right\rangle. \tag{15}$$

For an unselected expert, this contribution is zero while the selected set remains fixed. Since each bias is shared across tokens, the complete gradient of LM loss with respect to that bias is:

$$\frac{\partial \mathcal{L}_{\mathrm{LM}}}{\partial s_{\ell,e}} = \sum_{v\in\mathcal{V}} G_{\ell,e}(h_{\ell,v}), \tag{16}$$

where $\mathcal{V}$ indexes the input token occurrences in the calibration batch and $h_{\ell,v}$ is the representation at occurrence v. Any averaging factor in the LM loss is already included in $g_\ell(h)$.

A negative inner product in Equation 15 yields a contribution to the gradient of LM loss that favors increasing the expert's bias under gradient descent; a positive inner product gives the opposite contribution. The actual update depends on the sum of these contributions, the diversity regularizer, and the optimizer. Thus, frequent activation alone does not determine an expert's learned bias.

Softmax routing without selected-weight renormalization. Some routers compute softmax probabilities over all experts and directly use the selected probabilities as mixing weights. In this case, $p_{\ell,j}(h)$ denotes the full-softmax probability for every expert, including those outside the selected set:

$$p_{\ell,j}(h) = \frac{\exp(r'_{\ell,j}(h))}{\sum_{m=1}^{E_\ell} \exp(r'_{\ell,m}(h))}, \qquad y_\ell(h) = \sum_{j \in \mathcal{T}_\ell(h)} p_{\ell,j}(h) f_{\ell,j}(h). \tag{17}$$

With no additional output scaling, the output derivative for any expert e is:

$$\frac{\partial y_\ell(h)}{\partial s_{\ell,e}} = \frac{\alpha_\ell}{\tau} p_{\ell,e}(h) \left(\mathbf{1}[e \in \mathcal{T}_\ell(h)] \, f_{\ell,e}(h) - y_\ell(h) \right). \tag{18}$$

For a selected expert, the contribution to the gradient of LM loss has the same form as Equation 15. For an unselected expert, its bias still affects the full-softmax denominator, so its contribution to the gradient of LM loss is generally nonzero.

Sigmoid routing with selected-weight normalization. For a sigmoid router, let $u_{\ell,j}(h) = \sigma(r'_{\ell,j}(h))$ denote the score of expert j. Normalizing these scores within the selected set gives:

$$p_{\ell,j}(h) = \frac{u_{\ell,j}(h)}{\sum_{m \in \mathcal{T}_\ell(h)} u_{\ell,m}(h)}, \qquad \bar{y}_\ell(h) = \sum_{j \in \mathcal{T}_\ell(h)} p_{\ell,j}(h) f_{\ell,j}(h). \tag{19}$$

Here, the expression for $p_{\ell,j}(h)$ applies to selected experts, and $p_{\ell,j}(h) = 0$ otherwise. If the router uses a fixed positive scaling factor κ_ℓ, the routed output is $y_\ell(h) = \kappa_\ell \bar{y}_\ell(h)$. Applying the chain rule gives the contribution to the gradient of LM loss for a selected expert e:

$$G_{\ell,e}(h) = \kappa_\ell \frac{\alpha_\ell}{\tau} p_{\ell,e}(h) \left(1 - u_{\ell,e}(h)\right) \left\langle g_\ell(h), f_{\ell,e}(h) - \bar{y}_\ell(h) \right\rangle. \tag{20}$$

The contribution is zero for unselected experts under fixed membership. When no output scaling is applied, $\kappa_\ell = 1$. The positive sigmoid and scaling factors preserve the interpretation of the inner product as determining the sign of the local contribution, although the resulting gradient of LM loss need not match that of a softmax router.

B Details of Directed Proxy Fidelity and Assignment

This section details the target-conditioned token weights, affine fitting, and proxy assignment used in Section 3.3.1. We use two non-overlapping calibration subsets, $\mathcal{D}_{\text{fit}}$ and $\mathcal{D}_{\text{score}}$, to fit the affine predictors and evaluate their approximation fidelity, respectively. All pretrained parameters remain frozen throughout this procedure.

Target-conditioned token weights. For a fixed MoE layer ℓ, let $z_{s,t,e}$ denote the pretrained router logit of expert e for token position t in sequence s, and let $\mathcal{V}_s$ denote the valid token positions in that sequence. We omit the layer index from the logits and token weights for brevity. Using the pretrained logits without the learned router biases, we define the relevance $\rho_{s,t,i}$ of token (s,t) to target expert i. For a softmax router, this relevance is:

$$\rho_{s,t,i} = \frac{\exp(z_{s,t,i}/T)}{\sum_{e=1}^{E_\ell} \exp(z_{s,t,e}/T)}. \tag{21}$$

For a sigmoid router, we normalize the sigmoid scores across all routed experts:

$$\rho_{s,t,i} = \frac{\sigma(z_{s,t,i}/T)}{\sum_{e=1}^{E_\ell} \sigma(z_{s,t,e}/T)}, \tag{22}$$

where $T > 0$ is the temperature used in relevance estimation for expert approximation, E_ℓ is the number of routed experts in the pretrained layer, and σ denotes the sigmoid function. These relevance scores are computed over all routed experts before top-k selection and are distinct from the mixing weights applied to selected experts during the model forward pass.

We convert these relevance scores into target-conditioned token weights by normalizing within each sequence:

$$\omega_{s,t,i} = \frac{\rho_{s,t,i}^{p}}{\sum_{u \in \mathcal{V}_s} \rho_{s,u,i}^{p}}, \qquad \sum_{t \in \mathcal{V}_s} \omega_{s,t,i} = 1. \tag{23}$$

Here, $p > 0$ is a scalar weighting exponent, distinct from the routing weight $p_{\ell,e}(h)$ in the main text. Larger values of p concentrate the weights on tokens with higher relevance to target expert i. Normalization within each sequence assigns equal total weight to each sequence regardless of its number of valid tokens. We use $T = 2.0$ and $p = 2.0$, and compute the weights independently on the fit and scoring splits. The weights depend on the target expert i, but not on the candidate proxy j.

Weighted affine fitting. For a target expert i and a retained candidate proxy j, let n index valid token positions (s, t) in the fit split, and write $\omega_{n,i} = \omega_{s,t,i}$. We evaluate both experts on the same hidden state:

$$x_n = f_{\ell,j}(h_n), \qquad y_n = f_{\ell,i}(h_n), \tag{24}$$

where $x_n, y_n \in \mathbb{R}^{d_o}$ and d_o is the expert output dimension. The affine predictor applies a separate scale and bias to each output channel:

$$A_{\ell,i,j}(x) = \widehat{a}_{\ell,i,j} \odot x + \widehat{b}_{\ell,i,j}, \tag{25}$$

where $\widehat{a}_{\ell,i,j}, \widehat{b}_{\ell,i,j} \in \mathbb{R}^{d_o}$ and $\odot$ denotes element-wise multiplication. We estimate these parameters using weighted ridge regression:

$$(\widehat{a}_{\ell,i,j}, \widehat{b}_{\ell,i,j}) = \arg\min_{a,b} \sum_{n \in \mathcal{D}_{\text{fit}}} \omega_{n,i} \, \|y_n - a \odot x_n - b\|_2^2 + \eta \|a\|_2^2, \tag{26}$$

where $\eta > 0$ is the ridge coefficient and the intercept b is not penalized.

The channel-wise parameterization decomposes this objective into independent scalar regressions. Let

$$\bar{x} = \frac{\sum_n \omega_{n,i} x_n}{\sum_n \omega_{n,i}}, \qquad \bar{y} = \frac{\sum_n \omega_{n,i} y_n}{\sum_n \omega_{n,i}} \tag{27}$$

denote the weighted means on the fit split. For output channel d, the closed-form solution is:

$$\widehat{a}_{\ell,i,j,d} = \frac{\sum_n \omega_{n,i}(x_{n,d} - \bar{x}_d)(y_{n,d} - \bar{y}_d)}{\sum_n \omega_{n,i}(x_{n,d} - \bar{x}_d)^2 + \eta}, \qquad \widehat{b}_{\ell,i,j,d} = \bar{y}_d - \widehat{a}_{\ell,i,j,d} \bar{x}_d. \tag{28}$$

Directed proxy fidelity and assignment. After fitting, we evaluate each affine predictor on the disjoint scoring split using the weighted coefficient of determination in Equation 10. The target mean in its denominator is:

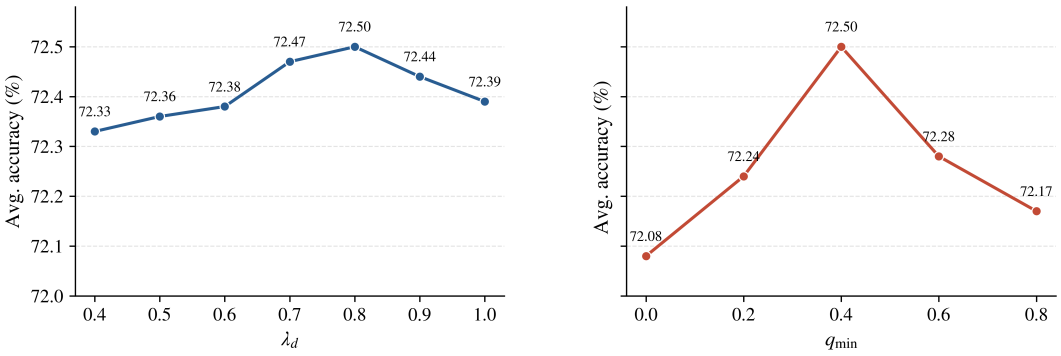
$$\mu_{\ell,i}^{\text{score}} = \frac{\sum_{(s,t) \in \mathcal{D}_{\text{score}}} \omega_{s,t,i} f_{\ell,i}(h_{s,t})}{\sum_{(s,t) \in \mathcal{D}_{\text{score}}} \omega_{s,t,i}}. \tag{29}$$

We define the proxy fidelity as $q_{\ell,i,j} = \max(0, R_{\ell,i,j}^2)$. Using a separate scoring split avoids evaluating the fitted predictor on the same tokens used to estimate its parameters. The score is directional: the predictor approximates target expert i from candidate proxy j, and the token weights are conditioned on i. Each pruned expert is assigned the retained candidate with the highest proxy fidelity according to Equation 11. The affine map associated with the selected pair is then reused for expert approximation without additional optimization.

C Analysis of Hyperparameters

In this section, we analyze the sensitivity of MoRA to the diversity regularization coefficient λ_{d} and the proxy-fidelity threshold $q_{\min}$ on Qwen3-30B-A3B. As shown in Figure 4, performance remains relatively stable across the tested values of λ_{d}, with average scores ranging from 72.33 to 72.50.

The highest score is obtained at $\lambda_{\mathrm{d}} = 0.8$, while further increasing the coefficient slightly reduces performance. This trend suggests that moderate diversity regularization helps balance predictive importance and routing diversity, whereas stronger regularization may place excessive emphasis on separating experts with similar routing preferences. The proxy-fidelity threshold has a more pronounced effect at low values: increasing $q_{\min}$ from 0 to 0.4 improves the average score from 72.08 to 72.50. A low threshold admits proxies that poorly approximate the pruned experts, potentially introducing output errors that offset the benefits of expert approximation. Raising the threshold filters out these unreliable substitutions. Further increasing the threshold reduces the number of eligible proxies, eventually reducing MoRA to pruning based solely on learned biases when no proxies meet the threshold.



(a) Average accuracy with varing regularizer coefficient

(b) Average accuracy with varing threshold

Figure 4: Sensitivity of MoRA to the diversity regularization coefficient λ_{d} (left) and the proxy-fidelity threshold $q_{\min}$ (right) on Qwen3-30B-A3B. We report the average score (%) across nine zero-shot benchmarks.

D Robustness to the Calibration Corpus

We examine the robustness of MoRA to the choice of calibration corpus on Qwen3-30B-A3B. Specifically, we learn the router biases using Zyda-2 (Tokpanov et al., 2024) for 1,000 iterations and compare the resulting pruning performance with the C4 results reported in the main text. We evaluate both settings at 25% and 50% expert pruning across the same nine zero-shot benchmarks.

Table 7: Calibration-corpus comparison for MoRA on Qwen3-30B-A3B. Avg. is the mean score (%) across nine zero-shot benchmarks. Bold indicates the higher score between the two calibration settings at each pruning ratio.

Pruning ratio	Calibration	ARC-c	ARC-e	BoolQ	HellaSwag	MMLU	OBQA	PIQA	RTE	WinoGrande	Avg.
25%	C4	**57.42**	**80.18**	**88.81**	**79.28**	**72.32**	44.80	**81.18**	76.17	**72.38**	**72.50**
	Zyda-2	56.06	79.25	87.89	78.66	71.91	**45.60**	80.25	**79.06**	71.82	72.28
50%	C4	46.25	69.07	82.51	75.20	53.14	41.80	**80.47**	75.09	68.98	65.83
	Zyda-2	**49.66**	**72.35**	**85.93**	**75.84**	**54.12**	**42.60**	79.11	**78.34**	**70.17**	**67.57**

As shown in Table 7, MoRA maintains comparable or higher average performance when the calibration corpus is changed from C4 to Zyda-2. At 25% pruning, the average score changes only slightly, from 72.50 to 72.28. At 50% pruning, Zyda-2 calibration achieves an average score of 67.57 compared with 65.83 for C4, with improvements on eight of the nine benchmarks. These results support the robustness of MoRA to the calibration corpus in the evaluated settings: effective expert selection is maintained with an alternative corpus. Since the two settings use different numbers of optimization steps, this comparison does not isolate the effect of the calibration corpus alone.

E Details of Token-Category Routing Analysis

Token annotation. We annotate the raw C4 text with the spaCy `en_core_web_trf` part-of-speech tagger before model tokenization. Special-token insertion is disabled. For fast tokenizers, each subword token inherits the category of the spaCy word whose character span overlaps the token's offset mapping. For tokenizers without character-level offsets, such as the Python tokenizer used by Moonlight, we use tokenizer-specific word-boundary markers to propagate the category of the corresponding whitespace-delimited word to its subwords. Tokens with zero-length offsets are discarded. We group the annotations into five categories: content words (NOUN, PROPN, VERB, ADJ, and ADV); function words (DET, PRON, ADP, AUX, CCONJ, SCONJ, PART, and INTJ); numbers; punctuation; and others.

Routing collection. We run the frozen pretrained model and record the top-k experts selected for each token in every routed MoE layer. Let $\mathcal{T}_\ell(h_t)$ denote the selected expert indices and $\widetilde{w}_{\ell,t,e}$ denote the normalized routing weight of expert e when it is selected. The learned router biases are not applied during this forward pass. The original pretrained router, including any native router correction terms, is kept unchanged. After collecting the routes, we associate each expert with its learned router bias $s_{\ell,e}$ and rank experts within each layer according to the relative magnitude of this scalar.

Routing-mass heatmap. For the heatmaps, we compute the conditional routing mass $p(expert \mid wordcategory)$ as

$$M_{\ell,c,e} = \frac{\sum_{t\in\mathcal{V}} \mathbb{I}[c_t = c]\mathbb{I}[e \in \mathcal{T}_\ell(h_t)]\widetilde{w}_{\ell,t,e}}{\sum_{t\in\mathcal{V}} \mathbb{I}[c_t = c]}, \tag{30}$$

where c_t is the category assigned to token t. Thus, each heatmap shows how the routing mass from a given token category is distributed across experts. Experts are ordered from left to right according to either their activation frequency or their learned router bias $s_{\ell,e}$.

F Additional Expert Importance and Routing Analyses

We extend the analyses in the main text to DeepSeek-V2-Lite and Moonlight-16B-A3B, following the same evaluation protocols.

F.1 Relationship Between Expert Scores and Deletion-Induced Loss

Figures 5 and 6 compare learned router biases and activation frequencies with the increase in LM loss caused by removing an individual expert. Within each layer, we consider the top 50% of experts ranked by each score separately and report the corresponding Spearman correlations.

The results on both models are consistent with the findings in the main text: learned biases exhibit a more consistent positive association with deletion-induced loss increases than activation frequency. This suggests that experts assigned higher biases tend to be more important for preserving predictive performance. Activation frequency measures how often an expert is selected, but does not directly capture the consequences of its removal.

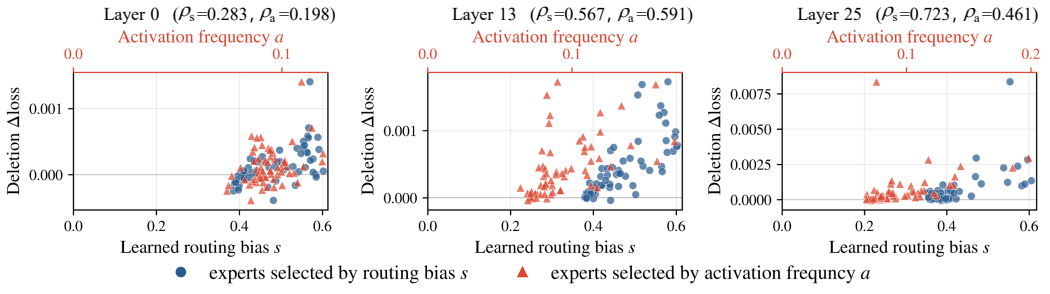


Figure 5: Relationship between expert scores and deletion-induced loss in DeepSeek-V2-Lite. Each point represents an expert in the indicated layer. The bias and activation-frequency analyses use their respective top-50% expert subsets.

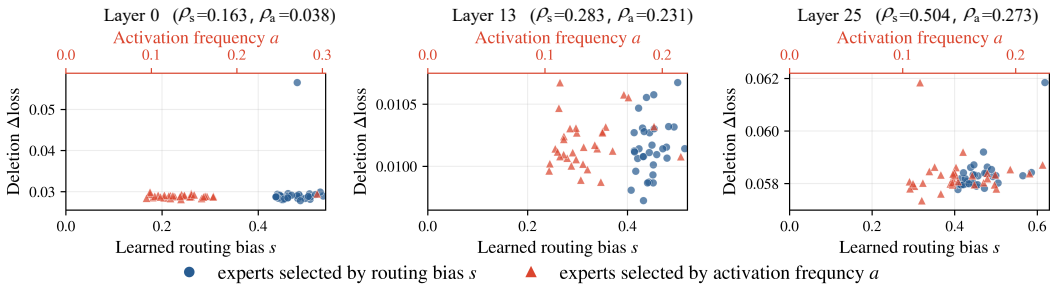
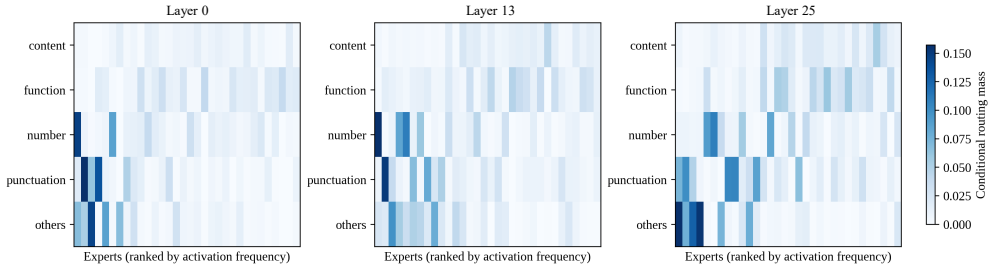


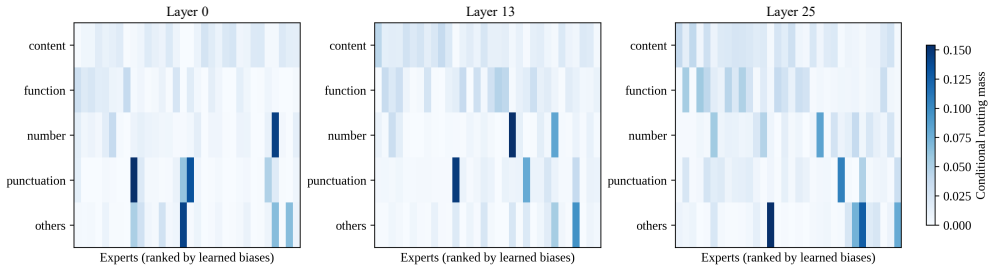
Figure 6: Relationship between expert scores and deletion-induced loss in Moonlight-16B-A3B. Each point represents an expert in the indicated layer. The bias and activation-frequency analyses use their respective top-50% expert subsets.

F.2 Word-to-Expert Routing Probabilities

Figures 7 and 8 visualize word-to-expert routing probabilities in DeepSeek-V2-Lite and Moonlight-16B-A3B, respectively. These heatmaps complement the analysis in the main text by illustrating how routing preferences vary across words and experts in the two additional architectures. In both models, routing mass for numbers and punctuation is often concentrated on a small subset of experts, whereas content words are generally distributed more broadly.

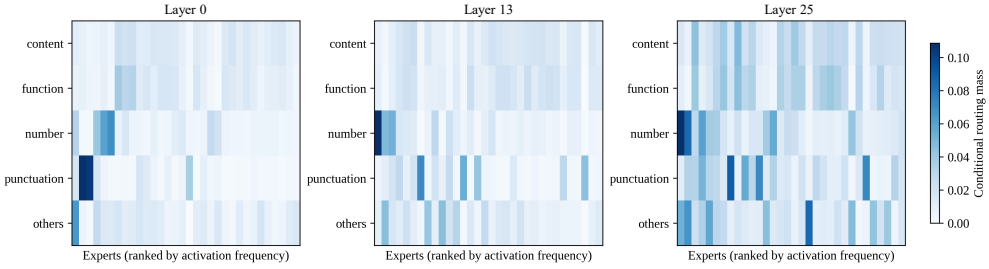


(a) Words routing probabilities with experts ranked by activation frequency in DeepSeek-V2-Lite.

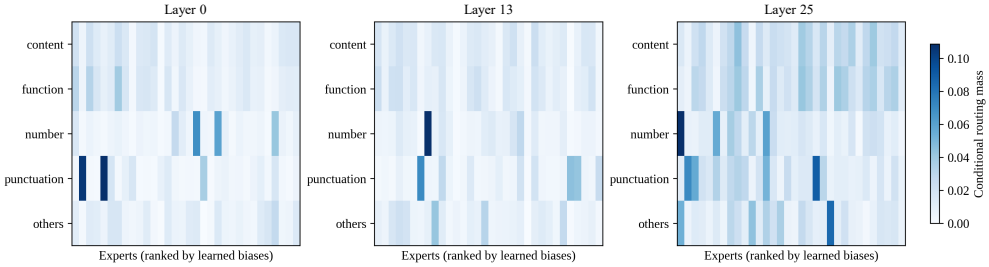


(b) Words routing probabilities with experts ranked by learned biases in DeepSeek-V2-Lite.

Figure 7: Heatmap visualization of word-to-expert routing probabilities in DeepSeek-V2-Lite. Color intensity indicates the routing probability assigned to each expert for a given word.



(a) Words routing probabilities with experts ranked by activation frequency in Moonlight.



(b) Words routing probabilities with experts ranked by learned biases in Moonlight.

Figure 8: Heatmap visualization of word-to-expert routing probabilities in Moonlight-16B-A3B. Color intensity indicates the routing probability assigned to each expert for a given word.