AIMER: Calibration-Free Task-Agnostic MoE Expert Pruning

Anonymous ACL submission

Abstract

Mixture-of-Experts (MoE) language models increase parameter capacity without proportional per-token computation, yet deployment still requires storing the full expert pool, making expert pruning important for reducing memory and serving overhead. Existing task-agnostic expert-pruning methods are typically calibration-dependent: they estimate expert importance from routing or activation statistics on a calibration set, making pruning decisions sensitive to calibration-data variation while introducing substantial preprocessing cost. We propose AIMER (**A**bsolute mean over root mean square **IM**portance for **E**xpert **R**anking), a simple calibration-free criterion that identifies more distinct experts by capturing the concentration pattern of expert weights, making it well suited for task-agnostic expert pruning. Across 7B to 47B MoE language models with distinct architectures and 16 diverse benchmarks, AIMER consistently delivers stronger capability balance across diverse tasks than existing calibration-free methods. Surprisingly, AIMER also achieves better balance than strong calibration-based expert-pruning baselines calibrated on the widely used task-agnostic C4 corpus, while requiring only 0.22–2.06 seconds to score all experts.

1 Introduction

Mixture-of-Experts (MoE) models extend Transformer architectures by replacing the dense feed-forward block with a set of expert FFNs and a router that activates only the top-k experts for each token (Vaswani et al., 2017; Shazeer et al., 2017). This conditional computation paradigm decouples parameter growth from per-token computation, making it possible to scale model capacity without incurring the full inference cost of dense models. As a result, recent MoE large language models (Jiang et al., 2024; Muennighoff et al., 2024; Liu et al., 2024a; Meta, 2025; Baidu, 2025; GLM-5-Team et al., 2026; Team et al., 2026; Qwen Team, 2026) achieve strong performance while maintaining relatively low per-token compute. Despite this advantage, efficient deployment of MoE models remains challenging because all experts must still be stored and managed at inference time, creating substantial memory and serving overhead. Recent routing analyses show that expert usage is often highly imbalanced and that many experts are functionally redundant (Huang et al., 2024), motivating a growing body of work on expert-level compression through pruning, merging, and related strategies (Li et al., 2023; Lu et al., 2024; Zhang et al., 2025; Lee et al., 2025; Lasby et al., 2025).

Existing MoE expert pruning methods are typically calibration-dependent, relying on routing or activation statistics collected from a calibration set to rank experts. Prior work has shown that pruning outcomes are highly sensitive to the choice of calibration corpus (Liu et al., 2024b; Zhang et al., 2026; Yang et al., 2025b; Liu et al., 2026). In task-agnostic pruning, C4 (Allen Institute for AI, 2024) is widely adopted as a general-domain calibration corpus (Frantar and Alistarh, 2023; Bandari et al., 2024; Xia et al., 2024; Ling et al., 2024). Yet using a standard general-domain corpus does not remove the sensitivity to calibration data. As shown in Figure 1, even with the corpus fixed to C4, varying only the number of calibration tokens already leads to substantially different pruning outcomes. This observation suggests that expert ranking remains inherently tied to the sampled distribution, introducing an irreducible bias toward experts that are useful for the sampled data rather than those that generalize across tasks. This raises a natural question for task-agnostic MoE pruning: *Can removable experts be identified without relying on calibration data?*

In this work, we propose AIMER, a calibration-free expert-pruning criterion that scores each expert by the concentration pattern of its weights, com-

puted as the mean absolute value normalized by the root-mean-square value. Empirically, through CKA (Kornblith et al., 2019) analysis we find that AIMER tends to retain more distinct experts, which helps explain why it is well suited for task-agnostic pruning: the retained expert pool should preserve diverse functionality rather than specialize to a particular capability. We evaluate AIMER on five representative MoE model families spanning distinct architectures and scales from 7B to 47B parameters. We compare against both calibration-free baselines and calibration-based expert-pruning baselines calibrated on C4 across 16 benchmarks covering coding, creative writing, mathematical reasoning, and multiple-choice question answering. Across these settings, AIMER achieves the best average rank on four of the five model families and ties for best on the fifth, indicating the strongest overall capability balance, while eliminating most of the calibration overhead: expert scoring takes only 0.22–2.06 seconds, compared with 0.75–2.96 hours for calibration-based REAP on C4.

Our main contributions are as follows:

- We highlight calibration dependence as a central limitation of task-agnostic MoE expert pruning, and provide evidence that pruning outcomes can remain sensitive even when calibration data is sampled from the same corpus.
- We propose AIMER, a simple calibration-free criterion for task-agnostic expert ranking in MoE language models that captures the concentration pattern of expert weights, and empirically show that it retains a more distinct set of experts after pruning.
- We show that across five MoE model families and 16 benchmarks, AIMER delivers stronger overall capability balance than strong expert-pruning baselines while reducing expert-scoring time from hours to seconds.

2 Related Work

2.1 Expert Pruning for MoE language models

Expert pruning in MoE models was first explored in task-adaptive settings. Chen et al. (2022) progressively drop non-professional experts for a downstream task, showing that substantial redundancy can be removed after fine-tuning. In multilingual machine translation, Koishekenov et al. (2023) prune language-specific experts to improve memory efficiency at deployment time. These studies establish that MoE models often contain significant redundancy, but their pruning strategies are limited to task-specific settings. More recent work studies task-agnostic compression for modern MoE language models. NAEE (Lu et al., 2024) selects retained experts by minimizing Frobenius-norm reconstruction error between the original and pruned layer outputs, and EEP (Liu et al., 2024b) uses a gradient-free evolutionary strategy to search for effective expert subsets. Zhang et al. (2025) prune groups of similar experts to preserve diversity among the retained experts, and STUN (Lee et al., 2025) clusters experts first to prune redundant ones and then applies unstructured pruning on remaining experts. Seer-MoE (Muzio et al., 2024) scores experts with heavy-hitters counting, either through hard activation counts or, in its soft-counting variant, by accumulating router softmax probabilities as weighted expert frequencies. REAP (Lasby et al., 2025) estimates expert importance from router-weighted activations in a one-

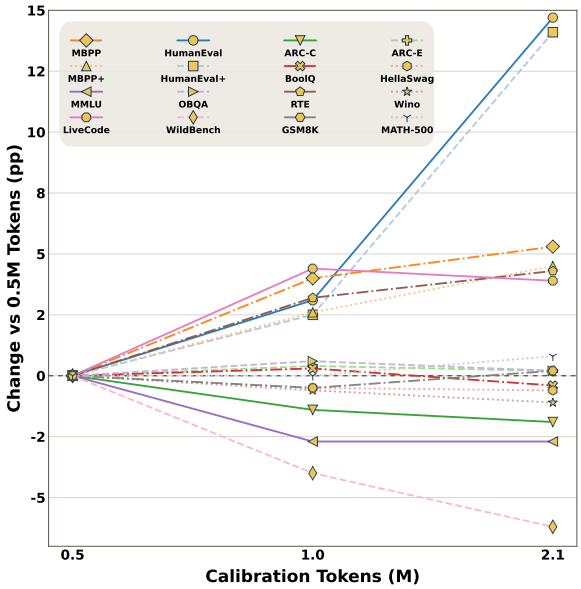

Figure 1: **Sensitivity of REAP (Lasby et al., 2025) to calibration set size on Qwen3-30B at 50% pruning ratio.** We fix the calibration corpus to C4 (Allen Institute for AI, 2024) and vary only the size of the calibration set from 0.5M to 2.1M tokens. The horizontal axis reports calibration tokens in millions (M = million tokens), and the vertical axis reports performance change relative to the 0.5M-token setting (pp = percentage points). Half of the benchmarks show significant variation. Some benchmarks improve while others degrade, showing that performance is highly sensitive to calibration set size even with the same corpus.

shot pruning pipeline. Jaiswal et al. (2025) benchmark 16 expert-importance criteria for expert dropping and report expert activation norm (EAN) as the strongest criterion among them.

2.2 Expert Merging for MoE language models

MEO (He et al., 2023) performs merging online at inference time, constructing a merged expert for each token as a router-score-weighted combination of the activated experts. Subsequent methods instead adopt offline merging. MC-SMoE (Li et al., 2023) first aligns neurons across experts and then merges routing-based groups using activation-frequency-weighted averaging. HC-SMoE (Chen et al., 2024) builds a hierarchical clustering of experts based on output similarity and merges experts within each cluster by frequency-weighted averaging. DERN (Zhou et al., 2025) goes beyond whole-expert merging: after pruning redundant experts, it decomposes them into neuron-level segments and reallocates those segments to compatible retained experts. Despite differences in scoring criteria and compression pipelines, most expert pruning and merging methods still rely on routing statistics, activation measurements, or calibration-set evaluations to decide which experts to remove or merge.

2.3 Calibration-Free Model Pruning

Calibration-free pruning studies whether removable structure can be identified from the pretrained model itself, without collecting activations on a held-out calibration set. Early work in dense networks approached this question through explicit weight-space redundancy. Srinivas and Babu (2015) show that pruning can be performed one neuron at a time by identifying similar neurons and removing those whose contribution can be absorbed by the remaining weights. Mussay et al. (2019) develop this perspective further by casting pruning as a coreset construction problem, selecting a small weighted subset of neurons that approximates the original layer with provable guarantees for arbitrary future inputs. Subsequent work shifts from neuron selection toward reconstruction-based compensation. RED++ (Yvinec et al., 2022) exploits redundancies in neuron weights through data-free hashing and removes input-wise redundant operations via input splitting and output merging, showing that structured pruning can be carried out without access to data. More recently, Sengupta et al. (2025) bring the calibration-free view to dense LLM compression with PruneNet, which reformulates pruning as policy learning over intrinsic model properties instead of relying on calibration examples. Taken together, these studies suggest that informative compression signals can often be derived directly from pretrained weights, without relying on external calibration sets. Yet calibration-free expert-level pruning for MoE models remains largely unexplored.

3 Preliminary

Mixture-of-Experts. An MoE layer replaces the dense feed-forward network in a standard Transformer block with a collection of n expert FFNs $\{E_i\}_{i=1}^{n}$ and a router that activates only a small subset of them for each token (Shazeer et al., 2017). This design preserves large model capacity while avoiding the cost of evaluating every expert on every input. Given a token representation $h \in \mathbb{R}^d$, the router first computes an expert logit vector

$$z = \mathbf{W}_r h \in \mathbb{R}^n,$$

where $\mathbf{W}_r \in \mathbb{R}^{n \times d}$ is the router projection matrix. Let $\mathcal{E}(h) = \mathrm{TopK}(z, k)$ denote the indices of the top-k logits, with $k \ll n$. The router then normalizes scores only over these selected experts, yielding sparse gating weights

$$g_i(h) = \begin{cases} \dfrac{\exp(z_i)}{\sum_{j \in \mathcal{E}(h)} \exp(z_j)}, & i \in \mathcal{E}(h), \\ 0, & i \notin \mathcal{E}(h), \end{cases} \tag{1}$$

The layer output is the weighted combination of the activated experts, where $A_i(h) = E_i(h)$ denotes the output activation produced by expert E_i:

$$y(h) = \sum_{i \in \mathcal{E}(h)} g_i(h)\, A_i(h). \tag{2}$$

Since $g(h)$ has only k nonzero entries, each token is routed to just a few experts. As a result, the computation per-token scales with k rather than n, while the total parameter capacity still grows with the full expert pool (Fedus et al., 2022).

4 Methodology

4.1 Problem Formulation

The goal of task-agnostic expert pruning is to reduce the expert pool while preserving the model's overall capability across diverse tasks, rather than leaning to any specific one. Consider an MoE language model with L layers, where layer ℓ contains

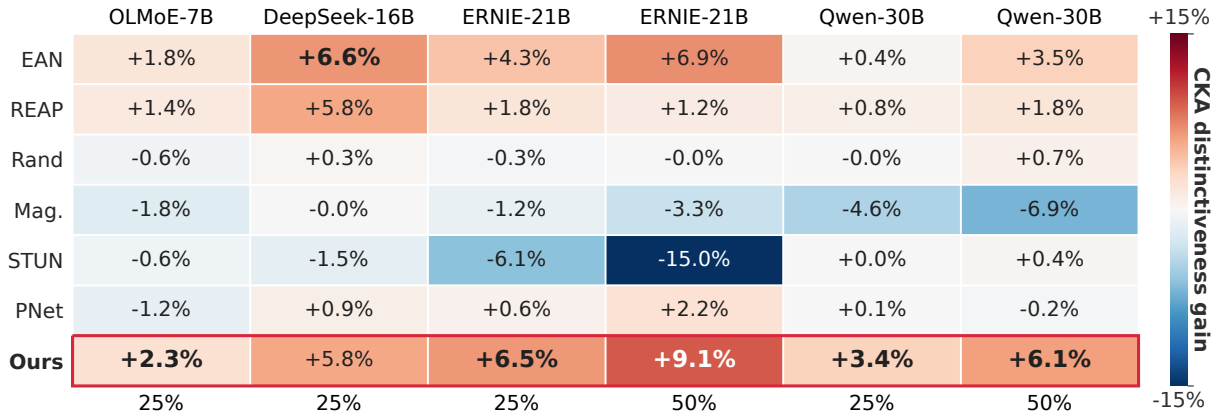

Figure 2: **Change in expert distinctiveness after pruning.** For each model and pruning ratio, we use 4096 C4 tokens to compute pairwise CKA similarities among expert outputs within each layer. Each cell reports the layer-averaged relative decrease in mean CKA after pruning, $(\mathrm{CKA}_{\mathrm{all}} - \mathrm{CKA}_{\mathrm{kept}})/\mathrm{CKA}_{\mathrm{all}}$. Positive values indicate that the retained experts are less similar to one another than the original expert pool, i.e., more functionally distinct; negative values indicate increased redundancy among retained experts. AIMER (Ours) yields positive distinctiveness gains in all shown settings and is strongest or near-strongest in most of them.

n_ℓ experts $\{E_i^{(\ell)}\}_{i=1}^{n_\ell}$. Given a target pruning ratio ρ, every layer removes the same fraction of experts. In this setup, the pruning problem reduces to a problem of expert ranking within the layer. For each layer ℓ, we assign each expert $E_i^{(\ell)}$ a scalar score $s_i^{(\ell)}$, sort experts within that layer by this score, and prune the ρn_ℓ experts judged as more redundant. After the pruning decision is made, we remove the selected expert FFN parameters together with the corresponding rows in the router matrix, and retain the same top-k routing rule over the remaining experts. The remaining question is therefore how to define the within-layer score $s_i^{(\ell)}$. We answer this with AIMER, a criterion that assigns each expert a score computed directly from its parameters.

4.2 Proposed Method

The AIMER criterion. For one expert, let d denote the input dimension and m denote the hidden dimension. Then $\mathbf{W}_{\mathrm{gate}}, \mathbf{W}_{\mathrm{up}} \in \mathbb{R}^{m\times d}$ and $\mathbf{W}_{\mathrm{down}} \in \mathbb{R}^{d\times m}$ are the gate, up, and down projection matrices of that expert. We define

$$\begin{aligned} N &= N_{\mathrm{gate}} + N_{\mathrm{up}} + N_{\mathrm{down}}, \\ P &= \|\mathbf{W}_{\mathrm{gate}}\|_1 + \|\mathbf{W}_{\mathrm{up}}\|_1 + \|\mathbf{W}_{\mathrm{down}}\|_1, \\ Q &= \|\mathbf{W}_{\mathrm{gate}}\|_F^2 + \|\mathbf{W}_{\mathrm{up}}\|_F^2 + \|\mathbf{W}_{\mathrm{down}}\|_F^2, \end{aligned} \tag{3}$$

where $N_{\mathrm{gate}} = N_{\mathrm{up}} = N_{\mathrm{down}} = md$ are the numbers of parameters in the three matrices. AIMER is then

$$\mathrm{AIMER} = \frac{P/N}{\sqrt{Q/N}} = \frac{P}{\sqrt{NQ}}. \tag{4}$$

and we prune experts with larger AIMER scores. The layer-wise procedure is shown in Algorithm 1.

If we flatten and concatenate the three projection matrices of an expert into a single vector $\mathbf{w} \in \mathbb{R}^N$, then

$$\mathrm{AIMER}(\mathbf{w}) = \frac{\|\mathbf{w}\|_1}{\sqrt{N}\,\|\mathbf{w}\|_2}. \tag{5}$$

This vector form is algebraically equivalent to Equation (4) and is in the same spirit as the Hoyer metric (Hoyer, 2004):

$$\mathrm{Hoyer}(\mathbf{w}) = \frac{\sqrt{N} - \|\mathbf{w}\|_1/\|\mathbf{w}\|_2}{\sqrt{N}-1}. \tag{6}$$

That is, both AIMER and the Hoyer metric are functions of the same underlying ℓ_1/ℓ_2 ratio. Hoyer and DeepHoyer (Yang et al., 2020) use the ℓ_1/ℓ_2 norm as a training-time regularizer, typically on vectors or channels within a weight matrix. In contrast, AIMER uses the same underlying quantity only after training, as a calibration-free weight-only criterion that treats the entire expert as a single vector and assigns it a scalar ranking score.

Basic properties. Let $\mathbf{w} \in \mathbb{R}^N$ denote the flattened parameter vector of one expert. AIMER has two useful properties for expert comparison. First, it is *scale-invariant*: for any nonzero scalar c, $\mathrm{AIMER}(c\mathbf{w}) = \mathrm{AIMER}(\mathbf{w})$. This is desirable for ranking because the goal is to compare experts by the relative pattern of their parameters, rather than by their overall scale. Second, it is *bounded*: combining $\|\mathbf{w}\|_1 \geq \|\mathbf{w}\|_2$ with Cauchy–Schwarz

gives

$$\frac{1}{\sqrt{N}} \leq \text{AIMER}(\mathbf{w}) \leq 1. \tag{7}$$

The upper bound is attained when all entries have equal absolute value, whereas the lower bound is attained when only one element is nonzero. Since N is fixed within a layer, all experts in that layer share the same bounded score range, making AIMER directly comparable across experts. The factor $1/\sqrt{N}$ therefore does not change the within-layer ranking and is kept only for interpretability.

Explanation. AIMER measures the concentration pattern of an expert's weights through a normalized ℓ_1/ℓ_2 ratio. A larger AIMER score means that the weight mass is more evenly distributed across parameters, whereas a smaller score means that the mass is concentrated in fewer large parameters. Empirically, Figure 2 shows that AIMER yields positive CKA distinctiveness gains in all settings and is the strongest method in all settings except DeepSeek-16B, where it ranks second. This suggests that AIMER tends to remove redundant experts while preserving diverse expert transformations, which is desirable for task-agnostic pruning where no target capability is preferred.

Algorithm 1 PyTorch-style expert ranking with AIMER

```
def aimer_rank(layer):
    scores = []
    num_experts = layer.gate.weight.
    shape[0]

    for e in range(num_experts):
        gate, up, down =
    get_proj_weights(layer, e)
        abs_sum = gate.abs().sum() + up.
    abs().sum() + down.abs().sum()
        numel = gate.numel() + up.numel
    () + down.numel()
        l2_sq = gate.square().sum() + up.
    square().sum() + down.square().sum()
        score = (abs_sum / numel) /
    torch.sqrt(l2_sq / numel)
        scores.append(score)

    scores = torch.stack(scores)
    _, sorted_idx = torch.sort(scores,
    descending=True)
    return sorted_idx
```

5 Experimental Results

5.1 Experimental Setup

Models and baselines. We evaluate AIMER on five representative MoE LLMs spanning different architectures and scales: OLMoE-1B-7B-0125-Instruct (Muennighoff et al., 2024), DeepSeek-V2-Lite-Chat (DeepSeek-AI, 2024), ERNIE-4.5-21B-A3B-PT (Baidu, 2025), Qwen3-30B-A3B-Instruct-2507 (Yang et al., 2025a), and Mixtral-8x7B-Instruct-v0.1 (Jiang et al., 2024). Their architectural details are summarized in Appendix A. We report the main results at 25% pruning for OLMoE and DeepSeek, and at 50% pruning for ERNIE, Qwen3, and Mixtral; additional pruning settings are provided in the appendix. We compare AIMER against four calibration-based expert-pruning baselines: REAP (Lasby et al., 2025), Expert Activation Norm (EAN) (Jaiswal et al., 2025), Frequency, and SEER soft counting (Muzio et al., 2024). Table 1 summarizes the extra signals required by these calibration-based criteria, and Appendix B gives the corresponding score definitions. For a controlled comparison, all calibration-based baselines use the same 4.2M-token calibration set sampled from C4 (Allen Institute for AI, 2024). We also compare with four calibration-free baselines: Random, Magnitude, STUN (Lee et al., 2025), and PruneNet (Sengupta et al., 2025).

Table 1: **Comparison of statistics requirements for AIMER and calibration-based baselines.** Data = calibration set, Act. = expert activations, and Route = router weights. AIMER ranks experts from pretrained weights alone, so none of these extra signals are required. Red marks required; green marks not required.

Method	Data	Act.	Route
Frequency	✓	×	✓
SEER	✓	×	✓
EAN	✓	✓	×
REAP	✓	✓	✓
AIMER (Ours)	×	×	×

Evaluation suite. To assess whether pruning decisions generalize beyond a narrow task distribution, we evaluate the pruned models in the zero-shot setting on 16 benchmarks covering both discriminative reasoning and open-ended generation. For multiple-choice evaluation, we report AI2 Reasoning Challenge (ARC-C/ARC-E) (Clark et al., 2018), BoolQ (Clark et al., 2019), HellaSwag (Zellers et al., 2019), MMLU (Hendrycks et al., 2020), OpenBookQA (OBQA) (Mihaylov et al., 2018), Recognizing Textual Entailment (RTE) (Bentivogli et al., 2009), and WinoGrande (WinoG.) (Sakaguchi et al., 2021), all implemented with lm-eval-harness (Gao et al., 2021). For

Table 2: **Expert-pruning results on OLMoE-1B-7B-0125-Instruct (OLMoE) and ERNIE-4.5-21B-A3B-PT (ERNIE).** AIMER is compared with calibration-based baselines calibrated on C4 and with calibration-free baselines. It achieves the best rank average in both settings, indicating the strongest overall capability balance.

			Coding			Creative Writing	Math			MC	Rank
Model	**Ratio**	**Method**	**Eval+**	**LiveCode**	**Code Avg**	**WildBench**	**GSM8K**	**MATH-500**	**Math Avg**	**MC Avg**	**Rank Avg**
OLMoE	0%	Full	0.341	0.033	0.187	0.444	0.682	0.222	0.452	0.653	N/A
	25%	Frequency	0.000	0.000	0.000	0.127	0.033	0.024	0.028	0.560	6.67
		SEER	0.000	0.000	0.000	0.141	0.037	0.012	0.025	0.564	6.00
		EAN	0.000	0.000	0.000	0.205	0.115	0.006	0.061	0.577	5.50
		REAP	0.014	0.000	0.007	0.260	0.202	**0.056**	0.129	0.588	2.92
		Random	**0.112**	0.000	**0.056**	**0.297**	0.140	0.034	0.087	0.562	3.50
		Magnitude	0.000	0.000	0.000	0.020	0.072	0.004	0.038	0.478	7.58
		STUN	0.000	0.000	0.000	0.089	0.072	0.008	0.040	0.497	6.92
		PruneNet	0.102	0.000	0.051	0.273	0.162	0.044	0.103	0.563	3.33
		AIMER (Ours)	0.026	0.000	0.013	0.236	**0.221**	**0.056**	**0.139**	**0.600**	**2.58**
ERNIE	0%	Full	0.867	0.247	0.557	0.479	0.829	0.780	0.804	0.721	N/A
	50%	Frequency	0.007	0.000	0.004	0.164	0.051	0.018	0.034	0.557	7.58
		SEER	0.003	0.000	0.002	0.174	0.106	0.020	0.063	0.543	7.08
		EAN	0.009	0.000	0.004	**0.253**	0.065	0.026	0.046	**0.605**	5.25
		REAP	0.024	0.000	0.012	0.199	0.567	0.178	0.373	0.592	4.08
		Random	0.220	0.011	0.116	0.158	0.420	0.198	0.309	0.584	4.17
		Magnitude	0.243	0.049	0.146	0.077	0.351	0.130	0.240	0.503	5.00
		STUN	0.117	0.005	0.061	0.044	0.103	0.066	0.085	0.487	6.83
		PruneNet	0.199	0.022	0.111	0.180	0.461	0.154	0.307	0.586	3.67
		AIMER (Ours)	**0.254**	**0.071**	**0.163**	0.217	**0.650**	**0.348**	**0.499**	0.598	**1.33**

open-ended generation, we evaluate code generation on EvalPlus (Liu et al., 2023) and 182 LiveCodeBench (Jain et al., 2024) problems collected between January and April 2025; mathematical reasoning on GSM8K (Cobbe et al., 2021) and MATH-500 (Hendrycks et al., 2021) using EvalScope (Team, 2024); and creative writing on 146 prompts sampled from WildBench (Lin et al., 2024). For WildBench, we use `gpt-oss-120b` (OpenAI, 2025) as the judge. Overall, this suite spans multiple-choice QA, coding, math, and creative generation, allowing us to assess pruning robustness across substantially different capabilities.

Protocol and hardware. All evaluations are conducted in the zero-shot setting to isolate the effect of expert pruning from task-specific adaptation. For open-ended generation, we use deterministic decoding with `do_sample=False`; when an explicit temperature parameter is exposed, we set `temperature=0`. In practice, this corresponds to greedy, non-sampled generation, improving reproducibility across pruning methods. Experiments are conducted on NVIDIA L40S 48GB GPUs.

5.2 Main Results

The main zero-shot results are reported in Tables 2 and 3, and the full per-benchmark results are provided in Appendix C. Across five model families and 16 benchmarks, AIMER delivers the strongest or tied-strongest overall capability balance among calibration-free methods. It achieves the best rank average on OLMoE-7B, DeepSeek-16B, ERNIE-21B, and Qwen3-30B, and ties STUN for the best rank average on Mixtral-8x7B. More surprisingly, AIMER also provides better overall balance than strong calibration-based expert-pruning baselines calibrated on C4, despite requiring no calibration set, activations, or router statistics. The category-level results show the same pattern: AIMER is strongest on math and multiple-choice evaluation for OLMoE, strongest on math and second-best on coding for DeepSeek, strongest on coding and math for ERNIE, and strongest on coding for both Qwen3 and Mixtral. On Qwen3 at 50% pruning, for example, AIMER reaches a 36.1% code average, compared with 4.6% for REAP and 0% for Frequency, SEER, and EAN. Although other methods can still win individual capability groups, such as REAP on Qwen3 math and STUN on part of Mixtral, AIMER provides the most balanced overall profile. This is exactly the behavior we want in task-agnostic pruning: the retained experts should support broad general ability rather than reflect the preferences of a particular calibration sample. Appendix D further studies how REAP changes under different calibration choices, including EvolCodeAlpaca-v1 (Luo et al., 2023) and mixed data.

5.3 Further Discussions

The trade-off in MoE expert pruning. Expert pruning is fundamentally a trade-off: because experts are removed entirely, being strong on one capability can easily come with severe degradation on another, especially at higher pruning ratios. The results in Tables 2 and 3 show this clearly. EAN

Table 3: **Expert-pruning results on DeepSeek-V2-Lite-Chat (DeepSeek), Qwen3-30B-A3B-Instruct-2507 (Qwen3), and Mixtral-8x7B-Instruct-v0.1 (Mixtral).** AIMER achieves the best rank average on DeepSeek and Qwen3 and ties for the best rank average on Mixtral, indicating a consistently strong overall capability balance across coding, creative writing, math, and multiple-choice evaluation.

			Coding			Creative Writing	Math			MC	Rank
Model	**Ratio**	**Method**	**Eval+**	**LiveCode**	**Code Avg**	**WildBench**	**GSM8K**	**MATH-500**	**Math Avg**	**MC Avg**	**Rank Avg**
DeepSeek	0%	Full	0.549	0.104	0.327	0.418	0.610	0.298	0.454	0.678	N/A
	25%	Frequency	0.000	0.000	0.000	0.291	0.023	0.012	0.017	0.602	6.17
		SEER	0.000	0.000	0.000	0.155	0.034	0.016	0.025	0.602	6.33
		EAN	0.000	0.000	0.000	**0.307**	0.290	0.018	0.154	0.625	4.00
		REAP	0.001	0.000	0.001	0.152	0.242	0.046	0.144	**0.640**	3.92
		Random	0.000	0.000	0.000	0.050	0.081	0.018	0.050	0.521	6.67
		Magnitude	0.000	0.000	0.000	0.003	0.008	0.002	0.005	0.419	8.25
		STUN	0.065	**0.011**	0.038	0.137	0.077	0.030	0.053	0.554	4.75
		PruneNet	**0.185**	0.005	**0.095**	0.221	0.281	0.038	0.160	0.604	3.00
		AIMER (Ours)	0.163	**0.011**	0.087	0.246	**0.493**	**0.134**	**0.313**	0.621	**1.92**
Qwen3	0%	Full	0.871	0.368	0.619	0.644	0.923	0.802	0.862	0.737	N/A
	50%	Frequency	0.000	0.000	0.000	0.015	0.000	0.000	0.000	0.437	7.00
		SEER	0.000	0.000	0.000	0.018	0.000	0.000	0.000	0.436	7.00
		EAN	0.000	0.000	0.000	**0.522**	0.650	0.032	0.341	**0.697**	4.00
		REAP	0.054	0.038	0.046	0.226	**0.895**	**0.804**	**0.849**	0.628	2.33
		Random	0.010	0.005	0.007	0.027	0.085	0.038	0.061	0.409	4.83
		Magnitude	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.372	7.75
		STUN	0.000	0.000	0.000	0.000	0.002	0.006	0.004	0.342	7.25
		PruneNet	0.151	0.055	0.103	0.151	0.387	0.250	0.319	0.595	3.17
		AIMER (Ours)	**0.601**	**0.121**	**0.361**	0.413	0.664	0.494	0.579	0.638	**1.67**
Mixtral	0%	Full	0.512	0.148	0.330	0.427	0.661	0.294	0.478	0.737	N/A
	50%	Frequency	0.033	0.000	0.017	0.139	0.054	0.020	0.037	0.520	8.25
		SEER	0.035	0.000	0.018	0.130	0.070	0.010	0.040	0.523	8.08
		EAN	0.152	0.000	0.076	0.188	0.205	0.038	0.121	**0.674**	4.67
		REAP	0.166	0.011	0.089	0.244	**0.347**	0.054	**0.201**	0.670	2.67
		Random	0.102	0.016	0.059	0.218	0.177	0.038	0.107	0.572	4.58
		Magnitude	0.076	0.000	0.038	0.217	0.138	0.022	0.080	0.491	7.00
		STUN	**0.214**	0.027	0.120	0.260	0.224	**0.058**	0.141	0.649	**2.00**
		PruneNet	0.121	0.005	0.063	0.189	0.157	0.022	0.089	0.538	5.75
		AIMER (Ours)	0.198	**0.044**	**0.121**	**0.263**	0.209	0.040	0.125	0.672	**2.00**

Table 4: **Efficiency comparison between AIMER (Ours) and REAP.** All calibration measurements are collected on two NVIDIA L40S 48GB GPUs, except Mixtral, which uses four GPUs. Calibration time and peak calibration memory measure the expert-scoring stage before structural pruning. Loading memory is reported as after / before pruning at 50% pruning ratio. Overall, AIMER incurs substantially lower resource consumption than calibration-based REAP during expert scoring.

Model	**Calibration Time**		**Peak Calibration Memory**		**Loading Memory (GB) After / Before**
	REAP	**AIMER (Ours)**	**REAP**	**AIMER (Ours)**	
OLMoE-7B	0.75 h	**0.22 s**	15.51 GB	**13.00 GB**	6.89 / 12.89
DeepSeek-16B	1.04 h	**0.51 s**	34.44 GB	**29.42 GB**	15.86 / 29.27
ERNIE-21B	1.37 h	**0.51 s**	44.72 GB	**40.85 GB**	21.67 / 40.66
Qwen3-30B	2.96 h	**1.27 s**	63.07 GB	**57.01 GB**	29.93 / 56.92
Mixtral-47B	1.61 h	**2.06 s**	93.18 GB	**92.27 GB**	44.99 / 86.99

can preserve MCQ or WildBench relatively well in some settings, yet it often performs poorly on coding and math, while REAP can remain strong on math but much weaker on coding under aggressive pruning. For task-agnostic pruning, the goal is therefore not to maximize one benchmark family, but to preserve general ability. This means pruning those that are more replaceable by the remaining expert set, while retaining those that contribute more distinctive transformations. The strong balance of AIMER across coding, creative writing, math, and MCQ suggests that it is effective at identifying such replaceable experts in task-agnostic pruning.

Magnitude versus AIMER. The comparison with Magnitude is useful because it separates two questions: whether pretrained weights contain pruning signal at all, and whether raw weight scale is the right way to use that signal. Magnitude answers the first question positively under moderate pruning. On Qwen3-30B at 25% pruning, for example, it is competitive on coding and math metrics (Tables 6 and 7), showing that absolute weight scale can carry meaningful expert-level information. However, these gains are uneven and do not transfer reliably once the pruning decision becomes harder. At 50% pruning on Qwen3-30B, Magnitude collapses across open-ended and reasoning benchmarks, whereas AIMER preserves substantially stronger coding, math, and multiple-choice performance (Tables 3, 6 and 7). This suggests that

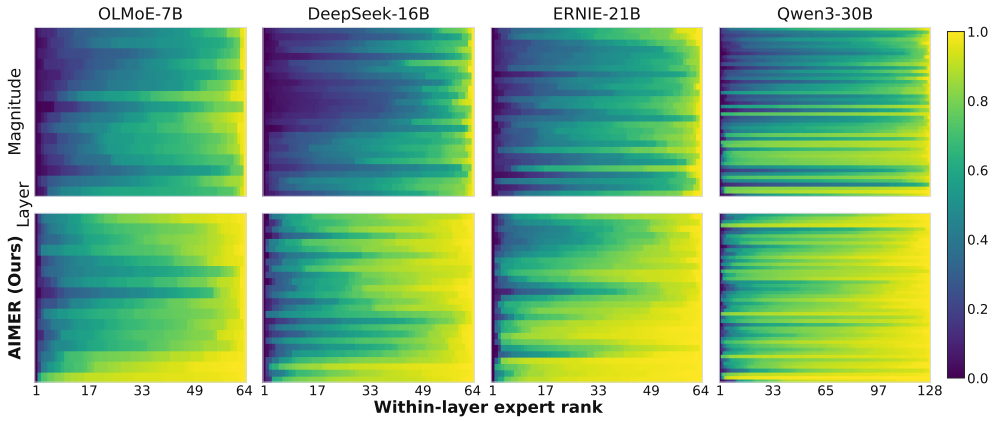

Figure 3: **Layer-wise Magnitude and AIMER score profiles across three MoE models.** Columns show OLMoE, DeepSeek,ERNIE, and Qwen3. The top row uses Magnitude, and the bottom row uses AIMER. Within each layer, experts are ranked by the corresponding score, and scores are min-max rescaled to $[0, 1]$. The x-axis reports within-layer expert rank, the y-axis reports layer index, and color indicates the rescaled score. Compared with Magnitude, AIMER yields a more separable distribution over experts, making the differences more distinguishable.

raw magnitude is a useful but incomplete proxy: it captures overall expert scale, but it does not distinguish experts whose weights have similar scale but different concentration patterns. This difference is consistent with the layer-wise score profiles in Figure 3. When experts are ranked by raw magnitude, many layers still contain a broad middle region of near-tied scores, leaving the pruning boundary only weakly defined. That ambiguity may be tolerable when pruning is mild, but it becomes much more costly once a larger fraction of experts is removed. AIMER, by contrast, produces a stratified within-layer score profile, shrinking this ambiguous middle region and yielding a decisive ranking.

Resource consumption. Because AIMER ranks experts directly from pretrained weights, it removes the expensive activation accumulation required by calibration-based methods. As shown in Table 4, expert scoring takes only 0.22–2.06 seconds for AIMER, whereas REAP requires 0.75–2.96 hours. AIMER also reduces peak calibration memory across all five models. These savings in calibration time and memory become more pronounced as model scale increases. At a fixed pruning ratio, expert pruning reduces loading memory relative to the unpruned model, and the post-pruning loading memory is the same across methods.

6 Conclusion

In this work, we propose AIMER, a simple yet effective calibration-free criterion for task-agnostic expert pruning in MoE models. AIMER scores experts by the concentration pattern of their pretrained weights and tends to retain more distinct experts after pruning. Across five MoE model families from 7B to 47B and 16 diverse benchmarks, AIMER remains competitive with, and often outperforms, strong expert-pruning baselines while reducing expert-scoring time from hours to seconds. These results show that pretrained weights alone can provide an effective signal for task-agnostic expert ranking, making calibration-free MoE pruning a practical and broadly applicable alternative to calibration-dependent methods.

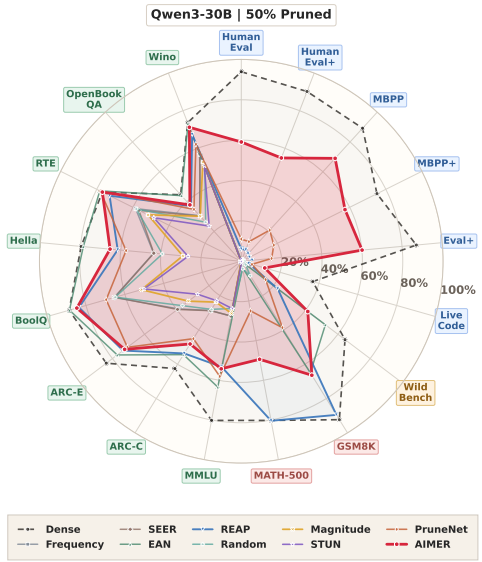

Figure 4: **Radar plot of Qwen3-30B performance across all benchmarks at 50% pruning ratio.** The dashed outline denotes the dense model, and each colored trace corresponds to one pruning method. Higher values indicate better task performance on the corresponding benchmark. Among the pruned models, AIMER encloses the largest area and has a capability profile closer to that of the full model. Additional radar plots for the other settings are provided in Appendix E.

Limitations

This work has one main limitation: AIMER is designed for *task-agnostic* expert pruning rather than task-specific compression. Since it ranks experts only from pretrained weights, it aims to identify experts that are broadly distinctive or replaceable across tasks, not experts that are most important for a particular downstream distribution. When pruning for a specific task or capability, calibration data or task-adaptive signals may still be necessary.

References

Allen Institute for AI. 2024. allenai/c4 · datasets at Hugging Face. https://huggingface.co/datasets/allenai/c4.

Baidu. 2025. Ernie 4.5 technical report. Technical report, Baidu. Technical report.

Abhinav Bandari, Lu Yin, Cheng-Yu Hsieh, Ajay Kumar Jaiswal, Tianlong Chen, Li Shen, Ranjay Krishna, and Shiwei Liu. 2024. Is c4 dataset optimal for pruning? an investigation of calibration data for llm pruning. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 18089–18099.

Luisa Bentivogli, Peter Clark, Ido Dagan, and Danilo Giampiccolo. 2009. The fifth pascal recognizing textual entailment challenge. *TAC*, 7(8):1.

I Chen, Hsu-Shen Liu, Wei-Fang Sun, Chen-Hao Chao, Yen-Chang Hsu, Chun-Yi Lee, and 1 others. 2024. Retraining-free merging of sparse moe via hierarchical clustering. *arXiv preprint arXiv:2410.08589*.

Tianyu Chen, Shaohan Huang, Yuan Xie, Binxing Jiao, Daxin Jiang, Haoyi Zhou, Jianxin Li, and Furu Wei. 2022. Task-specific expert pruning for sparse mixture-of-experts. *arXiv preprint arXiv:2206.00277*.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. 2019. Boolq: Exploring the surprising difficulty of natural yes/no questions. *arXiv preprint arXiv:1905.10044*.

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

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, and 1 others. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

DeepSeek-AI. 2024. Deepseek-v2: A strong, economical, and efficient mixture-of-experts language model. *Preprint*, arXiv:2405.04434.

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

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

Leo Gao, Jonathan Tow, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Kyle McDonell, Niklas Muennighoff, and 1 others. 2021. A framework for few-shot language model evaluation. *Zenodo*.

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, and 168 others. 2026. Glm-5: from vibe coding to agentic engineering. *Preprint*, arXiv:2602.15763.

Shwai He, Run-Ze Fan, Liang Ding, Li Shen, Tianyi Zhou, and Dacheng Tao. 2023. Merging experts into one: Improving computational efficiency of mixture of experts. *arXiv preprint arXiv:2310.09832*.

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

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874*.

Patrik O Hoyer. 2004. Non-negative matrix factorization with sparseness constraints. *Journal of machine learning research*, 5(Nov):1457–1469.

Wei Huang, Yue Liao, Jianhui Liu, Ruifei He, Haoru Tan, Shiming Zhang, Hongsheng Li, Si Liu, and Xiaojuan Qi. 2024. Mixture compressor for mixture-of-experts llms gains more. *arXiv preprint arXiv:2410.06270*.

Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. 2024. Livecodebench: Holistic and contamination free evaluation of large language models for code. *arXiv preprint arXiv:2403.07974*.

Ajay Jaiswal, Jianyu Wang, Yixiao Li, Pingzhi Li, Tianlong Chen, Zhangyang Wang, Chong Wang, Ruoming Pang, and Xianzhi Du. 2025. Finding fantastic

experts in moes: A unified study for expert dropping strategies and observations. *arXiv preprint arXiv:2504.05586*.

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, and 1 others. 2024. Mixtral of experts. *arXiv preprint arXiv:2401.04088*.

Yeskendir Koishekenov, Alexandre Berard, and Vassilina Nikoulina. 2023. 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)*, pages 3567–3585.

Simon Kornblith, Mohammad Norouzi, Honglak Lee, and Geoffrey Hinton. 2019. Similarity of neural network representations revisited. *Preprint*, arXiv:1905.00414.

Mike Lasby, Ivan Lazarevich, Nish Sinnadurai, Sean Lie, Yani Ioannou, and Vithursan Thangarasa. 2025. Reap the experts: Why pruning prevails for one-shot moe compression. *arXiv preprint arXiv:2510.13999*.

Jaeseong Lee, Seung-won Hwang, Aurick Qiao, Daniel F Campos, Zhewei Yao, and Yuxiong He. 2025. 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)*, pages 13660–13676.

Pingzhi Li, Zhenyu Zhang, Prateek Yadav, Yi-Lin Sung, Yu Cheng, Mohit Bansal, and Tianlong Chen. 2023. Merge, then compress: Demystify efficient smoe with hints from its routing policy. *arXiv preprint arXiv:2310.01334*.

Bill Yuchen Lin, Yuntian Deng, Khyathi Chandu, Faeze Brahman, Abhilasha Ravichander, Valentina Pyatkin, Nouha Dziri, Ronan Le Bras, and Yejin Choi. 2024. Wildbench: Benchmarking llms with challenging tasks from real users in the wild. *arXiv preprint arXiv:2406.04770*.

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

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

Enshu Liu, Junyi Zhu, Zinan Lin, Xuefei Ning, Matthew B Blaschko, Shengen Yan, Guohao Dai, Huazhong Yang, and Yu Wang. 2024b. Efficient expert pruning for sparse mixture-of-experts language models: Enhancing performance and reducing inference costs. *arXiv preprint arXiv:2407.00945*.

Jiawei Liu, Chunqiu Steven Xia, Yuyao Wang, and Lingming Zhang. 2023. Is your code generated by chatgpt really correct? rigorous evaluation of large language models for code generation. *Advances in Neural Information Processing Systems*, 36:21558–21572.

Zongfang Liu, Shengkun Tang, Boyang Sun, Zhiqiang Shen, and Xin Yuan. 2026. Evoesap: Non-uniform expert pruning for sparse moe. *arXiv preprint arXiv:2603.06003*.

Xudong Lu, Qi Liu, Yuhui Xu, Aojun Zhou, Siyuan Huang, Bo Zhang, Junchi Yan, and Hongsheng Li. 2024. Not all experts are equal: Efficient expert pruning and skipping for mixture-of-experts large language models. *arXiv preprint arXiv:2402.14800*.

Ziyang Luo, Can Xu, Pu Zhao, Qingfeng Sun, Xiubo Geng, Wenxiang Hu, Chongyang Tao, Jing Ma, Qingwei Lin, and Daxin Jiang. 2023. Wizardcoder: Empowering code large language models with evol-instruct.

AI Meta. 2025. The llama 4 herd: The beginning of a new era of natively multimodal ai innovation. *https://ai. meta. com/blog/llama-4-multimodal-intelligence/, checked on*, 4(7):2025.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. 2018. Can a suit of armor conduct electricity? a new dataset for open book question answering. *arXiv preprint arXiv:1809.02789*.

Niklas Muennighoff, Luca Soldaini, Dirk Groeneveld, Kyle Lo, Jacob Morrison, Sewon Min, Weijia Shi, Pete Walsh, Oyvind Tafjord, Nathan Lambert, and 1 others. 2024. Olmoe: Open mixture-of-experts language models. *arXiv preprint arXiv:2409.02060*.

Ben Mussay, Margarita Osadchy, Vladimir Braverman, Samson Zhou, and Dan Feldman. 2019. Data-independent neural pruning via coresets. *arXiv preprint arXiv:1907.04018*.

Alexandre Muzio, Alex Sun, and Churan He. 2024. Seer-moe: Sparse expert efficiency through regularization for mixture-of-experts. *arXiv preprint arXiv:2404.05089*.

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

Qwen Team. 2026. Qwen3.5: Towards native multimodal agents.

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

Ayan Sengupta, Siddhant Chaudhary, and Tanmoy Chakraborty. 2025. You only prune once: Designing calibration-free model compression with policy learning. *arXiv preprint arXiv:2501.15296*.

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. *arXiv preprint arXiv:1701.06538*.

Suraj Srinivas and R Venkatesh Babu. 2015. Data-free parameter pruning for deep neural networks. *arXiv preprint arXiv:1507.06149*.

Kimi Team, Tongtong Bai, Yifan Bai, Yiping Bao, S. H. Cai, Yuan Cao, Y. Charles, H. S. Che, Cheng Chen, Guanduo Chen, Huarong Chen, Jia Chen, Jiahao Chen, Jianlong Chen, Jun Chen, Kefan Chen, Liang Chen, Ruijue Chen, Xinhao Chen, and 307 others. 2026. Kimi k2.5: Visual agentic intelligence. *Preprint*, arXiv:2602.02276.

ModelScope Team. 2024. EvalScope: Evaluation framework for large models.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Mengzhou Xia, Tianyu Gao, Zhiyuan Zeng, and Danqi Chen. 2024. Sheared llama: Accelerating language model pre-training via structured pruning. In *International Conference on Learning Representations*, volume 2024, pages 5385–5409.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, and 1 others. 2025a. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.

Huanrui Yang, Wei Wen, and Hai Li. 2020. Deephoyer: Learning sparser neural network with differentiable scale-invariant sparsity measures. In *International Conference on Learning Representations*.

Xican Yang, Yuanhe Tian, and Yan Song. 2025b. Moe pathfinder: Trajectory-driven expert pruning. *arXiv preprint arXiv:2512.18425*.

Edouard Yvinec, Arnaud Dapogny, Matthieu Cord, and Kevin Bailly. 2022. Red++: Data-free pruning of deep neural networks via input splitting and output merging. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(3):3664–3676.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. Hellaswag: Can a machine really finish your sentence? *arXiv preprint arXiv:1905.07830*.

Geng Zhang, Yuxuan Han, Yuxuan Lou, Yiqi Zhang, Wangbo Zhao, and Yang You. 2026. Mone: Replacing redundant experts with lightweight novices for structured pruning of moe. *Preprint*, arXiv:2507.00390.

Zeliang Zhang, Xiaodong Liu, Hao Cheng, Chenliang Xu, and Jianfeng Gao. 2025. Diversifying the expert knowledge for task-agnostic pruning in sparse mixture-of-experts. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 86–102.

Yixiao Zhou, Ziyu Zhao, Dongzhou Cheng, Jie Gui, Yi Yang, Fei Wu, Yu Cheng, Hehe Fan, and 1 others. 2025. Dropping experts, recombining neurons: Retraining-free pruning for sparse mixture-of-experts llms. *arXiv preprint arXiv:2509.10377*.

Table 5: **Architectural diversity of the evaluated MoE models.**

Model	Total params	Experts per layer	Routing sparsity	Shared experts	Dense/MoE layout
OLMoE-1B-7B-0125-Instruct	7B	64	top-8	No	0 dense, 16 MoE layers
DeepSeek-V2-Lite-Chat	16B	64 routed	top-6	Yes (2 shared)	1 dense, 26 MoE layers
ERNIE-4.5-21B-A3B-PT	21B	64 routed	top-6	Yes (2 shared)	1 dense, 27 MoE layers
Qwen3-30B-A3B-Instruct-2507	30B	128	top-8	No	0 dense, 48 MoE layers
Mixtral-8x7B-Instruct-v0.1	47B	8	top-2	No	0 dense, 32 MoE layers

A Model Architecture Details

Table 5 summarizes the architectural diversity of the evaluated MoE models.

B Score Definitions of Pruning Criteria

For completeness, we spell out the expert-ranking scores used by the pruning criteria compared in Table 1. Let $\mathcal{E}(t)$ denote the experts selected for token t, let $g_i(t)$ denote the router weight of expert i on token t, and let $A_i(h_t)$ denote expert i's output activation on hidden state h_t. We write

$$n_i = \sum_{t:\, i \in \mathcal{E}(t)} 1$$

for the number of tokens routed to expert i.

Frequency. Frequency scores an expert by how often it is selected:

$$s_i^{\text{Freq}} = \sum_{t:\, i \in \mathcal{E}(t)} 1.$$

This is a pure routing-frequency signal and depends only on the routing decisions collected from a calibration set.

SEER soft counting. SEER replaces hard counts with the accumulated router weights:

$$s_i^{\text{SEER}} = \sum_{t:\, i \in \mathcal{E}(t)} g_i(t).$$

Compared with Frequency, this still depends on routing events from a calibration set, but it also uses the router confidence assigned to each selected expert.

Expert Activation Norm (EAN). EAN measures an expert by the accumulated norm of its output activations:

$$s_i^{\text{EAN}} = \sum_{t:\, i \in \mathcal{E}(t)} \|A_i(h_t)\|_2.$$

This criterion depends on expert activations collected over the calibration tokens rather than on router weights.

REAP. REAP combines router weights and activation magnitudes, then normalizes by the number of routed tokens:

$$s_i^{\text{REAP}} = \frac{1}{n_i} \sum_{t:\, i \in \mathcal{E}(t)} g_i(t)\, \|A_i(h_t)\|_2.$$

It therefore uses both routing information and expert activations estimated from a calibration set.

C Full Benchmark Comparison

Tables 6, 7, and 8 report the full benchmark comparison. We use the same method names as in the main text, and bold and underlined entries denote the best and second-best distinct pruned results within each model and pruning-ratio block.

Table 6: Full benchmark comparison on WildBench and math benchmarks.

Model	Ratio	Method	WildBench	Math		
			WildBench	GSM8K	MATH-500	Math Avg
OLMoE	0%	Full	0.444	0.682	0.222	0.452
	25%	Frequency	0.127	0.033	0.024	0.028
		SEER	0.141	0.037	0.012	0.025
		EAN	0.205	0.115	0.006	0.061
		REAP	0.260	0.202	**0.056**	0.129
		Random	**0.297**	0.140	0.034	0.087
		Magnitude	0.020	0.072	0.004	0.038
		STUN	0.089	0.072	0.008	0.040
		PruneNet	0.273	0.162	0.044	0.103
		AIMER (Ours)	0.236	**0.221**	**0.056**	**0.139**
ERNIE	0%	Full	0.479	0.829	0.780	0.804
	25%	Frequency	0.352	0.647	0.316	0.482
		SEER	0.381	0.748	0.368	0.558
		EAN	0.396	0.672	0.356	0.514
		REAP	**0.405**	0.782	0.474	0.628
		Random	0.377	0.767	0.606	0.687
		Magnitude	0.228	0.747	0.640	0.693
		STUN	0.231	0.662	0.558	0.610
		PruneNet	0.381	0.764	0.610	0.687
		AIMER (Ours)	0.402	**0.830**	**0.692**	**0.761**
	50%	Frequency	0.164	0.051	0.018	0.034
		SEER	0.174	0.106	0.020	0.063
		EAN	**0.253**	0.065	0.026	0.046
		REAP	0.199	0.567	0.178	0.373
		Random	0.158	0.420	0.198	0.309
		Magnitude	0.077	0.351	0.130	0.240
		STUN	0.044	0.103	0.066	0.085
		PruneNet	0.180	0.461	0.154	0.307
		AIMER (Ours)	0.217	**0.650**	**0.348**	**0.499**
DeepSeek	0%	Full	0.418	0.610	0.298	0.454
	25%	Frequency	0.291	0.023	0.012	0.017
		SEER	0.155	0.034	0.016	0.025
		EAN	**0.307**	0.290	0.018	0.154
		REAP	0.152	0.242	0.046	0.144
		Random	0.050	0.081	0.018	0.050
		Magnitude	0.003	0.008	0.002	0.005
		STUN	0.137	0.077	0.030	0.053
		PruneNet	0.221	0.281	0.038	0.160
		AIMER (Ours)	0.246	**0.493**	**0.134**	**0.313**
Qwen3	0%	Full	0.644	0.923	0.802	0.862
	25%	Frequency	**0.632**	0.904	0.196	0.550
		SEER	0.612	0.913	0.202	0.557
		EAN	0.626	0.907	0.200	0.554
		REAP	0.568	**0.926**	0.778	**0.852**
		Random	0.461	0.807	0.574	0.691
		Magnitude	0.119	0.884	**0.808**	0.846
		STUN	0.001	0.001	0.004	0.002
		PruneNet	0.546	0.616	0.502	0.559
		AIMER (Ours)	0.604	0.895	0.756	0.826
	50%	Frequency	0.015	0.000	0.000	0.000
		SEER	0.018	0.000	0.000	0.000
		EAN	**0.522**	0.650	0.032	0.341
		REAP	0.226	**0.895**	**0.804**	**0.849**
		Random	0.027	0.085	0.038	0.061
		Magnitude	0.000	0.000	0.000	0.000
		STUN	0.000	0.002	0.006	0.004
		PruneNet	0.151	0.387	0.250	0.319
		AIMER (Ours)	0.413	0.664	0.494	0.579
Mixtral	0%	Full	0.427	0.661	0.294	0.478
	25%	Frequency	0.356	0.236	0.058	0.147
		SEER	0.135	0.102	0.012	0.057
		EAN	0.384	0.475	0.156	0.316
		REAP	0.397	0.503	**0.176**	0.340
		Random	0.398	**0.541**	0.144	0.342
		Magnitude	0.387	0.453	0.122	0.287
		STUN	0.387	0.523	**0.176**	**0.350**
		PruneNet	0.389	0.396	0.124	0.260
		AIMER (Ours)	**0.431**	0.485	0.148	0.317
	50%	Frequency	0.139	0.054	0.020	0.037
		SEER	0.130	0.070	0.010	0.040
		EAN	0.188	0.205	0.038	0.121
		REAP	0.244	**0.347**	0.054	**0.201**
		Random	0.218	0.177	0.038	0.107
		Magnitude	0.217	0.138	0.022	0.080
		STUN	0.260	0.224	**0.058**	0.141
		PruneNet	0.189	0.157	0.022	0.089
		AIMER (Ours)	**0.263**	0.209	0.040	0.125

Table 7: Full benchmark comparison on coding benchmarks.

Model	Ratio	Method	Code						
			HumanEval	HumanEval+	MBPP	MBPP+	Eval+	LiveCode	Code Avg
OLMoE	0%	Full	0.354	0.323	0.373	0.312	0.341	0.033	0.187
	25%	Frequency	0.000	0.000	0.000	0.000	0.000	0.000	0.000
		SEER	0.000	0.000	0.000	0.000	0.000	0.000	0.000
		EAN	0.000	0.000	0.000	0.000	0.000	0.000	0.000
		REAP	0.012	0.012	0.016	0.016	0.014	0.000	0.007
		Random	**0.079**	**0.073**	**0.164**	**0.132**	**0.112**	0.000	**0.056**
		Magnitude	0.000	0.000	0.000	0.000	0.000	0.000	0.000
		STUN	0.000	0.000	0.000	0.000	0.000	0.000	0.000
		PruneNet	0.067	0.067	0.140	**0.132**	0.102	0.000	0.051
		AIMER (Ours)	0.018	0.018	0.037	0.029	0.026	0.000	0.013
ERNIE	0%	Full	0.909	0.878	0.915	0.765	0.867	0.247	0.557
	25%	Frequency	0.201	0.165	0.360	0.288	0.254	0.055	0.154
		SEER	0.232	0.195	0.317	0.278	0.256	0.060	0.158
		EAN	0.250	0.201	0.317	0.262	0.258	0.044	0.151
		REAP	0.177	0.159	0.272	0.222	0.208	0.044	0.126
		Random	0.677	0.640	0.725	0.606	0.662	0.170	0.416
		Magnitude	0.713	0.671	0.741	0.624	0.687	0.181	0.434
		STUN	0.744	0.695	0.751	0.653	0.711	0.143	0.427
		PruneNet	0.665	0.646	0.720	0.601	0.658	**0.192**	0.425
		AIMER (Ours)	**0.762**	**0.738**	**0.767**	**0.669**	**0.734**	0.176	**0.455**
	50%	Frequency	0.012	0.012	0.003	0.003	0.007	0.000	0.004
		SEER	0.000	0.000	0.005	0.005	0.003	0.000	0.002
		EAN	0.000	0.000	0.019	0.019	0.009	0.000	0.004
		REAP	0.018	0.012	0.040	0.026	0.024	0.000	0.012
		Random	0.189	0.165	0.288	0.238	0.220	0.011	0.116
		Magnitude	0.244	0.195	0.288	0.243	0.243	0.049	0.146
		STUN	0.091	0.091	0.156	0.130	0.117	0.005	0.061
		PruneNet	0.110	0.104	**0.307**	**0.275**	0.199	0.022	0.111
		AIMER (Ours)	**0.287**	**0.250**	0.254	0.225	**0.254**	**0.071**	**0.163**
DeepSeek	0%	Full	0.591	0.524	0.585	0.497	0.549	0.104	0.327
	25%	Frequency	0.000	0.000	0.000	0.000	0.000	0.000	0.000
		SEER	0.000	0.000	0.000	0.000	0.000	0.000	0.000
		EAN	0.000	0.000	0.000	0.000	0.000	0.000	0.000
		REAP	0.000	0.000	0.003	0.003	0.001	0.000	0.001
		Random	0.000	0.000	0.000	0.000	0.000	0.000	0.000
		Magnitude	0.000	0.000	0.000	0.000	0.000	0.000	0.000
		STUN	0.030	0.030	0.106	0.093	0.065	**0.011**	0.038
		PruneNet	0.146	0.122	**0.265**	**0.209**	**0.185**	0.005	**0.095**
		AIMER (Ours)	**0.152**	**0.140**	0.196	0.164	0.163	**0.011**	0.087
Qwen3	0%	Full	0.939	0.902	0.892	0.751	0.871	0.368	0.619
	25%	Frequency	0.000	0.000	0.000	0.000	0.000	0.000	0.000
		SEER	0.006	0.006	0.000	0.000	0.003	0.000	0.002
		EAN	0.000	0.000	0.000	0.000	0.000	0.000	0.000
		REAP	**0.927**	0.872	**0.905**	**0.751**	**0.864**	0.313	0.589
		Random	0.884	0.866	0.831	0.712	0.823	0.286	0.554
		Magnitude	0.890	0.854	0.892	0.746	0.845	0.313	0.579
		STUN	0.000	0.000	0.000	0.000	0.000	0.000	0.000
		PruneNet	0.854	0.793	0.817	0.680	0.786	0.286	0.536
		AIMER (Ours)	0.921	**0.890**	0.862	0.706	0.845	**0.346**	**0.595**
	50%	Frequency	0.000	0.000	0.000	0.000	0.000	0.000	0.000
		SEER	0.000	0.000	0.000	0.000	0.000	0.000	0.000
		EAN	0.000	0.000	0.000	0.000	0.000	0.000	0.000
		REAP	0.061	0.061	0.050	0.045	0.054	0.038	0.046
		Random	0.006	0.006	0.013	0.013	0.010	0.005	0.007
		Magnitude	0.000	0.000	0.000	0.000	0.000	0.000	0.000
		STUN	0.000	0.000	0.000	0.000	0.000	0.000	0.000
		PruneNet	0.110	0.104	0.209	0.180	0.151	0.055	0.103
		AIMER (Ours)	**0.591**	**0.549**	**0.690**	**0.574**	**0.601**	**0.121**	**0.361**
Mixtral	0%	Full	0.543	0.488	0.558	0.460	0.512	0.148	0.330
	25%	Frequency	0.232	0.189	0.381	0.307	0.277	0.038	0.158
		SEER	0.055	0.055	0.233	0.206	0.137	0.000	0.069
		EAN	0.341	0.287	0.500	0.426	0.388	0.082	0.235
		REAP	0.384	0.329	0.481	0.426	0.405	0.093	0.249
		Random	**0.402**	**0.378**	**0.537**	**0.468**	**0.446**	0.088	**0.267**
		Magnitude	0.311	0.274	0.386	0.315	0.322	0.033	0.177
		STUN	**0.402**	0.348	0.458	0.373	0.395	0.082	0.239
		PruneNet	0.293	0.262	0.429	0.370	0.338	0.066	0.202
		AIMER (Ours)	0.366	0.311	0.474	0.415	0.391	**0.104**	0.247
	50%	Frequency	0.024	0.024	0.048	0.037	0.033	0.000	0.017
		SEER	0.018	0.018	0.053	0.050	0.035	0.000	0.018
		EAN	0.128	0.116	0.204	0.161	0.152	0.000	0.076
		REAP	0.122	0.122	0.241	0.180	0.166	0.011	0.089
		Random	0.085	0.067	0.138	0.116	0.102	0.016	0.059
		Magnitude	0.073	0.067	0.093	0.071	0.076	0.000	0.038
		STUN	**0.183**	**0.152**	**0.294**	0.228	**0.214**	0.027	0.120
		PruneNet	0.073	0.073	0.188	0.151	0.121	0.005	0.063
		AIMER (Ours)	0.140	0.128	0.288	**0.235**	0.198	**0.044**	**0.121**

Table 8: Full benchmark comparison on multiple-choice benchmarks.

			Multiple Choice								
Model	**Ratio**	**Method**	**MMLU**	**ARC-C**	**ARC-E**	**HellaSwag**	**BoolQ**	**OpenBookQA**	**RTE**	**WinoGrande**	**MC Avg**
OLMoE	0%	Full	0.534	0.490	0.758	0.808	0.766	0.470	0.711	0.684	0.653
	25%	Frequency	0.325	0.382	0.582	0.727	0.691	0.416	0.690	**0.665**	0.560
		SEER	0.320	0.388	0.590	0.733	0.694	0.414	0.711	0.661	0.564
		EAN	0.347	0.432	0.627	0.748	**0.734**	0.430	0.632	**0.665**	0.577
		REAP	0.391	**0.462**	**0.685**	**0.766**	0.727	**0.448**	0.581	0.642	0.588
		Random	0.432	0.403	0.598	0.705	0.706	0.430	0.599	0.621	0.562
		Magnitude	0.311	0.322	0.475	0.583	0.699	0.310	0.567	0.560	0.478
		STUN	0.384	0.340	0.503	0.629	0.585	0.404	0.545	0.585	0.497
		PruneNet	0.350	0.421	0.620	0.707	0.694	0.406	0.686	0.622	0.563
		AIMER (Ours)	**0.435**	0.433	0.675	0.733	0.733	0.408	**0.758**	0.628	**0.600**
ERNIE	0%	Full	0.739	0.564	0.782	0.814	0.872	0.462	0.816	0.717	0.721
	25%	Frequency	0.571	0.518	0.727	0.719	0.849	0.390	0.791	0.679	0.655
		SEER	0.599	0.511	0.750	0.736	0.845	0.400	0.747	0.676	0.658
		EAN	0.593	**0.534**	0.753	**0.789**	0.844	0.452	0.805	**0.721**	**0.687**
		REAP	0.643	0.499	0.717	0.780	0.847	**0.454**	**0.827**	0.711	0.685
		Random	0.643	0.514	0.732	0.751	0.833	0.408	0.751	0.681	0.664
		Magnitude	0.594	0.507	0.752	0.699	0.821	0.416	0.744	0.654	0.648
		STUN	0.656	0.514	**0.756**	0.648	0.833	0.360	0.726	0.592	0.636
		PruneNet	**0.665**	0.506	0.694	0.758	0.853	0.448	0.787	0.706	0.677
		AIMER (Ours)	0.647	0.514	0.718	0.775	**0.864**	0.430	0.805	0.694	0.681
	50%	Frequency	0.488	0.386	0.583	0.579	0.740	0.324	0.733	0.625	0.557
		SEER	0.460	0.386	0.599	0.573	0.711	0.324	0.675	0.613	0.543
		EAN	0.481	**0.447**	**0.675**	**0.678**	0.725	**0.398**	0.744	**0.692**	**0.605**
		REAP	0.479	0.422	0.621	0.659	0.768	0.392	0.740	0.651	0.592
		Random	**0.523**	0.440	0.630	0.613	0.767	0.362	0.733	0.601	0.584
		Magnitude	0.414	0.357	0.572	0.508	0.716	0.324	0.592	0.543	0.503
		STUN	0.422	0.337	0.506	0.476	0.649	0.326	0.635	0.542	0.487
		PruneNet	0.493	0.423	0.607	0.626	0.774	0.360	**0.791**	0.617	0.586
		AIMER (Ours)	0.503	0.426	0.641	0.659	**0.815**	0.354	0.708	0.680	0.598
DeepSeek	0%	Full	0.567	0.541	0.785	0.808	0.829	0.456	0.726	0.712	0.678
	25%	Frequency	0.417	0.428	0.667	0.751	0.729	0.424	**0.726**	0.671	0.602
		SEER	0.394	0.451	0.680	0.754	0.722	0.422	**0.726**	0.669	0.602
		EAN	0.445	0.477	0.700	0.780	0.754	**0.434**	0.711	**0.699**	0.625
		REAP	**0.485**	**0.508**	**0.756**	**0.784**	**0.771**	**0.434**	0.693	0.693	**0.640**
		Random	0.459	0.319	0.438	0.559	0.710	0.382	0.675	0.626	0.521
		Magnitude	0.288	0.269	0.324	0.456	0.668	0.270	0.549	0.533	0.419
		STUN	0.409	0.391	0.641	0.650	0.717	0.366	0.625	0.635	0.554
		PruneNet	0.438	0.451	0.673	0.723	**0.771**	0.400	0.708	0.669	0.604
		AIMER (Ours)	0.454	0.481	0.711	0.749	0.761	0.412	0.718	0.683	0.621
Qwen3	0%	Full	0.802	0.625	0.838	0.797	0.887	0.446	0.769	0.736	0.737
	25%	Frequency	0.761	0.625	0.839	0.795	0.886	0.442	0.773	0.732	0.732
		SEER	0.760	0.630	0.845	0.795	**0.888**	**0.446**	0.776	0.732	0.734
		EAN	**0.765**	0.631	**0.849**	**0.796**	0.887	0.444	0.780	**0.736**	**0.736**
		REAP	0.733	**0.637**	0.848	0.766	0.871	**0.446**	0.780	0.718	0.725
		Random	0.717	0.531	0.763	0.707	0.883	0.402	0.762	0.707	0.684
		Magnitude	0.714	0.560	0.803	0.665	0.859	0.398	0.718	0.642	0.670
		STUN	0.240	0.247	0.263	0.270	0.508	0.234	0.520	0.506	0.348
		PruneNet	0.721	0.573	0.813	0.716	0.874	0.402	0.783	0.714	0.700
		AIMER (Ours)	0.715	0.615	0.842	0.772	0.886	0.428	**0.794**	0.732	0.723
	50%	Frequency	0.276	0.287	0.391	0.436	0.655	0.306	0.585	0.564	0.437
		SEER	0.276	0.288	0.396	0.435	0.651	0.302	0.567	0.571	0.436
		EAN	**0.634**	**0.544**	**0.770**	**0.786**	**0.885**	**0.440**	**0.783**	**0.731**	**0.697**
		REAP	0.532	0.537	0.734	0.615	0.836	0.370	0.729	0.674	0.628
		Random	0.238	0.280	0.365	0.393	0.649	0.262	0.581	0.501	0.409
		Magnitude	0.263	0.238	0.328	0.291	0.514	0.302	0.516	0.527	0.372
		STUN	0.239	0.230	0.271	0.265	0.501	0.236	0.484	0.508	0.342
		PruneNet	0.580	0.451	0.707	0.573	0.696	0.348	0.780	0.622	0.595
		AIMER (Ours)	0.542	0.482	0.724	0.653	0.847	0.378	0.769	0.712	0.638
Mixtral	0%	Full	0.691	0.649	0.841	0.862	0.887	0.500	0.726	0.742	0.737
	25%	Frequency	0.403	0.467	0.676	0.759	0.804	0.424	0.625	0.592	0.594
		SEER	0.260	0.317	0.467	0.626	0.675	0.328	0.531	0.522	0.466
		EAN	0.614	**0.610**	0.820	0.840	0.871	**0.494**	0.686	0.732	0.708
		REAP	0.630	0.609	**0.823**	0.835	0.875	0.482	**0.765**	0.713	**0.717**
		Random	0.620	0.603	0.798	0.829	0.859	0.482	0.704	0.725	0.702
		Magnitude	0.518	0.495	0.724	0.763	0.803	0.420	0.650	0.634	0.626
		STUN	**0.639**	**0.610**	0.795	0.831	0.869	0.488	0.740	0.729	0.713
		PruneNet	0.473	0.490	0.728	0.751	0.800	0.410	0.625	0.592	0.609
		AIMER (Ours)	0.627	0.603	0.822	**0.843**	**0.876**	0.486	0.671	**0.738**	0.708
	50%	Frequency	0.323	0.381	0.572	0.662	0.745	0.366	0.549	0.566	0.520
		SEER	0.329	0.381	0.600	0.678	0.720	0.366	0.541	0.570	0.523
		EAN	0.530	0.550	0.754	**0.802**	0.853	**0.458**	0.737	0.707	**0.674**
		REAP	**0.542**	0.527	0.758	0.791	0.861	0.444	**0.751**	0.688	0.670
		Random	0.407	0.414	0.634	0.693	0.782	0.402	0.639	0.601	0.572
		Magnitude	0.321	0.355	0.548	0.577	0.683	0.354	0.549	0.542	0.491
		STUN	0.522	0.503	0.743	0.760	0.819	0.438	0.722	0.688	0.649
		PruneNet	0.367	0.396	0.619	0.653	0.756	0.352	0.607	0.558	0.538
		AIMER (Ours)	0.524	**0.567**	**0.779**	0.795	**0.867**	0.446	0.693	**0.709**	0.672

D Effect of Calibration Data Choice

Table 9 reports REAP results under several calibration data choices, including C4, Evol-CodeAlpaca-v1, and Evol-CodeAlpaca-v1–C4 mixtures, with AIMER included as a calibration-free reference. All REAP settings use the same calibration budget of approximately 4.2M tokens. These results show that calibration-based pruning can achieve balanced performance when the calibration data and mixture proportion are adjusted. For example, adding code data substantially improves coding performance over C4-only calibration in several settings, while increasing the C4 proportion often helps recover WildBench or multiple-choice performance. However, the best trade-off varies across models and pruning ratios, so this benefit comes with an additional calibration-data selection step. AIMER avoids tuning the calibration corpus or mixture ratio while remaining a simpler and effective task-agnostic alternative.

Table 9: **Effect of calibration data choice.** We report summary performance using C4, CodeAlpaca, and CodeAlpaca–C4 mixtures. All REAP settings use approximately 4.2M calibration tokens; CodeAlpaca denotes Evol-CodeAlpaca-v1.

Model	Ratio	Calibration Data	Method	Code Avg	WildBench	MC Avg
ERNIE	25%	C4	REAP	0.126	0.405	0.685
		CodeAlpaca	REAP	**0.534**	0.352	0.665
		CodeAlpaca:C4 = 1:1	REAP	0.520	0.410	**0.692**
		CodeAlpaca:C4 = 1:3	REAP	0.501	**0.426**	**0.692**
		N/A	AIMER (Ours)	0.455	0.402	0.681
	50%	C4	REAP	0.012	0.199	0.592
		CodeAlpaca	REAP	**0.457**	0.204	0.577
		CodeAlpaca:C4 = 1:1	REAP	0.434	0.218	0.586
		CodeAlpaca:C4 = 1:3	REAP	0.301	**0.236**	**0.598**
		N/A	AIMER (Ours)	0.163	0.217	**0.598**
Qwen3	25%	C4	REAP	0.589	0.568	0.725
		CodeAlpaca	REAP	0.617	0.572	0.706
		CodeAlpaca:C4 = 1:1	REAP	**0.625**	0.591	0.738
		CodeAlpaca:C4 = 1:3	REAP	0.602	**0.607**	**0.740**
		N/A	AIMER (Ours)	0.595	0.604	0.723
	50%	C4	REAP	0.046	0.226	0.628
		CodeAlpaca	REAP	**0.590**	0.285	0.596
		CodeAlpaca:C4 = 1:1	REAP	0.545	0.329	0.666
		CodeAlpaca:C4 = 1:3	REAP	0.547	0.301	**0.670**
		N/A	AIMER (Ours)	0.361	**0.413**	0.638

E Radar Plots by Benchmark

Figures 5, 6, 7, and 8 visualize the per-benchmark trade-offs behind the appendix tables for DeepSeek at 25% pruning, ERNIE-21B at 25% and 50% pruning, Mixtral at 25% and 50% pruning, and Qwen3-30B and OLMoE-7B at 25% pruning. The dashed outline denotes the dense model, and each colored trace corresponds to one pruning method.

The radar plots make a point that is less visible from category averages alone: the key distinction between pruning criteria is not only how much performance they preserve, but how evenly that performance is preserved across benchmark families. Across settings, AIMER tends to produce a smoother contraction of the dense-model contour, whereas several baselines preserve one capability cluster while collapsing sharply on others. For task-agnostic pruning, this difference matters because a method that wins on a few spokes but caves in on the rest is not preserving general ability in a meaningful sense.

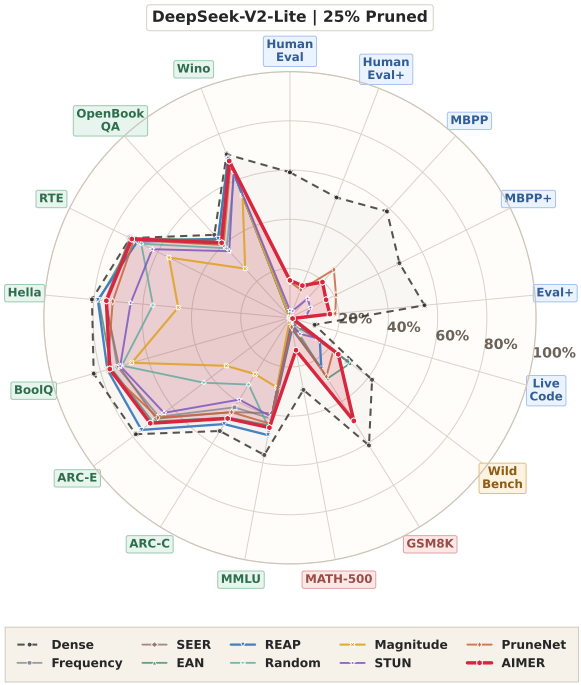

Figure 5: **Radar plot of DeepSeek performance across all benchmarks at 25% pruning.**

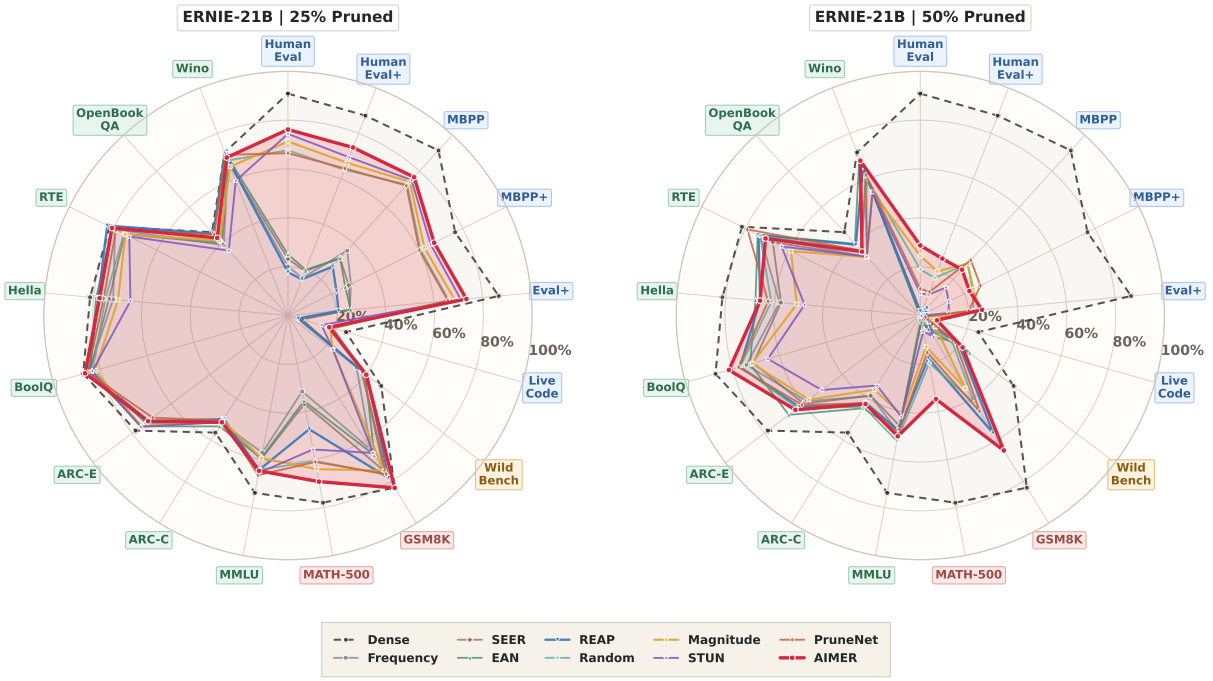

Figure 6: **Radar plots of ERNIE-21B performance across all benchmarks.** The left and right panels show 25% and 50% pruning ratios, respectively.

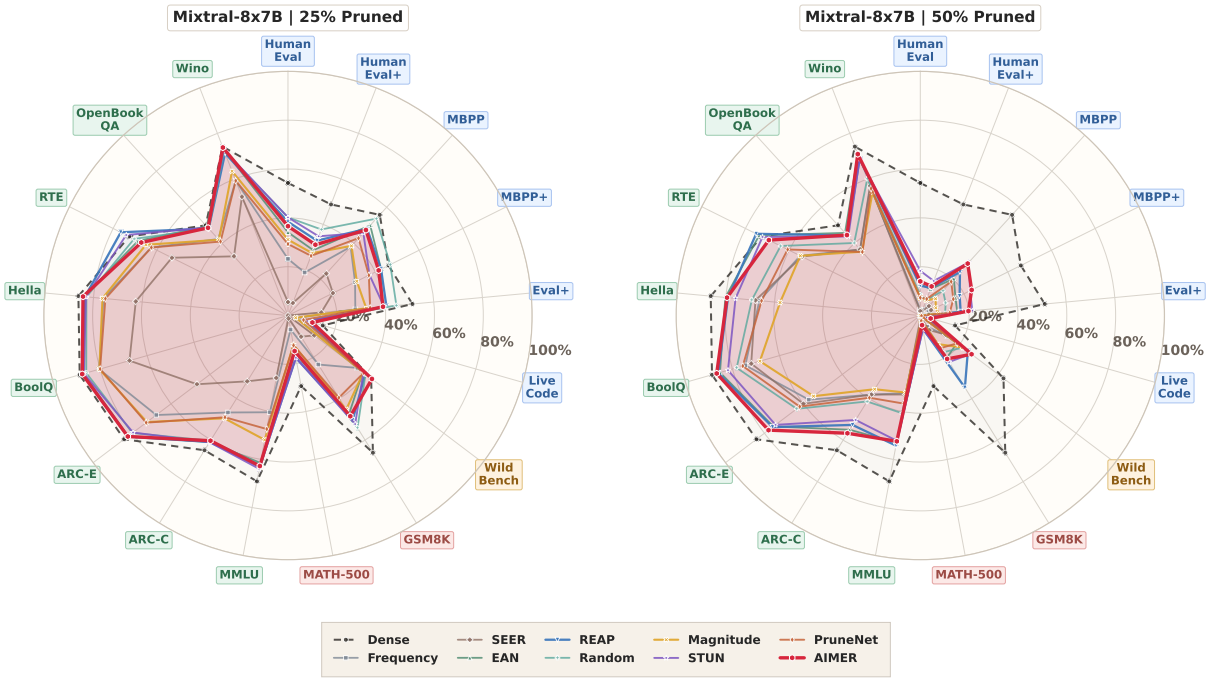

Figure 7: **Radar plots of Mixtral performance across all benchmarks.** The left and right panels show 25% and 50% pruning ratios, respectively.

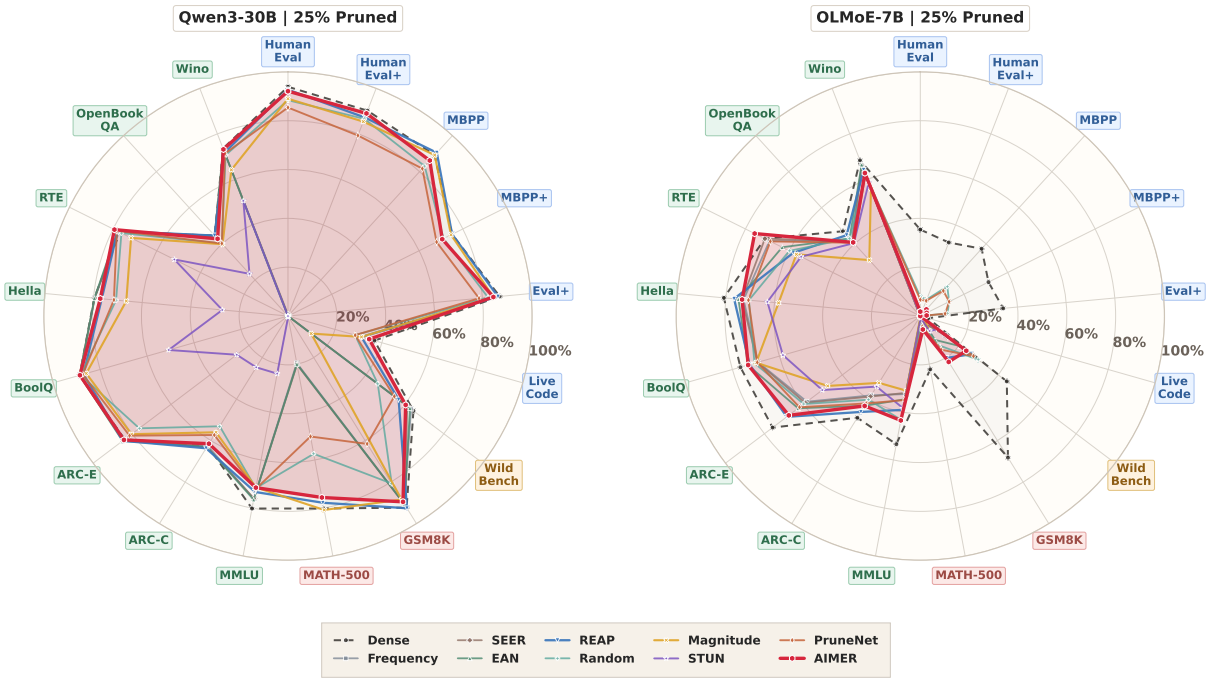

Figure 8: **Radar plots of Qwen3-30B and OLMoE-7B performance across all benchmarks at 25% pruning.** The left and right panels show Qwen3-30B and OLMoE-7B, respectively.