Preserving Long-Tailed Expert Information in Mixture-of-Experts Tuning

Haoze He[1] **Xingyuan Ding**[1] **Xuan Jiang**[2] **Xinkai Zou**[3] **Alex Cheng**[1] **Yibo Zhao**[4] **Juncheng Billy Li**[1*] **Heather Miller**[1*]

[1]Carnegie Mellon University [2]MIT [3]UCSD [4]Johns Hopkins University

{haozeh, xingyuad, abcheng, junchenl, heather.miller}@cs.cmu.edu
xuanj@mit.edu x9zou@ucsd.edu yzhao231@jh.edu

https://github.com/HectorHHZ/ExpertCondenser

Abstract

Despite MoE models leading many benchmarks, supervised fine-tuning (SFT) for the MoE architectures remains difficult because its router layers are fragile. Methods such as DenseMixer and ESFT mitigate router collapse with dense mixing or auxiliary load-balancing losses, but these introduce noisy gradients that often degrade performance. In preliminary experiments, we systematically pruned experts and observed that while certain "super experts" are activated far more frequently, discarding less used experts still leads to notable performance degradation. This suggests that even rarely activated experts encode non-trivial knowledge useful for downstream tasks. Motivated by this, we propose an auxiliary-loss-free MoE SFT framework that combines bias-driven sparsification with always-active gated condenser experts. Rather than enforcing balanced activation across all experts, our method encourages task-relevant experts to remain active while pushing long-tailed experts toward inactivity. The condenser experts provide a persistent, learnable pathway that alleviates gradient starvation and facilitates consolidation of information that would otherwise remain fragmented across sparsely activated experts. Analysis further suggest that this design better preserves long-tailed expert information under sparse routing. Experiments on large-scale MoE models demonstrate that our approach outperforms state-of-the-art SFT baselines such as DenseMixer and ESFT, achieving average gain of 2.5%+ on both mathematical reasoning and commonsenseQA benchmarks.

1 Introduction

Mixture-of-Experts (MoE) models scale language models efficiently by activating only a small subset of experts per token, enabling massive capacity without increasing per-token compute. Yet the same sparse routing that drives their success also makes them fragile: MoE relies on a non-differentiable Top-K router, which blocks straightforward gradient flow and makes post-training, such as supervised fine-tuning (SFT), far more difficult than for dense LLMs.

Over the years, researchers have sought to stabilize MoE training through progressively refined routing strategies. GShard (Lepikhin et al., 2020) introduced top-2 gating with heavy auxiliary balancing losses, while Switch Transformers (Fedus et al., 2022) simplified this to a single expert per token. More recent work, such as DeepSeek-MoE (Wang et al., 2024a) and DeepSeek-V3 (Liu et al., 2024b), explored bias-based routers and minimized auxiliary losses to reduce gradient noise and improve efficiency. However, these advances primarily address pre-training. In the post-training setting, SFT remains underexplored: ESFT (Wang

*Co-advisor.

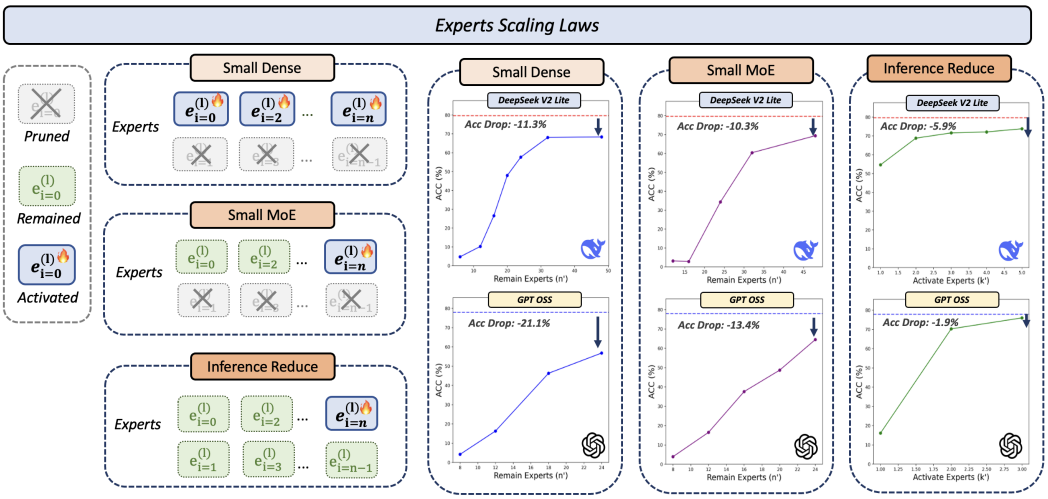


Figure 1: Illustration of the three expert scaling strategies. (*Small-Dense*) Experts are pruned, and all remaining experts are always activated. (*Small MoE*) Experts are pruned and only a subset of them are activated per token. (*Inference Reduce*) Fewer experts are selected per token while the expert pool is unchanged. A substantial performance gap remains between the base model and pruned variants under Small Dense and small MoE categories.

et al., 2024b) routes all gradients to the most-activated expert, while DenseMixer (Yao et al., 2025) improves slightly by applying a Straight-Through Estimator (STE) (Bengio et al., 2013) to approximate updates for inactive experts, yet STE introduces biased gradients.

In parallel, recent studies have identified the existence of "super experts"Su et al. (2025) or "super weights"Yu et al. (2025), whose activations dominate the routing and whose removal leads to sharp performance degradation. These findings suggest that a small subset of experts carries disproportionate importance. However, our observations reveal a complementary phenomenon: even the rarely activated experts encode indispensable information, and pruning them also causes substantial performance decline. This highlights the need for fine-tuning strategies that not only preserve the capacity of frequently activated super experts, but also consolidate information from the long tail of rarely activated experts.

Motivated by these observations, we introduce **condenser experts**: experts that are always selected yet remain gated per token. This creates a learnable shared pathway that alleviates gradient starvation, narrows the train–inference routing gap, and facilitates preservation of information that would otherwise remain distributed across long-tailed experts. Experimental results show that our method consistently outperforms SoTA MoE SFT methods by 2.5%+ when post-training popular MoE LLMs on commonsense reasoning (PIQA, ARC, SIQA) and math reasoning benchmarks (MATH-500, AIME-25, GPQA, GSM8K, etc). Our implementation is open-source.

1.1 Contributions

- We extend scaling-law analysis to MoE compression by relating performance to the number of expert parameters retained. Through this analysis, we find that performance degrades substantially when rarely activated experts are removed, showing that these long-tailed experts encode indispensable knowledge beyond the most frequently activated "super experts."
- We propose **ExpertCondenser**, a MoE post-training method combines auxiliary-free bias-based routing with always-active gated condenser experts to alleviate gradient starvation, narrow the train–inference routing gap, and facilitate consolidation of information from long-tailed experts.

2 Does Saving "Super Experts" Mean Saving Model Performance?

Previous work (Lu et al., 2024; Su et al., 2025) shows that pruning frequently activated "super experts" leads to significant performance degradation, highlighting their importance.

However, it remains unclear whether retaining only these experts is sufficient to preserve model quality.

To answer this, we study how performance scales with the total number of retained expert parameters. We consider three compression strategies (Fig. 1) that vary the expert pool size n and activation budget k. For token t, the selected experts are

$$S_t = \mathrm{TopK}(\{s_{j,t}\}_{j=1}^{n}, k),$$

where $s_{j,t}$ denotes the gating score.

We examine: (i) **Dense conversion via pruning:** $n \to n'$, $k = n'$, activating all remaining experts (dense model); (ii) **Smaller MoE via pruning:** $n \to n'$, $k < n'$, preserving sparsity within a reduced expert pool; (iii) **Reduced activation budget:** n fixed, $k \to k' < k$, increasing sparsity without changing model size. We use ES-Act and ES-Mag (Wang et al., 2024b) to select experts prior to pruning, with ES-Act as default (details in Appendix E).

To conduct a scaling-law analysis of experts, we design experiments using GPT-OSS (OpenAI, 2025) and DeepSeek-Coder-V2-Lite (Liu et al., 2024a). In figure 1, we summarize the results across the three strategies (more details in Appendix D, Tab. 8 and 9). Even though expert activation is highly skewed and a few "super experts" dominate routing (shown in Appendix P, Fig. 9), a substantial performance gap still remains between the base model and pruned variants. Retaining the top 75% of experts still results in more than a 10% drop. **These results show that preserving "Super Experts" is not sufficient to preserve model performance.** Crucially, our experiments reveal that rarely activated experts nonetheless encode indispensable domain knowledge, which is distributed in a fragmented manner across long-tailed experts.

3 Proposed Method

Our findings in Section 2 suggest that effective post-training should preserve knowledge from infrequently activated experts. This motivates a mechanism that consolidates such long-tailed information into a small set of persistent components.

While traditional shared experts provide persistent computation, they are ungated and mix signals uniformly across inputs, limiting expressiveness. More generally, persistent computation paths are more expressive when they remain learnably mixed rather than fixed. A detailed discussion between condenser experts and shared experts is in Appendix B.

We therefore introduce **condenser experts**: experts that are always selected yet remain gated per token. This design enables a persistent but input-adaptive pathway, allowing the model to better preserve and consolidate information that would otherwise remain fragmented across rarely activated experts. We refer to the overall training framework as **ExpertCondenser**.

3.1 Condenser Experts

In a standard MoE layer, for token $\mathbf{x}_t$, the output is typically written as

$$\mathbf{h}_t = \underbrace{\sum_{i=1}^{n} g_i(\mathbf{x}_t)\, E_i(\mathbf{x}_t)}_{\text{sparse experts}} + \underbrace{\sum_{k=1}^{\hat{n}} E_k(\mathbf{x}_t)}_{\text{static shared experts}} \quad , \tag{1}$$

where $E(\cdot)$ are expert functions, $g(\cdot)$ are computed gate weight from the router, and x_t denote token t. The static shared experts are always active with fixed weight, shown as green boxes in Fig. 2.

We introduce condenser experts, that are **always activated yet dynamically weighted** shown in Fig. 2. Let $J \subset \{1, \ldots, n\}$ denote the fixed set of condenser experts with $|J| = r$ and $\tilde{s}_i(x_t)$ denote the routing score for expert i. We discuss the condenser experts selection in § 3.2. For each token x_t, we define the active expert set as

$$S(x_t) = J \cup \mathrm{TopK}_{i \notin J}\left(\tilde{s}_i(x_t),\, k - r\right). \tag{2}$$

We reformulate the MoE output as

$$h'_t = \underbrace{\sum_{j \in J} g_j(x_t)\, E_j(x_t)}_{\text{gated condenser experts}} + \underbrace{\sum_{i \in S(x_t) \setminus J} g_i(x_t)\, E_i(x_t)}_{\text{selected sparse experts}} + \underbrace{\sum_{k=1}^{\hat{n}} E_k(\mathbf{x}_t)}_{\text{static shared experts}} \tag{3}$$

Condenser Experts Avoid Gradient Bottleneck Introduced by Router. Let

$$Z_i(x) = \frac{\partial \ell}{\partial h(x)} \cdot g_i(x)\, \frac{\partial E_i(x)}{\partial \theta_i}.$$

For a routed expert $i \notin J$, the gradient can be written as $\nabla_{\theta_i} \mathcal{L} = \mathbb{E}\big[\mathbf{1}\{i \in S(x)\}\, Z_i(x)\big]$. Let $p_i = \mathbb{P}(i \in S(x))$. By conditioning on the routing event, $\nabla_{\theta_i} \mathcal{L} = p_i\, \mathbb{E}\big[Z_i(x) \mid i \in S(x)\big]$, showing that routing induces a multiplicative attenuation by $p_i < 1$ on expert-specific gradient flow. This attenuation also manifests at the level of training dynamics. Let $B_t^{(i)} = \mathbf{1}\{i \in S(x_t)\}$ denote whether expert i is selected at step t, and define the total number of updates over T steps as $N_i(T) = \sum_{t=1}^{T} B_t^{(i)}$. Let $p_i = \mathbb{P}(i \in S(x))$ denote the marginal activation probability of expert i. Then $\mathbb{E}[N_i(T)] = Tp_i$, so the effective number of updates scales linearly with p_i. In practice, routing distributions are highly non-uniform (Figure 9), and p_i can vary significantly across experts, leading to substantial imbalance in training exposure. Although routing decisions are input-dependent and not strictly independent, $N_i(T)$ behaves similarly to a sum of Bernoulli trials with mean Tp_i. By a Chernoff bound, for any $0 < \delta < 1$,

$$\Pr(N_i(T) \le (1-\delta)Tp_i) \le \exp\left(-\frac{\delta^2}{2} Tp_i\right). \tag{4}$$

Thus, with high probability, routed experts receive only $\Theta(Tp_i)$ effective updates over training. Under mild conditions, $N_i(T)$ concentrates around its mean, so long-tailed experts consistently receive only $\Theta(Tp_i)$ updates over training.

In contrast, for a condenser expert $j \in J$, we have $\nabla_{\theta_j} \mathcal{L} = \mathbb{E}\big[Z_j(x)\big]$, so condenser experts avoid both the per-step gradient attenuation and the reduced effective sample size induced by routing sparsity. This highlights that gradient starvation is fundamentally a multiplicative bottleneck in both gradient magnitude and update frequency, which is eliminated by always-active condenser experts. In § 6.1 we show that ExpertCondenser achieve more substantial differences of MoE outputs than SFT, and in § 6.2, we show the gradient norm condenser experts is consistently larger than routed experts during training, suggesting condenser expert avoid gradient bottleneck.

Condenser Experts Consolidate Domain Information. Condenser experts receive gradients that reflect domain-specific patterns by co-activated with routed experts. Assume the training distribution is a mixture $\mathcal{D} = \sum_{d=1}^{m} \pi_d \mathcal{D}_d$. Then for a condenser expert $j \in J$,

$$\nabla_{\theta_j} \mathcal{L} = \sum_{d=1}^{m} \pi_d \mathbb{E}_{(x,y)\sim\mathcal{D}_d} \left[\frac{\partial \ell}{\partial h(x)} \cdot g_j(x)\, \frac{\partial E_j(x)}{\partial \theta_j} \right], \tag{5}$$

This shows that condenser experts accumulate gradient contributions across all domains, and integrates signals that are otherwise distributed across sparsely activated experts (more details in Appendix S). Given the decomposition in Eq. 3, the gated condenser expert term can be represented as a weighted low-dimensional shared representation that captures common structure across routed experts. Our analysis in § 6.3 empirically suggesting condenser experts consolidate in-domain information.

3.2 Auxiliary Free Sparsity Enforcement

Auxiliary losses inject competing gradient signals that can hinder convergence on specialized downstream tasks. [1] We eliminate this issue by adopting an **auxiliary-loss-free** routing

[1] More details about auxiliary losses can be found in Appendix F and Appendix G.

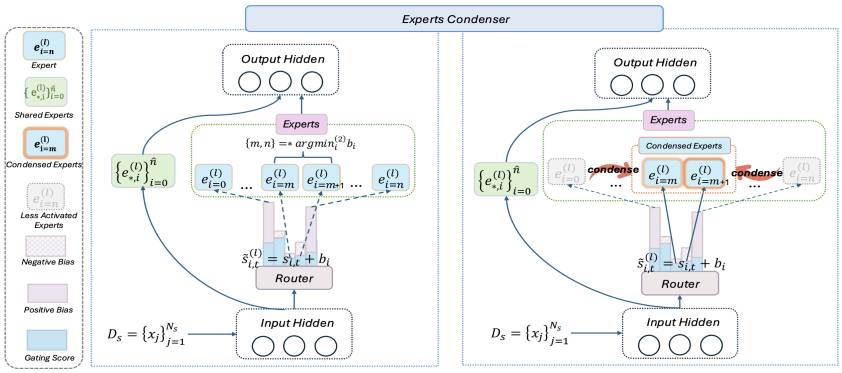


Figure 2: Representation of our Experts Condenser framework. *(Left)* An auxiliary-free router adds trainable biases b_i to logits $s_{i,t}$. Less relevant experts accumulate negative biases, and a small set of experts are designated as condenser experts. *(Right)* These condenser experts are always selected during training, ensuring they receive gradients and act as repositories that condense knowledge from inactive experts—preserving information while enforcing sparsity.

strategy for fine-tuning, where the routing logits are directly modified with trainable bias parameters b_i, one per expert. We define the router's biased logits as

$$\tilde{s}_i(x_t) = s_i(x_t) + b_i. \tag{6}$$

where s_i is the original raw logit for expert i. Following the DeepSeek-2/3 settings (Wang et al., 2024a), final gating weights ρ_i are still computed from original unbiased logits s_i. [2]

Our objective in fine-tuning is to induce task-adaptive sparsification while preserving useful information carried by rarely activated experts. Biases for experts that are less useful for the target task are progressively decreased, separating the expert pool into a task-relevant active set and a long tail of weakly selected experts. The bias mechanism narrows the train–inference routing gap. In § 6.1, we empirically show that the bias mechanism avoid heavily perturbing fragile routers and can further encourage stable training.

Tradeoff in the number of condenser experts. Let r denote the number of condenser experts. Increasing r improves aggregation capacity but reduces gradient concentration per expert, while smaller r increases concentration but may create a capacity bottleneck. This predicts an interior optimum for small r. In practice, we find that $r = 2$ provides a balance between capacity and consolidation. Condenser experts are selected as the r experts with the lowest bias values b_i, which correspond to consistently activated experts under the auxiliary-free routing dynamics. We further validate these design choices through ablation studies: in Appendix Q.1, we analyze the effect of varying r, and in Appendix Q.2, we study different strategies for selecting condenser experts.

4 Background and Related Works

Post-training for Mixture-of-Experts (MoE) large language models remains relatively under-explored. Recent efforts have primarily focused on how to adapt experts so that they better align with downstream domains. Two representative approaches are Expert Supervised Fine-Tuning (ESFT) (Wang et al., 2024b) and DenseMixer (Yao et al., 2025), which propose different strategies for handling gradient propagation through the non-differentiable Top-k routing mechanism.

ESFT [3] ESFT strengthens the role of the most frequently activated experts by routing gradients only through the Top-k set S_t. Formally, if $w_{i,t}$ denotes the routing weight of expert i for token t, then the gradient with respect to router parameters θ is approximated as $\nabla_\theta \mathcal{L} \approx \sum_{i \in S_t} \left(\frac{\partial \mathcal{L}}{\partial y_t} \cdot v_i \right) \frac{\partial w_{i,t}}{\partial \theta}$. Here $S_t = \mathrm{TopK}(\{w_{j,t}\}_{j=1}^{n}, k)$ is the set of selected experts, and v_i is the output of expert i. Notably, ESFT adopts the same router training objective as

[2] Find additional discussion on bias-based routing design and expert selection in Appendix L.
[3] More details about ESFT and DenseMixer are in Appendix H.

Table 1: Evaluation of post-trained models (zero-shot results) on downstream math reasoning datasets, including SingleEq, MultiArith, AddSub, GSM8K, SVAMP, AQuA, and MAWPS. ExpertCondenser achieves relatively stronger performance on simpler arithmetic datasets such as SingleEq and AddSub, which primarily require short-horizon reasoning.

Dataset	Model	Model Size	Post-train Type	GSM8K	SingleEq	SVAMP	MultiArith	AddSub	AQuA	MAWPS	AVG
math7k	Qwen1.5-MoE	14B	SFT	45.8	70.2	53.6	76.2	53.3	27.6	67.8	56.4
			ESFT	46.9	69.3	54.1	75.7	52.2	27.6	68.1	56.2
			DenseMixer	54.6	72.1	52.4	81.8	58.7	27.8	72.5	60.0
			ExpertCondenser (Ours)	57.2	73.6	55.7	86.0	61.8	33.1	75.6	**63.3**
	DeepSeek-V2-Lite	16B	SFT	58.4	80.6	66.2	90.2	61.0	24.8	75.8	65.3
			ESFT	58.6	80.9	65.8	90.7	62.3	27.6	76.1	66.0
			DenseMixer	57.6	88.6	67.8	92.3	72.8	33.1	80.6	70.4
			ExpertCondenser (Ours)	59.4	92.5	69.1	91.5	79.5	36.1	83.6	**73.1**
	OLMoE	7B	SFT	63.6	74.8	67.7	92.3	57.8	27.6	72.4	65.2
			ESFT	62.2	75.2	68.0	93.3	58.2	28.7	73.1	65.5
			DenseMixer	64.6	78.2	68.4	92.3	59.2	36.3	74.2	67.6
			ExpertCondenser (Ours)	68.4	79.8	71.2	93.8	63.4	36.3	78.8	**70.2**
	GPT-OSS	20B	SFT	78.2	92.9	81.2	98.2	82.5	36.8	91.2	80.1
			ESFT	76.6	92.9	80.2	98.2	82.0	35.4	90.3	79.4
			DenseMixer	80.1	92.3	83.2	98.7	82.5	37.4	90.8	80.7
			ExpertCondenser (Ours)	81.7	93.2	82.5	98.5	85.6	38.6	91.6	**81.7**
math14k	Qwen1.5-MoE	14B	SFT	51.8	74.8	55.4	87.2	66.7	27.6	74.8	62.6
			ESFT	52.5	76.0	54.1	86.2	62.3	29.5	71.4	61.7
			DenseMixer	55.8	79.6	57.2	88.7	70.8	31.6	76.6	65.8
			ExpertCondenser (Ours)	58.8	81.2	59.2	91.6	72.8	33.2	78.4	**67.9**
	DeepSeek-V2-Lite	16B	SFT	57.6	76.4	67.6	90.1	59.7	30.6	74.3	65.2
			ESFT	58.2	75.8	65.2	89.0	56.5	29.5	73.5	64.0
			DenseMixer	61.6	79.4	69.6	91.6	57.8	28.6	78.7	66.8
			ExpertCondenser (Ours)	63.6	81.2	71.8	93.8	60.8	33.2	81.4	**69.4**
	OLMoE	7B	SFT	65.3	78.4	69.8	82.8	68.5	31.4	75.8	67.4
			ESFT	64.4	77.0	68.9	81.8	64.1	30.7	74.8	65.9
			DenseMixer	66.2	78.4	68.6	83.6	66.8	32.8	74.6	67.3
			ExpertCondenser (Ours)	67.8	81.6	72.4	86.8	68.8	32.8	79.6	**70.0**

Table 2: Evaluation of post-trained models (*Pass@4*) on downstream math reasoning benchmarks after fine-tuning with *Stanford-S1*, including GPQA Diamond, AIME 24/25, and MATH-500. Note that Qwen1.5 and DeepSeek-Coder-V2-Lite are earlier-generation MoE models with limited mathematical reasoning capacity, leading to near-zero performance on AIME 24/25 across all methods.

Model	Model Size	Distill Type	GPQA Diamond	AIME 2024	AIME 2025	MATH-500	AVG
Qwen1.5-MoE	14B	SFT	27.8	0.8(1/120)	0.0(0/120)	20.1	12.2
		ESFT	26.4	0.8(1/120)	0.8(1/120)	18.1	11.5
		DenseMixer	31.6	1.7(2/120)	1.7(2/120)	26.7	15.4
		ExpertCondenser (Ours)	34.6	4.1(5/120)	2.5(3/120)	28.6	**17.4**
DeepSeek-Coder-V2-Lite	16B	SFT	34.2	2.5(3/120)	2.5(3/120)	64.6	26.0
		ESFT	32.2	2.5(3/120)	2.5(3/120)	63.0	25.0
		DenseMixer	34.8	2.5(3/120)	2.5(3/120)	64.8	26.1
		ExpertCondenser (Ours)	38.7	3.3(4/120)	2.5(3/120)	66.8	**27.8**
Qwen3	30B	SFT	58.6	63.3(76/120)	48.3(58/120)	94.8	66.3
		ESFT	52.7	61.7(74/120)	44.2(53/120)	92.0	62.7
		DenseMixer	61.0	65.8(79/120)	46.7(56/120)	95.8	67.3
		ExpertCondenser (Ours)	65.8	68.3(82/120)	50.0(60/120)	96.8	**70.2**

SFT (AWS Samples, 2024; Hugging Face, 2024).

DenseMixer. [3] DenseMixer instead addresses the non-differentiability of Top-k routing by adopting a straight-through estimator (STE). In this view, the backward pass ignores the hard selection and treats TopK as the identity map: $\frac{\partial \, \mathrm{TopK}(w_{1,t},\ldots,w_{n,t})_i}{\partial w_{j,t}} \approx \delta_{i,j}$. As a result, gradients flow to *all* experts' routing weights, not just those in S_t: $\nabla_\theta \mathcal{L} \approx \sum_{i=1}^{n} \left(\frac{\partial \mathcal{L}}{\partial y_t} \cdot v_i \right) \frac{\partial w_{i,t}}{\partial \theta}$. However, both ESFT (Wang et al., 2024b) and DenseMixer (Yao et al., 2025) still suffer from the additional noise introduced by auxiliary balancing losses. Moreover, they overlook the fact that less frequently activated experts also encode indispensable knowledge and contribute significantly to overall model performance. ESFT explicitly reinforces the "super experts" that dominate activation, the less activated experts are frozen and treated as trivial experts.

5 Experimental Results

In this section, we investigate how ExpertCondenser compared to the state-of-the-art methods (Wang et al., 2024a; Yao et al., 2025) in terms of model performance and system efficiency.

Table 3: Accuracy comparison of OLMoE and Qwen1.5-MoE with various post-training methods on commonsense reasoning datasets.

Model	Model Size	Post-train method	BoolQ	PIQA	SIQA	HellaSwag	WinoGrande	ARC-e	ARC-c	OBQA	AVG
OLMoE	**7B**	SFT	62.5	65.8	62.8	70.7	71.4	78.4	63.7	70.6	68.2
		ESFT	63.5	63.0	58.9	64.7	62.8	74.8	63.8	63.4	64.4
		DenseMixer	62.8	68.7	65.3	71.6	73.5	81.3	63.5	71.3	69.8
		ExpertCondenser (Ours)	67.7	71.4	69.7	69.8	75.8	76.8	71.0	73.6	**72.0**
Qwen1.5-MoE	**14B**	SFT	68.8	84.7	74.5	76.8	75.6	84.6	72.8	76.4	76.8
		ESFT	69.7	85.3	75.4	78.2	74.2	84.0	71.8	75.0	76.7
		DenseMixer	70.8	85.7	74.6	75.8	78.9	82.6	74.8	77.8	77.6
		ExpertCondenser (Ours)	72.1	84.9	75.6	81.6	79.8	88.5	78.1	84.4	**80.6**

To answer this, in § 5.1 and § 5.2, we evaluate ExpertCondenser on mathematical and commonsense reasoning benchmarks in term of model performance. In § 5.3, we analyze the system efficiency of ExpertCondenser in terms of training system efficiency. We evaluate **ExpertCondenser**, our proposed method, against two state-of-the-art MoE post-training approaches: ESFT (Wang et al., 2024b) and DenseMixer (Yao et al., 2025). In § 5.4, we present ablation studies showing that all components of our approach contribute to the overall performance improvement. Appendix C further provides the detailed experimental settings. We provide additional analysis of cross-domain generalization in Appendix K and training stability in Appendix R.

5.1 Math Reasoning

We report performance on two math reasoning datasets (*Math-7K* [4] and *Math-14K*) in Table 1. Across all benchmarks (GSM8K, SingleEq, SVAMP, MultiArith, AddSub, AQuA, and MAWPS), ExpertCondenser consistently outperforms both ESFT and DenseMixer on DeepSeek-V2-Lite, Qwen1.5-MoE, and OLMoE. On *Math-7K*, it improves the average accuracy of Qwen1.5-MoE by 3.3%, DeepSeek-V2-Lite by 2.7%, and OLMoE by 2.6%, and GPT-OSS by 1.0%. On *Math-14K*, ExpertCondenser also boosts performance.

Table 2 further demonstrates that on most SoTA math reasoning benchmarks, ExpertCondenser outperforms baseline methods across Qwen3, DeepSeek-Coder-V2-Lite, and Qwen1.5 [5]. ExpertCondenser enhances accuracy of DenseMixer by 2.9%, 1.7%, and 2.0% on Qwen3, DeepSeek-Coder-V2-Lite, and Qwen1.5 respectively.

5.2 Other Datasets

To ensure that our findings above are generalizable, we further examine the performance of ExpertCondenser under the common sense reasoning dataset `CommonSense-15K`, including downstream test datasets, `BoolQ`, `PIQA`, `SIQA`, `HellaSwag`, `ARC-e`, `ARC-c`, and `OBQA`.

Table 3 reports the performance of DenseMixer, ESFT, and ExpertCondenser on the `CommenSense` dataset. We can observe that ExpertCondenser surpasses the DenseMixer by 2.2% on post-trained OLMoE. On post-trained Qwen1.5-MoE, ExpertCondenser surpasses the best performance of DenseMixer and ESFT by 3.0% and 3.9%, respectively.

5.3 System Efficiency under Parameter Offloading and Expert Parallelism

Training large-scale MoE models under memory constraints often relies on parameter offloading (e.g., ZeRO-Offload (Ren et al., 2021)), where expert weights are dynamically transferred between CPU and GPU. In such settings, efficiency is sensitive to expert activation patterns, as frequent parameter movement can become a major bottleneck.

Although our primary goal is to improve post-training effectiveness, ExpertCondenser naturally exhibits a favorable systems property under offloading. Condenser experts are always selected across tokens, resulting in consistently high activation frequency. This allows their parameters to remain resident in GPU memory, reducing CPU–GPU transfers.

[4] More details about the `Math-7K` dataset can be found in Appendix N.

[5] OLMoE has a maximum context length of only 4096 tokens which is not suitable for *Stanford-S1*.

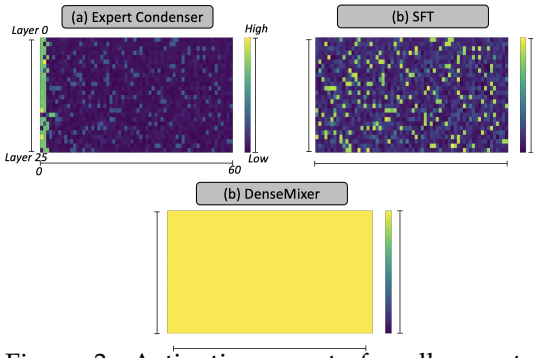


Figure 3: Activation counts for all experts across layers for:(a) ExpertCondenser, (b) SFT, and (c) DenseMixer.

Table 4: The experiments involved Post-training, ESFT, DenseMixer, and ExpertCondenser on 8 × H100 80GB GPUs using parameter offloading, with a batch size of 32. Communication between the GPU and CPU was facilitated via PCIe-G4.

DeepSeek-V2-Lite			
Methods	**Activate #Params%**	**Time/s**	**Speedup**
DenseMixer	16B	362.83	1×
SFT	2.4B	152.78	2.37×
ExpertCondenser	2.4B	126.24	2.87×

Table 5: Expert-parallel benchmark on 8×H100 with one Qwen1.5-MoE-A2.7B MoE layer.[6]

Method	**Step time (ms)**	**Dispatch (MB/step)**	**Peak mem/rank (GB)**
DenseMixer	67.00 (3.109×)	7516.19 (16.017×)	10.00
SFT	21.55 (1.000×)	469.26 (1.000×)	0.84
ExpertCondenser	**17.24** (0.800×)	**234.68** (0.500×)	**0.66**

Figure 3(a) shows that Condenser experts dominate activations across layers, and therefore never need to be swapped out of GPU memory. In contrast, SFT selects Top-k experts dynamically, leading to volatile activation patterns (Figure 3(b)) and increased offloading overhead. DenseMixer activates all experts (Figure 3(c)), removing sparsity benefits and requiring all experts to be loaded into GPU memory. We evaluate this effect under parameter offloading on DeepSeek-V2-Lite (Table 4). ExpertCondenser achieves a 2.87× speedup over DenseMixer and outperforms SFT.

Offloading is a single-node, memory-constrained regime. Large MoE models are more commonly served under *expert parallelism* (EP), where experts are sharded across devices and every routed expert computation crosses an all-to-all. To exam directly under EP determines whether the condenser experts design helps or hurts, we replicate condenser experts on every rank, exactly as Qwen1.5-MoE, DeepSeek-V2-Lite, OLMoE, and GPT-OSS already replicate their shared experts under EP. Replication costs r/n of the MoE parameters per device (3.3% at r=2 against the n=60 routed experts of Qwen1.5-MoE) and in exchange keeps r of each token's k expert computations on-device, so all-to-all volume falls by a factor of r/k. Table 5 confirms both effects on 8×H100. ExpertCondenser moves half the dispatch bytes of SFT (0.500×, matching the predicted $r/k = 2/4$ for Qwen1.5-MoE) and completes a step in 0.800× its time, while DenseMixer pays 16.0× the traffic. Peak per-rank memory also falls below SFT (0.66 vs. 0.84 GB), because the smaller all-to-all buffers outweigh the two replicated experts. The always-active pattern that helps under offloading is therefore neutral-to-favorable for EP communication as well.

We emphasize that this efficiency gain is a secondary effect of the proposed design. Nonetheless, it highlights that stable expert activation can improve training efficiency in both offloading- and EP-based settings.

5.4 Dissecting the ExpertCondenser Algorithm

Table 6 presents an ablation study of ExpertCondenser on DeepSeek-V2-Lite and Qwen1.5-MoE under the **math7k** benchmark. We evaluate progressively variants: (1) *aux-free*, removing auxiliary losses; (2) *aux-free+bias*, introducing bias-based routing; (3) *aux-free+bias+condenser experts*, enabling always-active gated experts; and (4) *with-aux+condenser experts*, which retains auxiliary loss while adding condenser experts.

Results from (1) to (3) show that each component contributes incrementally, while their combination yields the best performance. Results in (4) further reveal that incorporating

[6] 32,768 tokens per step, bf16, NCCL `all_to_all_single`, 5 warmup and 20 timed steps. Step time is the median forward + backward time max-reduced across ranks; dispatch bytes are summed over the 8 ranks. The r=2 condenser experts are replicated on every rank.

condenser experts on top of the auxiliary-loss setting still improves performance over the SFT baseline, demonstrating that the benefit of condenser experts.

A further control isolates the gating itself: replacing the router weight $g_j(x_t)$ on the condenser path with a fixed ungated shared expert removes the gain, so the benefit comes from the persistent path being token-dependent rather than merely always active (Appendix Q.3).

Table 6: Evaluation of variant post-trained models on downstream Math Reasoning tasks.

Dataset	Model	Post-train Type	GSM8k	SingleEq	SVAMP	MultiArith	AddSub	AQuA	mawps	AVG
math7k	DeepSeek-V2-Lite	aux-free+bias+condenser experts	59.4	92.5	69.1	91.5	79.5	36.1	83.6	73.1
		aux-free+bias	58.8	90.7	69.3	88.7	74.2	36.1	80.3	71.2
		aux-free	57.6	89.6	68.6	87.5	73.8	35.2	80.4	70.4
		with-aux+condenser experts	58.3	88.4	68.9	88.6	75.8	35.6	81.7	71.0
	Qwen1.5-MoE	aux-free+bias+condenser experts	57.2	74.6	55.7	86.0	61.8	33.1	75.6	63.4
		aux-free+bias	48.2	76.6	52.7	80.3	59.2	26.8	71.8	59.4
		aux-free	47.2	74.0	51.8	82.0	58.7	30.3	71.8	59.0
		with-aux+condenser experts	49.5	76.8	53.6	82.7	60.1	30.9	72.6	60.9

6 Further Analysis

6.1 Expert Routing and Output Analysis

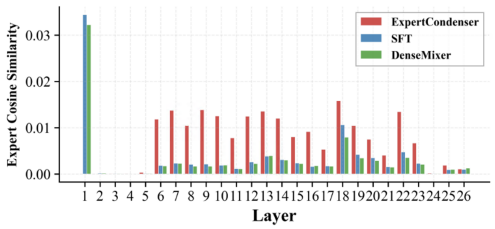


Figure 4: Weighted MoE Output Divergence.

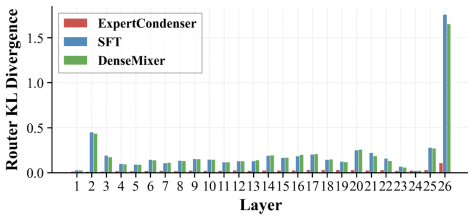


Figure 5: Router KL Divergence.

To understand how different methods adapt MoE models, we analyze two metrics: (1) **Router KL Divergence** in Fig. 4, which measures shifts in routing distributions relative to the base model, and (2) **Weighted MoE Output Divergence** in Fig. 5, which measures changes in layer outputs under each model's routing weights.

As shown in Fig.5, both SFT and DenseMixer significantly alter routing distributions, indicating that adaptation is partly achieved by rerouting tokens to different experts. In contrast, ExpertCondenser exhibits minimal router divergence, preserving the base model's routing structure. Meanwhile, Fig.4 reveals that ExpertCondenser induces more substantial changes in expert outputs than other methods. Together, these results suggest that ExpertCondenser adapts MoE models by directly transforming expert parameters rather than disrupting fragile routing behavior, supporting our motivation in § 3.2. The forced expert mechanism and learned biases enable this targeted adaptation, maintaining training-inference alignment while concentrating representational changes within the experts themselves. Preserving the base router also shows up downstream: after fine-tuning on general instruction data that contains no mathematical supervision, ExpertCondenser retains more of the base model's math ability than SFT (Appendix K).

6.2 Gradient Norm Analysis

To empirically validate the gradient starvation effect and the role of Condenser Experts, we analyze the gradient norms of expert FFN parameters during training. Specifically, for each expert, we measure the ℓ_2 norm of its parameter gradients and report the average gradient norm across (i) Condenser Experts and (ii) standard routed experts at each training step.

Figure 6 shows the evolution of gradient norms over training. We observe that routed experts exhibit lower gradient magnitudes, consistent with their limited activation probability. In contrast, condenser experts maintain consistently higher gradient norms throughout training. These results provide empirical support for our theory in §3.1, suggesting that

condenser experts effectively mitigate gradient starvation and enable sustained learning across training, and persistent gradient flow is crucial to retain task-relevant information.

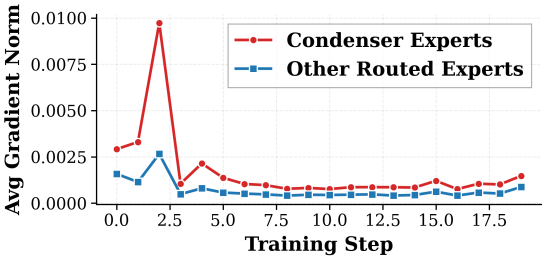


Figure 6: Average FFN gradient norm per expert during training.

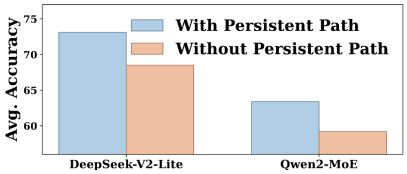


Figure 7: Persistent path ablation. "With" uses always-active gated condenser experts; "w.o." treats them as routed experts (i.e., not always selected) during inference.

6.3 Condenser Experts Analysis

We investigate the following research questions in this subsection: **Do condenser experts facilitate information consolidation from long-tailed experts?** To answer this, we examine two empirical signatures: Persistent path ablation in this subsection and expert correlation analysis in Appendix M. **For persistent path ablation.** We test the consolidation hypothesis in § 3.1 via a causal intervention on the persistent path. After training, we compare standard inference—where condenser experts are always selected—with a variant in which they are treated as standard routed experts. This preserves model capacity, parameters, and activation budget, while removing only the persistent aggregation pathway.

If condenser experts do not accumulate task-relevant information, this change should have minimal effect; otherwise, performance should degrade. As shown in Fig. 7, we observe consistent drops on Math7K for both Qwen1.5-MoE and DeepSeek-V2-Lite, indicating that condenser experts indeed consolidate task-relevant information. The observation further provides empirical support for our theoretical analysis in § 3.1.

7 Conclusion

Our empirical findings reveal that even rarely activated experts encode indispensable knowledge, and pruning them directly leads to substantial performance degradation. Motivated by this observation, we proposed ExpertCondenser, a post-training framework that combines bias-driven sparsification with consistently active gated experts. This design alleviates gradient starvation, narrows the train–inference routing gap, and provides a persistent pathway that helps preserve and consolidate information. Across math and commonsense reasoning benchmarks, ExpertCondenser consistently outperforms existing post-training methods such as ESFT and DenseMixer.

Acknowledgments

We extend our gratitude to Yu (Ivy) Yang for her helpful suggestions on manuscript preparation and research presentation.

This research was conducted using the ORCHARD high-performance computing cluster operated by the Office of Research Computing and Data at Carnegie Mellon University. We gratefully acknowledge Carnegie Mellon University for providing access to these shared computing resources.

This research also used the Bridges-2 system at the Pittsburgh Supercomputing Center and the Delta advanced computing system. These resources are supported by the National Science Foundation under Grants No. 2138259, 2138286, 2138307, 2137603, and 2138296.

References

AWS Samples. Esft patch for qwen2 mixture-of-experts models. https://github.com/aws-samples/sample-ESFT/blob/main/model_patch/patch_qwen2_moe.py, 2024. GitHub repository, accessed March 2026.

Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or propagating gradients through stochastic neurons for conditional computation, 2013. URL https://arxiv.org/abs/1308.3432.

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

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.

Sylvain Gugger, Lysandre Debut, Thomas Wolf, Philipp Schmid, Zachary Mueller, Sourab Mangrulkar, Marc Sun, and Benjamin Bossan. Accelerate: Training and inference at scale made simple, efficient and adaptable. https://github.com/huggingface/accelerate, 2022.

Nathan Habib, Clémentine Fourrier, Hynek Kydlíček, Thomas Wolf, and Lewis Tunstall. Lighteval: A lightweight framework for llm evaluation, 2023. URL https://github.com/huggingface/lighteval.

Haoze He, Juncheng Billy Li, Xuan Jiang, and Heather Miller. Sparse matrix in large language model fine-tuning. *arXiv preprint arXiv:2405.15525*, 2024.

Mohammad Javad Hosseini, Hannaneh Hajishirzi, Oren Etzioni, and Nate Kushman. Learning to solve arithmetic word problems with verb categorization. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 523–533, 2014.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models, 2021. URL https://arxiv.org/abs/2106.09685.

Zhiqiang Hu, Lei Wang, Yihuai Lan, Wanyu Xu, Ee-Peng Lim, Lidong Bing, Xing Xu, Soujanya Poria, and Roy Ka-Wei Lee. Llm-adapters: An adapter family for parameter-efficient fine-tuning of large language models. *arXiv preprint arXiv:2304.01933*, 2023.

Hugging Face. Sft trainer implementation in trl. https://github.com/huggingface/trl/blob/main/trl/trainer/sft_trainer.py, 2024. GitHub repository, accessed March 2026.

Rik Koncel-Kedziorski, Hannaneh Hajishirzi, Ashish Sabharwal, Oren Etzioni, and Siena Dumas Ang. Parsing algebraic word problems into equations. *Transactions of the Association for Computational Linguistics*, 3:585–597, 2015.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*, 2023.

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.

Wang Ling, Dani Yogatama, Chris Dyer, and Phil Blunsom. Program induction by rationale generation: Learning to solve and explain algebraic word problems. *arXiv preprint arXiv:1705.04146*, 2017.

Aixin Liu, Bei Feng, Bin Wang, Bingxuan Wang, Bo Liu, Chenggang Zhao, Chengqi Deng, et al. Deepseek-v2: A strong, economical, and efficient mixture-of-experts language model. *arXiv preprint arXiv:2405.04434*, 2024a.

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

Shih-Yang Liu, Chien-Yi Wang, Hongxu Yin, Pavlo Molchanov, Yu-Chiang Frank Wang, Kwang-Ting Cheng, and Min-Hung Chen. Dora: Weight-decomposed low-rank adaptation, 2024c. URL https://arxiv.org/abs/2402.09353.

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. *arXiv preprint arXiv:2402.14800*, 2024.

Niklas Muennighoff, Luca Soldaini, Dirk Groeneveld, Kyle Lo, Jacob Morrison, Sewon Min, Weijia Shi, Pete Walsh, Oyvind Tafjord, Nathan Lambert, Yuling Gu, Shane Arora, Akshita Bhagia, Dustin Schwenk, David Wadden, Alexander Wettig, Binyuan Hui, Tim Dettmers, Douwe Kiela, Ali Farhadi, Noah A. Smith, Pang Wei Koh, Amanpreet Singh, and Hannaneh Hajishirzi. Olmoe: Open mixture-of-experts language models, 2024. URL https://arxiv.org/abs/2409.02060.

Niklas Muennighoff, Zitong Yang, Weijia Shi, Xiang Lisa Li, Li Fei-Fei, Hannaneh Hajishirzi, Luke Zettlemoyer, Percy Liang, Emmanuel Candès, and Tatsunori Hashimoto. s1: Simple test-time scaling, 2025. URL https://arxiv.org/abs/2501.19393.

OpenAI. gpt-oss-120b & gpt-oss-20b model card, 2025. URL https://arxiv.org/abs/2508.10925.

Arkil Patel, Satwik Bhattamishra, and Navin Goyal. Are NLP models really able to solve simple math word problems? In Kristina Toutanova, Anna Rumshisky, Luke Zettlemoyer, Dilek Hakkani-Tur, Iz Beltagy, Steven Bethard, Ryan Cotterell, Tanmoy Chakraborty, and Yichao Zhou (eds.), *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 2080–2094, Online, June 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.naacl-main.168. URL https://aclanthology.org/2021.naacl-main.168.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. GPQA: A graduate-level google-proof q&a benchmark. In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=Ti67584b98.

Jie Ren, Samyam Rajbhandari, Reza Yazdani Aminabadi, Olatunji Ruwase, Shuangyang Yang, Minjia Zhang, Dong Li, and Yuxiong He. Zero-offload: Democratizing billion-scale model training. *ArXiv*, abs/2101.06840, 2021. URL https://arxiv.org/abs/2101.06840.

Subhro Roy and Dan Roth. Solving general arithmetic word problems. *arXiv preprint arXiv:1608.01413*, 2016.

Zunhai Su, Qingyuan Li, Hao Zhang, YuLei Qian, Yuchen Xie, and Kehong Yuan. Unveiling super experts in mixture-of-experts large language models. *arXiv preprint arXiv:2507.23279*, 2025.

Leandro von Werra, Younes Belkada, Lewis Tunstall, Edward Beeching, Tristan Thrush, Nathan Lambert, Shengyi Huang, Kashif Rasul, and Quentin Gallouédec. Trl: Transformer reinforcement learning. https://github.com/huggingface/trl, 2020.

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

Zihan Wang, Deli Chen, Damai Dai, Runxin Xu, Zhuoshu Li, and Yu Wu. Let the expert stick to his last: Expert-specialized fine-tuning for sparse architectural large language models. *arXiv preprint arXiv:2407.01906*, 2024b.

Zhenda Xie, Yixuan Wei, Huanqi Cao, Chenggang Zhao, Chengqi Deng, Jiashi Li, Damai Dai, Huazuo Gao, Jiang Chang, Kuai Yu, et al. mhc: Manifold-constrained hyper-connections. *arXiv preprint arXiv:2512.24880*, 2025.

An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, Guanting Dong, Haoran Wei, Huan Lin, Jialong Tang, Jialin Wang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Ma, Jin Xu, Jingren Zhou, Jinze Bai, Jinzheng He, Junyang Lin, Kai Dang, Keming Lu, Keqin Chen, Kexin Yang, Mei Li, Mingfeng Xue, Na Ni, Pei Zhang, Peng Wang, Ru Peng, Rui Men, Ruize Gao, Runji Lin, Shijie Wang, Shuai Bai, Sinan Tan, Tianhang Zhu, Tianhao Li, Tianyu Liu, Wenbin Ge, Xiaodong Deng, Xiaohuan Zhou, Xingzhang Ren, Xinyu Zhang, Xipin Wei, Xuancheng Ren, Yang Fan, Yang Yao, Yichang Zhang, Yu Wan, Yunfei Chu, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zhihao Fan. Qwen2 technical report. *arXiv preprint arXiv:2407.10671*, 2024.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Feng Yao, Junxia Cui, Ruohan Zhang, Liyuan Liu, Shibo Hao, Li Zhang, Chengyu Dong, Shuohang Wang, Yelong Shen, Jianfeng Gao, and Jingbo Shang. Densemixer: Improving moe post-training with precise router gradient, June 2025. URL https://fengyao.notion.site/moe-posttraining.

Mengxia Yu, De Wang, Qi Shan, Colorado Reed, and Alvin Wan. The super weight in large language models, 2025. URL https://arxiv.org/abs/2411.07191.

Defa Zhu, Hongzhi Huang, Zihao Huang, Yutao Zeng, Yunyao Mao, Banggu Wu, Qiyang Min, and Xun Zhou. Hyper-connections. *arXiv preprint arXiv:2409.19606*, 2024.

A Notation

For clarity, we summarize the main notations used throughout the theoretical sections.

- h_i Raw output vector of expert i.
- v_i Decomposed expert output: $v_i = \rho_i e_i$.
- ρ_i Magnitude (norm) of expert i's output.
- e_i Normalized direction vector of expert i, $\|e_i\| = 1$.
- y Dense output: $y = \sum_{i=1}^{n} v_i$.
- $\hat{y}$ Sparse approximation using only k experts.
- S Index set of selected experts, $|S| = k$.
- λ_i Binary indicator of expert selection.
- n, k Total number of experts, and number of selected experts.
- $v_{i,j,k}^{(l)}$ Output of expert i at layer l for token k in x_j.
- $g_{i,j,k}^{(l)}$ Gate score assigned to expert i at layer l for token k.
- $g_{i,t}$ Indicator if expert i is selected for token t (shorthand).
- $s_{i,t}$ Gating score (logit) of expert i for token t before normalization.
- $w_{i,t}$ Softmax-normalized routing weight assigned to expert i for token t, and softmax-normalized routing weight from the *unbiased* scores $s_{i,t}$.
- L_j Length (number of tokens) of sample x_j.
- $\mathcal{D}_s, N_s$ Subset of training data and its size.
- x_j jth token for training
- $s_i^{(l)}$ Magnitude-based expert score (ES-Mag) at layer l.
- $r_i^{(l)}$ Activation-ratio expert score (ES-Act) at layer l.
- K Number of experts selected per token.
- $S^{(l)}$ Set of selected experts at layer l.
- $\mathcal{L}_{\text{Balance}}$ Auxiliary loss for balancing expert utilization.
- f_i Normalized fraction of tokens routed to expert i.
- P_i Average routing weight assigned to expert i across tokens.
- α Hyperparameter controlling auxiliary loss strength.
- T Sequence length (number of tokens).
- θ Router parameters used to compute gating scores.
- $\mathcal{L}$ Generic training loss depending on model output y.
- b_i Expert-wise bias used only for selection to improve load balance.
- $\tilde{s}_{i,t}$ Biased gating score for expert i on token t: $\tilde{s}_{i,t} = s_{i,t} + b_i$.
- S_t Top-K selection set for token t obtained from $\{\tilde{s}_{j,t}\}_{j=1}^{n}$.
- γ Bias update speed.

B Learnable Persistent Mixing Principle

Modern large-scale architectures increasingly rely on multi-path computation. Similar to Hyper-Connection, the designing of Mixture-of-Experts (MoE) share the same philosophy: increase MLP layer outputs' topological by composing multiple parallel experts or computational paths. However, beyond structural similarity, HC reveals a deeper principle which we named **Learnable Persistent Mixing(LPM): persistent computational paths should be governed by learnable mixing operators rather than fixed injection rules**. In this appendix section, we formalize this principle and show how it theoretically motivates our Condenser expert design.

Learnable Persistent Mixing Consider a collection of n parallel candidate transformations $\{f_i(\cdot)\}_{i=1}^{n}$, each mapping $\mathbb{R}^d \to \mathbb{R}^d$. A general multi-path layer can be written as

$$\mathbf{h}'(\mathbf{x}) = \sum_{i=1}^{n} m_i(\mathbf{x})\, f_i(\mathbf{x}), \qquad m(\mathbf{x}) \in \mathcal{M} \subseteq \mathbb{R}^n, \tag{7}$$

where $m(\mathbf{x})$ defines how information flows across parallel paths. This decomposition separates computation (the f_i) from connectivity (the mixing coefficients m_i). Different architectures correspond to different parameterizations and constraints on the admissible mixing set $\mathcal{M}$.

Hyper-Connection Serve as an Example of LPM Hyper-Connections (Zhu et al., 2024; Xie et al., 2025) explicitly parameterize connectivity across parallel residual streams. Let $\mathbf{x}_\ell \in \mathbb{R}^{n\times d}$ denote the expanded residual representation at layer ℓ. The HC update can be written in stream-wise form as

$$\mathbf{x}_{\ell+1,j} = \sum_{i=1}^{n} H^{\text{res}}_{\ell,ji}\, \mathbf{x}_{\ell,i} + H^{\text{post}}_{\ell,j} F_\ell\left(\sum_{i=1}^{n} H^{\text{pre}}_{\ell,i}\, \mathbf{x}_{\ell,i}\right), \tag{8}$$

where $H^{\text{res}}_\ell \in \mathbb{R}^{n\times n}$, $H^{\text{pre}}_\ell \in \mathbb{R}^{1\times n}$, and $H^{\text{post}}_\ell \in \mathbb{R}^{1\times n}$ are learnable parameters. Equation (8) reveals that every persistent path in HC is modulated by learnable coefficients. Residual propagation is controlled by H^{res}_ℓ, rather than being a fixed identity map. The transformed branch is read and written via H^{pre}_ℓ and H^{post}_ℓ. This full parameterization of connectivity is central to HC's expressive power. **Hyper-Connections provide a useful lens: persistent paths are most expressive when they remain learnably mixed rather than fixed.**

Condenser Experts Instead of Static Shared Experts Traditional shared component

$$h_{\text{static}}(x) = \sum_{i=1}^{\hat{n}} E_i(x)$$

is injected without any mixing weight. It is always active and contributes with fixed unit weight. Motivated by the Learnable Persistent Mixing (LPM) principle revealed by Hyper-Connections, we argue that persistent shared computation in MoE should also be governed by learnable mixing

$$h_{cond} = \sum_{j\in J} g_j(x_t)\, E_j(x_t)$$

Condenser experts serve as a special case of persistent computation paths, where a subset of experts is always active across tokens. Unlike static residual pathways, these experts remain input-gated, aligning with LPM and prior observations (Xie et al., 2025; Zhu et al., 2024) that persistent paths benefit from adaptive mixing. This interpretation complements our primary motivation of preserving long-tailed expert knowledge.

C Experimental Settings

Model Architecture and Dataset: In our experimental setup, we use open-weight GPT-OSS-20B (OpenAI, 2025), Deepseek-V2-Lite (Liu et al., 2024a), Deepseek-Coder-V2-Lite-Instruct (Liu et al., 2024a), OLMoE-7B-01-25 (Muennighoff et al., 2024), Qwen1.5-MoE-A2.7B (Yang et al., 2024), and Qwen3-30B-A3B (Yang et al., 2025) to conduct experiments.

In Section§ 2, We study the MoE expert scaling law using Deepseek-Coder-V2-Lite-Instruct base model and GPT-OSS-20B base model, and test the evaluation performance on `MultiArith`, `GSM_8K` (Cobbe et al., 2021), `AddSub`, `AQuA`, `SingleEq`, `SVAMP`, and `mawps`.

In Subsection § 5.1, we evaluate MoE post-training algorithms on math reasoning domains. We fine-tune on the `Math7K` and `Math14K` dataset using the DeepSeek-V2-Lite, Qwen1.5-MoE-A2.7B, OLMoE-7B-01-25, and GPT-OSS-20B and test the evaluation performance on downstream testsets `MultiArith`, `GSM_8K` (Cobbe et al., 2021), `AddSub`, `AQuA`, `SingleEq`, `SVAMP`, and `mawps`. Then, we post-train Qwen1.5-MoE-A2.7B, DeepSeek-Coder-V2-Lite-Instruct, and Qwen3-30B-A3B on `Stanford-S1` dataset (Muennighoff et al., 2025) and test the evaluation performance on downstream SOTA math reasoning benchmarks `AIME2025`, `AIME2024`, `GPQA-Diamond` (Rein et al., 2024), and `MATH-500`.

In Subsection § 5.2, we turn to commonsense reasoning. Following (Hu et al., 2023; He et al., 2024; Liu et al., 2024c), we merge the training sets of eight tasks into `commonsense_15k` and evaluate on their individual test sets: `BoolQ`, `PIQA`, `SIQA`, `HellaSwag`, `ARC-e`, `ARC-c`, and `OBQA`. Results are reported as accuracy, with an averaged score summarizing overall effectiveness. Across all datasets—`Stanford-S1K`, `Commonsense`, `Math7K`, and `Math14K`—our setup emphasizes the generalization ability of LLMs across diverse sub-tasks.

Training Framework and Hyper-parameters: We used the `huggingface-trl` (von Werra et al., 2020) library with zero-2 or zero-3 (Ren et al., 2021) for fine-tuning and `vllm` (Kwon et al., 2023), `lighteval` (Habib et al., 2023), and `accelerate` (Gugger et al., 2022) library for inference evaluation. Both training and evaluation are using dtype BF16.

MoE Post-train Baselines: For state-of-the-art (SOTA) MoE post-training baselines, we choose to include ESFT (Hu et al., 2021) and DenseMixer (Yao et al., 2025). We re-implemented ESFT and DenseMixer following the setups in (Yao et al., 2025).

Computational Resources: We conduct our experiments and implement SOTA baselines of ESFT and DenseMixer Yao et al. (2025) to post-train with 8 NVIDIA H100_80GB GPUs. Communication between the CPU and GPU is facilitated via PCIe-G4 and communication between GPUs is facilitated via Nvlink-3.

Baseline Reimplementation and Fair Comparison. To ensure a fair comparison across methods, we adopt a unified training protocol. Some baseline results are not directly available in prior work under our exact model and dataset configurations. Therefore, we re-implement and fine-tune all baselines within a unified training framework. All methods are trained with identical learning rates, batch sizes, optimizers, and number of epochs on the same model. For method-specific components (e.g., Number of experts trained in ESFT), we follow the configurations recommended in their original works. All models are initialized from the same pretrained checkpoints and trained on identical datasets and splits. We also repeat experiments with multiple random seeds and observe consistent performance gains of ExpertCondenser, with improvements exceeding variance across runs.

As an example, for the **Math7K** dataset with **DeepSeek-V2-Lite**, the following configuration is used consistently across all methods:

Parameter	**Value**
Data Type	bf16
Optimizer	AdamW
Learning Rate	1×10^{-5}
Batch Size	32
Sequence Length	4096
Seed	1234
Warmup Ratio	0.1
LR Scheduler	cosine_with_min_lr
Attention Implementation	flash_attention_2
ZeRO Stage	3

Table 7: Example training configuration used across all methods for Math7K with DeepSeek-V2-Lite.

All baselines and our method are trained with the above configuration. More hyper-parameters settings can also be found in our open-sourced repository.

D Table Results for Experts Scaling Laws

Tables 8 and 9 report detailed results of our scaling law experiments on two representative MoE models: GPT-OSS-20B and DeepSeek-Coder-V2-Lite. We evaluate performance under three pruning strategies introduced in Section §2:

(1) *Small Dense Conversion*, where the number of experts is reduced from n to n' and all surviving experts are activated ($k = n'$);

(2) *Inference Reduction*, where the total number of experts is fixed ($n' = n$) but the activation budget is reduced from k to $k' < k$; and

(3) *Small MoE Conversion*, where both the total number of experts is reduced ($n' < n$) and the activation budget is kept sparse ($k < n'$).

Table 8: Evaluating base GPT-OSS-20B model Zero-Shot Results on downstream Math Reasoning dataset, includes SingleEQ, MultiArith, AddSub, GSM8K, SVAMP, and AQuA.

Model	Strategies	Remain Experts (n')	Activate Experts (k')	GSM8k	SingleEq	SVAMP	MultiArith	AddSub	AQuA	mawps	AVG
GPT-OSS-20B	Base Model	**32**	**4**	78.0	84.6	84.1	92.0	81.0	33.9	92.0	77.9
	Small Dense	**8**	**8**	2.3	1.7	1.8	2.7	1.8	16.7	2.3	4.2
		12	**12**	4.8	17.8	14.6	18.4	18.6	22.8	17.3	16.3
		18	**18**	48.6	39.6	47.8	52.6	63.7	23.8	47.8	46.3
		24	**24**	56.9	58.2	63.5	68.5	67.5	24.6	58.7	56.8
	Inference Reduce	**32**	**1**	5.5	19.7	13.6	12.8	19.5	23.2	18.1	16.1
		32	**2**	70.8	74.6	76.4	88.2	75.4	29.9	76.9	70.3
		32	**3**	75.1	83.9	84.3	93.3	81.3	33.1	80.7	76.0
	Small MoE	**8**	**4**	1.6	2.2	3.2	3.2	2.0	12.6	2.5	3.9
		12	**4**	6.4	18.7	21.8	13.0	18.2	21.7	16.0	16.5
		16	**4**	20.2	43.7	48.1	43.7	41.0	24.0	42.4	37.6
		20	**4**	32.8	56.1	59.9	58.2	52.9	25.6	55.5	48.7
		24	**4**	49.6	72.4	74.5	83.3	72.4	29.1	70.2	64.5

Table 9: Evaluating base DeepSeek-Coder-V2-Lite-Instruct model Zero-Shot Results on downstream Math Reasoning dataset, includes SingleEQ, MultiArith, AddSub, GSM8K, SVAMP, and AQuA.

Model	Strategies	Remain Experts (n')	Activate Experts (k')	GSM8k	SingleEq	SVAMP	MultiArith	AddSub	AQuA	mawps	AVG
DeepSeek-Coder-V2-Lite	Base Model	**64**	**6**	82.6	95.1	83.6	94.3	89.6	26.4	86.6	79.7
	Small Dense	**6**	**6**	1.4	1.0	1.7	2.8	1.8	22.0	2.1	4.7
		12	**12**	1.7	11.0	4.5	7.2	13.2	24.0	9.7	10.2
		16	**16**	11.8	38.2	22.6	32.0	31.1	17.7	31.9	26.5
		20	**20**	26.0	63.2	46.0	67.7	56.5	17.7	58.4	47.9
		24	**24**	36.9	75.2	58.4	76.0	70.1	20.1	66.4	57.6
		32	**32**	47.8	83.5	71.4	88.2	81.8	24.4	79.6	68.1
		48	**48**	48.6	82.7	72.4	87.6	82.6	24.7	80.4	68.4
	Inference Reduce	**64**	**1**	28.5	68.7	50.7	73.0	64.6	24.0	72.6	54.6
		64	**2**	49.2	86.6	70.5	92.8	79.2	24.8	78.2	68.8
		64	**3**	54.3	87.2	76.6	92.3	80.3	25.6	84.9	71.6
		64	**4**	53.7	87.0	76.1	94.7	83.5	26.4	83.2	72.1
		64	**5**	57.9	89.0	79.8	95.7	84.6	25.2	84.5	73.8
	Small MoE	**12**	**6**	0.7	0.4	1.5	0.8	0.0	16.9	1.3	3.1
		16	**6**	0.8	0.2	1.4	1.2	1.3	13.4	1.3	2.8
		24	**6**	11.3	45.9	33.5	41.8	46.6	17.7	43.3	34.3
		32	**6**	35.6	76.4	63.4	80.7	69.7	22.0	74.8	60.4
		48	**6**	49.0	85.4	75.2	93.2	80.8	21.7	80.7	69.4

E Selecting Top-k Experts

We define two top-k selection rules, selecting by magnitude score and selecting by activation ratio. Let $v_{i,j,k}^{(l)}$ be the output of expert i at layer l for token k in sample x_j, with gate score $g_{i,j,k}^{(l)}$. Each sample has length L_j, and we draw a subset $\mathcal{D}_s = \{x_j\}_{j=1}^{N_s}$ from the training set. We compute a per-expert relevance score and pick Top-k experts for routing or distillation.

Magnitude Score (ES-Mag). Estimate expert importance by average output magnitude:

$$s_i^{(l)} = \frac{1}{N_s} \sum_{j=1}^{N_s} \frac{1}{L_j} \sum_{k=1}^{L_j} \left\| v_{i,j,k}^{(l)} \right\|.$$

When only a scalar amplitude $\rho_{i,j,k}^{(l)}$ is available, we approximate $\|v_{i,j,k}^{(l)}\| \approx \rho_{i,j,k}^{(l)}$; this criterion favors experts with larger norm contributions (see Appendix I for justification).

Activation Ratio (ES-Act). Estimate importance by how often the expert is selected:

$$r_i^{(l)} = \frac{1}{N_s} \sum_{j=1}^{N_s} \frac{1}{L_j} \sum_{k=1}^{L_j} \frac{\mathbf{1}\left[g_{i,j,k}^{(l)} > 0\right]}{K},$$

where K is the number of experts selected per token. This captures routing preference and data alignment.

Selection. For layer l, choose

$$S^{(l)} = \mathrm{TopK}\big(\{s_i^{(l)}\}_i\big) \quad \text{or} \quad S^{(l)} = \mathrm{TopK}\big(\{r_i^{(l)}\}_i\big),$$

and select the variant by validation performance: *ES-Mag* emphasizes magnitude-dominant contribution, while *ES-Act* reflects gate-driven frequency. The ablation studies to investigate between Magnitude Score(ES-Mag) and Activation Ratio(ES-Act) can be found in Appendix O.

F Auxiliary Loss for Load Balancing

Uncontrolled routing strategies in Mixture-of-Experts (MoE) models often suffer from load imbalance. This manifests in two ways: (i) *routing collapse*, where only a small subset of experts are consistently selected, leading to undertraining of the remaining experts; and (ii) *computational imbalance*, where uneven routing across devices increases latency and reduces efficiency. To mitigate these issues, an auxiliary loss is commonly introduced in SOTA MoE models (Liu et al., 2024b; Wang et al., 2024b).

For a sequence of length T, the auxiliary loss is defined as

$$\mathcal{L}_{\text{Balance}} = \alpha \sum_{i=1}^{n} f_i P_i,$$

where α is a hyperparameter controlling the strength of the regularization. Here,

$$f_i = \frac{n}{KT} \sum_{t=1}^{T} \mathbf{1}[g_{i,t} > 0], \qquad P_i = \frac{1}{T} \sum_{t=1}^{T} s_{i,t}.$$

The term f_i measures the fraction of tokens routed to expert i, normalized by the total number of tokens T, experts n, and the per-token selection budget K. The term P_i is the average routing probability assigned to expert i, where $s_{i,t}$ denotes the gating score of expert i for token t. The loss encourages alignment between routing frequency (f_i) and gating probability (P_i), thereby preventing collapse and promoting balanced utilization of experts.

G Auxiliary-Loss-Free Load Balancing Strategy

To improve load balance without introducing an additional loss term, Deepseek-V3 (Liu et al., 2024b) adjust the *selection* rule by adding an expert-wise bias to the gating scores. Let $s_{i,t}$ be the (un-normalized, before Softmax) gating score for expert i on token t. We define a biased score

$$\tilde{s}_{i,t} \;=\; s_{i,t} + b_i,$$

where b_i is an expert-specific bias that is updated by a balancing controller (e.g., based on utilization statistics; see remark below). The Top-k *selection set* for token t is then

$$S_t \;=\; \mathrm{TopK}\big(\{\tilde{s}_{j,t}\}_{j=1}^{n},\, K\big), \qquad g_{i,t} \;=\; \mathbf{1}[\, i \in S_t \,].$$

Important distinction (selection vs. weighting). The bias b_i is *only* used to influence which experts enter S_t. It does *not* modify the routing weights used to combine expert outputs. Weights are obtained from the *unbiased* scores via softmax:

$$w_{i,t} \;=\; \frac{\exp(s_{i,t})}{\sum_{j=1}^{n} \exp(s_{j,t})},$$

and the token-level MoE output is

$$y_t \;=\; \sum_{i=1}^{n} g_{i,t}\, w_{i,t}\, v_i \;=\; \sum_{i \in S_t} w_{i,t}\, v_i, \quad \text{with} \quad v_i = \rho_i e_i.$$

Thus b_i affects *who is selected* but never changes the weights $w_{i,t}$ applied to the selected experts in the forward pass.

Algorithm (per token t).

1. Compute unbiased scores $\{s_{i,t}\}_{i=1}^{n}$ and weights $w_{i,t} = \mathrm{softmax}(s_{i,t})$.
2. Form biased scores $\tilde{s}_{i,t} = s_{i,t} + b_i$ and select $S_t = \mathrm{TopK}(\{\tilde{s}_{j,t}\}_{j=1}^{n}, K)$.
3. Set indicators $g_{i,t} = \mathbf{1}[i \in S_t]$ and compute $y_t = \sum_{i \in S_t} w_{i,t} v_i$.

Bias updating b_i. Any load-balancing controller can be used to update the biases; for example, one may adjust b_i as a function of the observed utilization f_i and target utilization K/n (e.g., with a moving-average estimator). In DeepSeek-V3, during training, they monitoring the expert load on the whole batch of each training step. At the end of each step, we will decrease the bias term by γ if its corresponding expert is overloaded, and increase it by γ if its corresponding expert is underloaded, where γ is a hyper-parameter called bias update speed. Through the dynamic adjustment, DeepSeek-V3 keeps balanced expert load during training, and achieves better performance than models that encourage load balance through pure auxiliary losses.

H Gradient Propagation Through Top-k Routing

Consider a Mixture-of-Experts layer where the router produces gating scores $\{s_{i,t}\}_{i=1}^{n}$ for token t. These scores are normalized via a softmax to obtain the routing weights

$$w_{i,t} \;=\; \frac{\exp(s_{i,t})}{\sum_{j=1}^{n}\exp(s_{j,t})}, \qquad i = 1, \ldots, n.$$

We then select the top-k experts according to these weights and form the token-level output

$$y_t \;=\; \sum_{i \in S_t} w_{i,t}\, v_i,$$

where $S_t = \mathrm{TopK}\big(\{w_{j,t}\}_{j=1}^{n}\big)$ is the index set of the k largest weights, and $v_i = \rho_i e_i$ is the contribution of expert i (with ρ_i the magnitude of h_i and e_i its normalized direction). The final loss for this token is $\mathcal{L} = \mathcal{L}(y_t)$.

Gradient with respect to router parameters. Back-propagation through this layer requires differentiating the loss with respect to the router parameters θ:

$$\nabla_\theta \mathcal{L} \;=\; \sum_{i=1}^{n} \left(\frac{\partial \mathcal{L}}{\partial y_t} \cdot v_i \right) \cdot \frac{\partial\, \mathrm{TopK}(w_{1,t}, \ldots, w_{n,t})_i}{\partial w_{i,t}} \cdot \frac{\partial w_{i,t}}{\partial \theta}.$$

The Jacobian term $\partial w_{i,t}/\partial\theta$ is determined by the softmax of the gating scores $s_{i,t}$, while the middle factor contains the (non-differentiable) Top-k selection.

Conventional approximation: SFT and ESFT A common approximation in MoE training treats the Top-k operation as if it were differentiable by passing gradients only through the selected experts. Formally, one replaces

$$\frac{\partial\, \mathrm{TopK}(w_{1,t}, \ldots, w_{n,t})_i}{\partial w_{j,t}} \;\approx\; \delta_{i,j}\, \mathbf{1}[i \in S_t],$$

where $\delta_{i,j}$ is the Kronecker delta. Under this approximation, the router gradient reduces to

$$\nabla_\theta \mathcal{L} \;\approx\; \sum_{i \in S_t} \left(\frac{\partial \mathcal{L}}{\partial y_t} \cdot v_i \right) \frac{\partial w_{i,t}}{\partial \theta}.$$

Thus only the experts chosen in S_t receive gradient updates through the gating mechanism.

Straight-through (STE) approximation: DenseMixer An alternative, more precise approximation—used in methods like DenseMixer—employs a straight-through estimator (STE). In the backward pass, the Top-k operation is treated as the identity map:

$$\frac{\partial\, \mathrm{TopK}(w_{1,t}, \ldots, w_{n,t})_i}{\partial w_{j,t}} \;\approx\; \delta_{i,j}.$$

This allows gradients to flow to all experts' routing weights, yielding

$$\nabla_\theta \mathcal{L} \;\approx\; \sum_{i=1}^{n} \left(\frac{\partial \mathcal{L}}{\partial y_t} \cdot v_i \right) \frac{\partial w_{i,t}}{\partial \theta}.$$

In this view, the forward pass still uses a hard Top-k selection, but the backward pass distributes gradients as though the selection were an identity operator.

Summary. The conventional method restricts gradient updates to the selected experts S_t, while the straight-through method propagates gradients to all experts by overriding the Top-k operation in the backward pass.

I Theoretical Supports for Top-K Selection

In a Mixture-of-Experts (MoE) architecture, each expert contributes to the overall output as

$$v_i = \rho_i e_i,$$

where ρ_i is the gating weight corresponds to the i-th expert, e_i is the expert output, and v_i is the output vector of the i-th expert. In a dense model, the final output is given by

$$y = \sum_{i=1}^{n} v_i = \sum_{i=1}^{n} \rho_i e_i.$$

In a sparse Mixture-of-Experts (MoE) model, we aim to reduce computation by selecting only a subset of experts. Thus, we wish to approximate y using

$$\hat{y} = \sum_{i \in S} v_i,$$

where S is a subset of indices with $|S| = k \ll n$. This objective can be formulated as the minimization problem

$$\min_{\lambda_1,\ldots,\lambda_n \in \{0,1\}} \left\| \sum_{i=1}^{n} v_i - \sum_{i=1}^{n} \lambda_i v_i \right\|^2 \quad \text{subject to} \quad \sum_{i=1}^{n} \lambda_i = k,$$

where $\lambda_i = 1$ indicates that expert i is selected, and $\lambda_i = 0$ indicates it is omitted. In many practical scenarios, especially when the normalized directions e_i are not strongly correlated, this minimization is well approximated by selecting the experts with the largest values of ρ_i. Intuitively, experts with large ρ_i contribute most significantly to the norm of y, so preserving these in the approximation yields a smaller error. We analyze why selecting experts with large ρ_i is a reasonable approximation in the following.

In a Mixture-of-Experts (MoE) architecture, each expert contributes to the overall output as

$$v_i = \rho_i e_i,$$

where ρ_i is the gating weight and e_i is the expert output. When analyzing why the top-k selection rule arises, it is instructive to consider two scenarios: one in which the vectors v_i are orthonormal (or nearly so) and another in which they have general correlations.

In this appendix, we show that in the non-orthonormal case, selecting the top-k experts with the largest ρ_i provides a close approximation to the full model output while substantially reducing computational cost. In the orthonormal case, this selection is provably optimal; in the general case, it serves as a widely used and effective heuristic.

The Orthonormal (or Weakly-Correlated) Case

Assume that the vectors $v_1, v_2, \ldots, v_n$ are strictly orthonormal, i.e.,

$$v_i^\top v_j = \begin{cases} 0, & \text{if } i \neq j, \\ \|v_i\|^2, & \text{if } i = j. \end{cases}$$

Then, the squared norm of the omitted portion,

$$\left\| \sum_{i=1}^{n} (1 - \lambda_i) v_i \right\|^2,$$

expands as

$$\left\| \sum_{i=1}^{n} (1 - \lambda_i) v_i \right\|^2 = \sum_{i=1}^{n} (1 - \lambda_i)^2 \|v_i\|^2,$$

and since $\lambda_i \in \{0, 1\}$, we have $(1 - \lambda_i)^2 = (1 - \lambda_i)$. Therefore, the objective becomes

$$\sum_{i=1}^{n} (1 - \lambda_i) \|v_i\|^2,$$

subject to $\sum_{i=1}^{n} \lambda_i = k$. To minimize this quantity, it is optimal to set $\lambda_i = 1$ for the k vectors with the largest norms $\|v_i\|^2$ and $\lambda_i = 0$ for the others. In the orthonormal case, this strategy is provably optimal.

Even if the vectors are only weakly correlated, the same principle generally holds: larger magnitudes imply a larger contribution to the overall sum, so omitting vectors with small $\|v_i\|$ results in a minor error, making the top-k selection by magnitude a robust heuristic.

The General (Non-Orthonormal) Case

When the vectors v_i have significant correlations, the cross terms do not vanish. In this case, the error term becomes

$$\left\| \sum_{i=1}^{n} (1 - \lambda_i) v_i \right\|^2 = \sum_{i=1}^{n} (1 - \lambda_i) \|v_i\|^2 + 2 \sum_{1 \le i < j \le n} (1 - \lambda_i)(1 - \lambda_j) v_i^\top v_j.$$

Here, the cross terms $v_i^\top v_j$ can affect the error significantly. In principle, finding the subset S that minimizes this expression exactly is an NP-hard combinatorial problem. However, in practice, one commonly uses the heuristic of selecting the top k experts based on the individual magnitudes $\|v_i\|$ (or a predicted magnitude ρ_i). This approach is effective because, in many settings, the largest magnitude vectors still dominate the overall contribution even when correlations are present. In scenarios where two high-magnitude vectors are strongly correlated, more sophisticated selection methods might improve the approximation, but the top-k rule remains a strong and computationally efficient baseline.

Conclusion: Whether the expert output vectors are orthonormal or generally correlated, the top-k selection rule emerges from the objective of preserving the dominant contributions to the sum while minimizing approximation error. In an MoE architecture, each expert's output $v_i = \rho_i e_i$ contributes to the overall sum. By selecting the k experts with the largest ρ_i, one can achieve a good approximation of the full model output with significantly reduced computational cost. In the orthonormal case, this method is exactly optimal, while in the general case it remains a widely-used and effective heuristic.

J Gradient Lower Bound for Condenser Experts

Proof. We consider a Mixture-of-Experts model of the form

$$h(x) = \sum_{i \in S(x)} g_i(x)\, E_i(x;\theta_i) + \sum_{k=1}^{\hat{n}} E_k(x),$$

with expected risk

$$\mathcal{L}(\theta) = \mathbb{E}_{(x,y)\sim\mathcal{D}}\big[\ell(h(x), y)\big].$$

Let

$$\delta(x) := \frac{\partial \ell}{\partial h(x)}$$

denote the backpropagated gradient at the MoE output.

Gradient identity. Since $j \in J \subseteq S(x)$ for all x, the model output depends on θ_j through the term

$$g_j(x)\, E_j(x;\theta_j).$$

By the chain rule,

$$\nabla_{\theta_j} h(x) = g_j(x)\, \frac{\partial E_j(x;\theta_j)}{\partial \theta_j}.$$

Therefore,

$$\nabla_{\theta_j}\mathcal{L} = \mathbb{E}_{(x,y)\sim\mathcal{D}}\Big[\delta(x)\, g_j(x)\, \frac{\partial E_j(x;\theta_j)}{\partial \theta_j}\Big].$$

Lower bound. We now derive the gradient magnitude lower bound. Consider the event $\{g_j(x) \geq \epsilon\}$. Then

$$\mathbb{E}\big[\|\nabla_{\theta_j}\mathcal{L}\|\big] = \mathbb{E}\Big[\big\|\delta(x)\, g_j(x)\, \frac{\partial E_j(x)}{\partial \theta_j}\big\|\Big].$$

Restricting to the event $\{g_j(x) \geq \epsilon\}$ gives

$$\geq \mathbb{E}\Big[g_j(x)\, \big\|\delta(x)\, \frac{\partial E_j(x)}{\partial \theta_j}\big\| \mathbf{1}_{\{g_j(x)\geq\epsilon\}}\Big].$$

Using $g_j(x) \geq \epsilon$ on this event,

$$\geq \epsilon\, \mathbb{P}\big(g_j(x) \geq \epsilon\big)\, \mathbb{E}\Big[\big\|\delta(x)\, \frac{\partial E_j(x)}{\partial \theta_j}\big\| \;\big|\; g_j(x) \geq \epsilon\Big].$$

By assumption, $\mathbb{P}(g_j(x) \geq \epsilon) \geq p$, which yields

$$\mathbb{E}\big[\|\nabla_{\theta_j}\mathcal{L}\|\big] \geq p\,\epsilon\, \mathbb{E}\Big[\big\|\delta(x)\, \frac{\partial E_j(x)}{\partial \theta_j}\big\| \;\big|\; g_j(x) \geq \epsilon\Big].$$

□

K Out-of-Domain Generalization

To further evaluate the generalization ability of ExpertCondenser beyond the target training domain, we conduct an additional out-of-domain experiment. In this setting, we fine-tune the OLMoE-7B model on a **commonsense reasoning dataset**, and evaluate the resulting models on **mathematical reasoning benchmarks**, including GSM8K, SingleEq, SVAMP, MultiArith, AddSub, AQuA, and MAWPS.

We note that, similar to prior work such as DenseMixer Yao et al. (2025), our method is primarily designed to improve post-training performance on a given target task, rather than to preserve broad multi-domain capabilities during supervised fine-tuning. Nevertheless, this experiment provides additional insight into how ExpertCondenser behaves under cross-domain evaluation.

Table 10 reports the results. ExpertCondenser achieves performance comparable to SFT and ESFT across all benchmarks, and consistently outperforms DenseMixer. While all fine-tuning methods exhibit degradation relative to the base model when evaluated on an unrelated domain, ExpertCondenser does not introduce additional degradation compared to standard SFT-based approaches. These results indicate that ExpertCondenser maintains similar cross-domain generalization behavior to existing post-training methods. Despite enforcing structured sparsity and persistent expert activation, the model retains sufficient flexibility to support out-of-domain evaluation without additional performance loss.

FT Dataset	Model	Post-train Type	GSM8K	SingleEq	SVAMP	MultiArith	AddSub	AQuA	MAWPS	AVG
Commonsense	OLMoE-7B	Base Model	16.1	23.6	17.7	9.2	21.3	22.8	13.9	**17.8**
		ExpertCondenser (Ours)	8.9	13.7	11.6	5.4	13.8	18.6	11.3	11.9
		SFT	6.8	14.6	12.5	5.7	13.4	17.8	8.6	11.3
		ESFT	11.2	15.6	13.7	4.7	12.4	16.7	7.9	11.7
		DenseMixer	2.3	5.9	6.1	3.7	6.1	17.6	2.9	6.4

Table 10: Out-of-domain evaluation: models fine-tuned on commonsense reasoning and evaluated on mathematical reasoning tasks. ExpertCondenser performs comparably to SFT and ESFT, and consistently outperforms DenseMixer.

Generalization after tuning on general instruction data. The experiment above tunes on one reasoning domain and evaluates on another. A stricter probe removes reasoning supervision from the training data entirely and asks how much of the base model's math ability survives fine-tuning. We fine-tune Qwen1.5-MoE-A2.7B on 10K general-instruction examples from `tatsu-lab/alpaca`, which contain no mathematical data, and evaluate on the same seven math benchmarks under a matched two-epoch budget for both methods. Table 11 reports the results. ExpertCondenser retains a 1.60-point higher average than SFT (33.06 vs. 31.46). The advantage is not uniform—SFT leads on GSM8K, MultiArith, and AQuA—but ExpertCondenser leads on the remaining four benchmarks and on the average. This is consistent with the router-KL measurement in § 6.1: ExpertCondenser departs least from the base model's routing distribution, so fine-tuning on an unrelated domain overwrites less of the base model's behavior.

FT Dataset	Model	Post-train Type	GSM8K	SingleEq	SVAMP	MultiArith	AddSub	AQuA	MAWPS	AVG
Alpaca-10K	Qwen1.5-MoE-A2.7B	SFT	**14.4**	49.6	33.2	**13.2**	38.0	**26.8**	45.0	31.46
		ExpertCondenser (Ours)	11.7	**55.9**	**37.8**	9.5	**46.3**	24.0	**46.2**	**33.06**

Table 11: Out-of-domain evaluation after fine-tuning on general instruction data: Qwen1.5-MoE-A2.7B tuned on 10K `tatsu-lab/alpaca` examples (no mathematical supervision) and evaluated on the seven math benchmarks, with a matched two-epoch budget.

L Expert Selection and Bias Design

Our routing design follows the auxiliary-loss-free load-balancing principle introduced in DeepSeek Wang et al. (2024a), where the bias term is applied only to routing decisions, and not to the computation of gating weights. This separation is critical for both stability and performance. Given routing logits s_i, we introduce a bias term b_i and perform Top-k selection using

$$\mathrm{TopK}(\{s_i + b_i\}),$$

while the gating weights $g_{i,t}$ are computed from the original logits s_i without bias. As stated in DeepSeek Wang et al. (2024a), the bias term is used solely to adjust routing behavior and is not included in the computation of expert output weights.

(1) Bias controls selection, not representation. The bias term b_i is designed to influence *which experts are selected*, rather than *how much each expert contributes*. By applying bias only before Top-k selection, we modify the routing pattern while preserving the original gating function for combining expert outputs.

If the bias were also added to the gating weights (e.g., inside the softmax or sigmoid), it would directly distort the magnitude of expert contributions. This would alter the representation distribution across layers, introduce additional gradients into expert outputs, and interfere with the learned semantics of the routing function.

(2) Preserving stable gating geometry. The router logits are typically computed as

$$s_i = G(x^\top e_i),$$

where e_i denotes expert embeddings. Injecting bias into the gating weights would modify the geometry of this scoring function, shifting expert centroids and distorting the distribution of routing probabilities.

DeepSeek Wang et al. (2024a) explicitly avoids this design, as modifying gating weights introduces undesired gradients and destabilizes training. Their formulation emphasizes that auxiliary-free balancing should control load without interfering with the learned gating distribution.

(3) Avoiding gradient interference. If the bias term were included in the gating weights, it would affect both selection and weighting simultaneously. In this case, bias updates would propagate through expert outputs, mixing routing control with model optimization. This coupling introduces additional gradient interference, which can destabilize training and degrade performance.

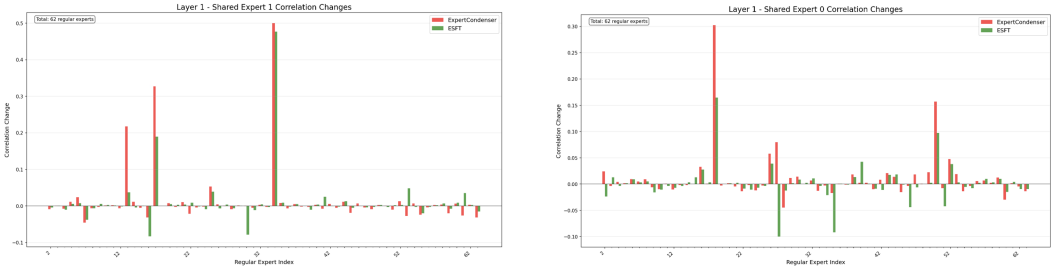
In contrast, our design isolates routing control (via b_i) from representation learning (via $g_{i,t}$), maintaining a clean separation between selection and computation.

DeepSeek Wang et al. (2024a) reports that incorporating bias directly into gating weights leads to unstable gate distributions and worse performance (see Table 4 in their paper). We adopt the same design principle in ExpertCondenser.

In addition, we observe that using $s_i + b_i$ for selection produces a substantially different expert ranking compared to using s_i alone, particularly in later layers where bias accumulation drives structured sparsity. This difference reflects the intended effect of bias-based routing: reshaping expert utilization without perturbing the learned gating function.

M Expert Correlation Analysis

To assess how fine-tuning alters dependencies between experts, we examine the similarity of their parameter updates. Specifically, we compute the Pearson correlation between the parameters of the condensed experts and those of other experts, comparing the fine-tuned model to the base model. This measure captures linear relationships in parameter changes, allowing us to track how post-training reshapes inter-expert dependencies. A formal definition of Pearson correlation is provided in Appendix M.1.



(a) Shared Expert 0 correlation changes.

(b) Shared Expert 1 correlation changes.

Figure 8: Correlation changes between the shared expert and regular experts at Layer 1. Each bar shows how the correlation between a regular expert and the shared expert changes after fine-tuning, compared to the base model. ExpertCondenser (red) causes concentrated shifts, markedly strengthening positive correlations while suppressing negative ones. In contrast, ESFT (green) produces smaller, more diffuse adjustments across experts. These results illustrate that ExpertCondenser induces larger and more concentrated correlation shifts, while ESFT exerts a milder influence.

Observation. Condenser experts consolidate the information, resulting in pronounced, concentrated shifts. Regularly routed experts are compared against these shared experts by computing Pearson correlations between their down-projection weight matrices before and after fine-tuning. Relative changes in these correlations—normalized by the base model—reflect how each method reshapes inter-expert dependencies. As shown in Fig. 8, both ExpertCondenser and ESFT increase correlations relative to the baseline, with ExpertCondenser driving a stronger global increase in correlations. Relative correlation increases average 0.005 more across all layers compared to ESFT. The earliest layer shows the most significant shifts, highlighting ExpertCondenser's central role in shaping foundational representations.

M.1 Definition of Correlation

Correlation on `down_proj` Weights. Consider an MoE layer with a set of experts

$$\mathcal{E} = \{1, \dots, E\},$$

partitioned into a *shared expert set* $\mathcal{E}_{\text{sh}}$ and a *regular expert set* $\mathcal{E}_{\text{reg}}$, with $\mathcal{E}_{\text{sh}} \cap \mathcal{E}_{\text{reg}} = \emptyset$, $\mathcal{E}_{\text{sh}} \cup \mathcal{E}_{\text{reg}} = \mathcal{E}$.

For expert $e \in \mathcal{E}$ under setting s, let

$$\mathbf{w}_{e,s}^{(l)} = \text{vec}\Big(W_{\text{down},s}^{(l,e)}\Big) \in \mathbb{R}^d,$$

and denote its sample mean by $\bar{w}_{e,s}^{(l)} = \frac{1}{d}\sum_{k=1}^{d}(\mathbf{w}_{e,s}^{(l)})_k$. Define the centered weight vector

$$\tilde{\mathbf{w}}_{e,s}^{(l)} = \mathbf{w}_{e,s}^{(l)} - \bar{w}_{e,s}^{(l)}\mathbf{1}.$$

The Pearson correlation between experts i and j in layer l under setting s is then

$$C_{ij,s}^{(l)} = \frac{\left\langle \tilde{\mathbf{w}}_{i,s}^{(l)}, \tilde{\mathbf{w}}_{j,s}^{(l)} \right\rangle}{\left\| \tilde{\mathbf{w}}_{i,s}^{(l)} \right\|_2 \left\| \tilde{\mathbf{w}}_{j,s}^{(l)} \right\|_2}.$$

We study three settings:

$$s = 1 : \text{Base}, \qquad s = 2 : \text{ESFT}, \qquad s = 3 : \text{ExpertCondenser}.$$

Percent Correlation Gain. For each shared expert $i \in \mathcal{E}_{\text{sh}}$ and each regular expert $j \in \mathcal{E}_{\text{reg}}$, the percent change in correlation relative to Base is defined as

$$\Delta_{ij,s}^{(l)}(\%) = 100\,\frac{\mathcal{C}_{ij,s}^{(l)} - \mathcal{C}_{ij,1}^{(l)}}{\left|\mathcal{C}_{ij,1}^{(l)}\right|}, \qquad s \in \{2,3\}.$$

This "relative effect size" formulation stabilizes interpretation when the Base correlation is negative or near zero.

Layer-Level Aggregation. The average correlation gain for setting s in layer l is

$$\bar{\Delta}_s^{(l)}(\%) = \frac{1}{|\mathcal{E}_{\text{sh}}| \cdot |\mathcal{E}_{\text{reg}}|} \sum_{i \in \mathcal{E}_{\text{sh}}} \sum_{j \in \mathcal{E}_{\text{reg}}} \Delta_{ij,s}^{(l)}(\%).$$

Model-Level Summary Across L Layers. Define the overall average and variability as

$$\bar{\Delta}_s(\%) = \frac{1}{L} \sum_{l=1}^{L} \bar{\Delta}_s^{(l)}(\%), \qquad \sigma_s = \text{Std}\left(\left\{ \bar{\Delta}_s^{(l)}(\%) \right\}_{l=1}^{L} \right).$$

These summarize how strongly fine-tuning reshapes global correlations between shared experts and regular experts across the entire model.

N Math7K Dataset

Math10K dataset can evaluate the effectiveness of LLMs on the arithmetic reasoning task. Math10K incorporate six subsets including GSM8k, SingleEq, SVAMP, MultiArith, AddSub, and AQuA.(1) the GSM8K (Cobbe et al., 2021) dataset consists of high quality linguistically diverse grade school math word problems created by human problem writers, (2) the SVAMP (Patel et al., 2021) benchmark consists of one-unknown arithmetic word problems for up-to-4 grade level students by making simple changes to a set of problems from another existing dataset, (3) the MultiArith (Roy & Roth, 2016) dataset of math word problems requiring multiple reasoning steps and operations, (4) the AddSub (Hosseini et al., 2014) dataset of addition and subtraction arithmetic word problems, (5) the AQuA (Ling et al., 2017) dataset of algebraic word problems with natural language rationales, and (6) the SingleEq (Koncel-Kedziorski et al., 2015) dataset of grade-school algebra word problems that map to single equations with varying length;

O ES-Act versus ES-Mag

In this Appendix, we conduct ablation studies to investigate between Magnitude Score(ES-Mag) and Activation Ratio(ES-Act), which one is the better metric to select preserving experts when we converting the original Mixture of Expert model into smaller models (either smaller dense models or smaller MoE models).

Table 12: Evaluation of post-trained models (Zero-Shot results) on downstream Math Reasoning datasets, including SingleEQ, MultiArith, AddSub, GSM8K, SVAMP, and AQuA.

Metric	Method	Remaining Experts	GSM8k	SingleEq	SVAMP	MultiArith	AddSub	AQuA	mawps	AVG
ES-Act	Smaller-Dense	6	1.4	1.0	1.7	2.8	1.8	22.0	2.1	4.7
		12	1.7	11.0	4.5	7.2	13.2	24.0	9.7	10.2
		16	11.8	38.2	22.6	32.0	31.1	17.7	31.9	26.5
		20	26.0	63.2	46.0	67.7	56.5	17.7	58.4	48.0
		24	36.9	75.2	58.4	76.0	70.1	20.1	66.4	57.6
		32	47.8	83.5	71.4	88.2	81.8	24.4	79.6	68.1
	Smaller-MoE	12	0.7	0.4	1.5	0.8	0.0	16.9	1.3	3.1
		16	0.8	0.2	1.4	1.2	1.3	13.4	1.3	2.8
		24	11.3	45.9	33.5	41.8	46.6	17.7	43.3	34.3
		32	35.6	76.4	63.4	80.7	69.7	22.0	74.8	60.4
		48	49.0	85.4	75.2	93.2	80.8	21.7	80.7	**69.4**
ES-Mag	Smaller-Dense	6	1.6	1.2	2.1	2.6	2.1	18.9	2.3	4.4
		12	1.8	11.8	5.2	6.8	13.6	23.8	9.4	10.3
		16	12.6	38.8	23.8	33.8	32.6	22.4	31.9	28.0
		20	25.4	64.8	45.6	68.8	56.3	18.2	59.8	48.4
		24	37.4	76.8	60.2	75.4	70.8	18.8	65.2	57.8
		32	48.2	82.8	72.2	87.8	82.2	24.2	83.2	68.6
	Smaller-MoE	12	0.7	0.4	1.7	0.6	0.0	18.6	1.6	3.4
		16	1.3	0.4	2.6	1.8	1.6	13.6	10.4	4.5
		24	10.8	44.7	32.8	42.3	44.8	21.3	44.8	34.5
		32	34.7	75.9	64.9	81.3	68.4	23.8	75.6	60.6
		48	47.6	86.7	75.8	92.7	81.7	23.9	81.3	70.0

P Expert Activation Probabilities in Base DeepSeek-V2-Lite

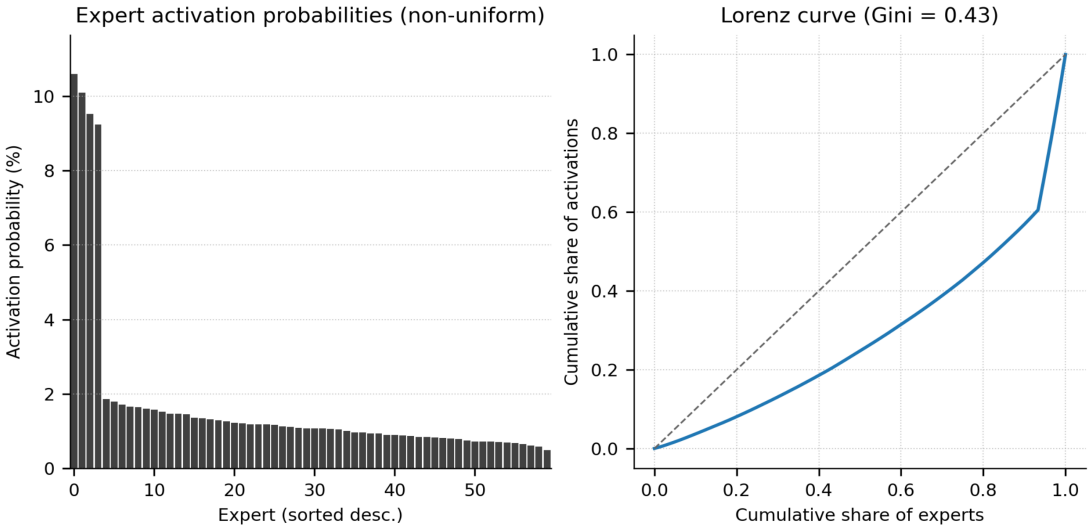


Figure 9: Expert activations rate in whole *math7K* dataset. Expert Activations are tested using the Deepseek-V2-Lite base model.

Figure 9 presents the activation statistics of all experts in the DeepSeek-V2-Lite base model when evaluated on the full *math7K* dataset. The left panel illustrates the sorted activation probabilities and reveals a highly non-uniform routing pattern, where a small subset of experts is used disproportionately often, with several exceeding a 10% activation rate. Beyond this small group, activation frequencies decrease sharply, forming a long tail of under-utilized experts with probabilities falling below 2% and eventually approaching zero. This distribution indicates that, although the MoE architecture allocates equal computational capacity to every expert, the router naturally converges toward a skewed, winner-take-most configuration.

The right panel shows the corresponding Lorenz curve, which characterizes the cumulative distribution of activations across experts. The curve deviates substantially from the diagonal line representing a perfectly uniform distribution, yielding a Gini coefficient of 0.43. This quantitatively confirms a notable degree of inequality in expert utilization, where a relatively small fraction of experts accounts for the majority of routing decisions.

Despite this concentration of activation mass, pruning experiments demonstrate that activation frequency alone is not a reliable indicator of expert importance. Removing the least-active experts leads to clear accuracy degradation, and even when retaining the top 75% of experts ranked by activation probability, the model still suffers a performance drop exceeding 10% on downstream math reasoning benchmarks. Details can be found in Appendix D. This observation suggests that low-activation experts encode specialized competencies that remain essential for certain categories of problems. Consequently, reducing experts solely based on activation sparsity risks eliminating critical functional diversity, thereby motivating the need for structured or distillation-based compression approaches—such as the ExpertCondenser method proposed in this work—to safely reduce mixture-of-experts models.

Q Ablation Studies

Q.1 Effect of the Number of Condenser Experts

In this section, we study the impact of the number of condenser experts r on downstream performance. As discussed in the main text, r controls a tradeoff between aggregation capacity and gradient concentration: larger r increases representational capacity but dilutes gradient updates across more experts, while smaller r concentrates gradients but may limit the ability to capture diverse patterns.

To evaluate this tradeoff, we conduct an ablation study on **DeepSeek-V2-Lite (16B)** by varying $r \in \{1, 2, 4\}$. All other training settings are kept fixed. The results are reported in Table 13.

We observe that increasing r from 1 to 2 improves performance across benchmarks, suggesting that a single condenser expert is insufficient to capture the diversity of routed expert knowledge. However, further increasing r to 4 yields mixed results: while performance improves on certain datasets (e.g., MultiArith and mawps), it degrades on others (notably AQuA), and does not improve the overall average. This behavior is consistent with the tradeoff described above: a larger r increases capacity but reduces gradient concentration per expert, weakening the effectiveness of knowledge aggregation.

Overall, $r = 2$ achieves the best balance between capacity and consolidation, validating the tradeoff described in the main text.

Table 13: Ablation study on the number of condenser experts r using DeepSeek-V2-Lite (16B). Results are zero-shot accuracy (%) on math reasoning benchmarks.

Dataset	Model	r	GSM8k	SingleEq	SVAMP	MultiArith	AddSub	AQuA	mawps	AVG
math7k	DeepSeek-V2-Lite	$r = 1$	58.9	92.1	68.8	90.3	77.6	35.9	82.5	72.2
		$r = 2$	**59.4**	**92.5**	**69.1**	**91.5**	**79.5**	**36.1**	**83.6**	**73.1**
		$r = 4$	58.6	92.1	69.7	92.7	78.7	33.5	85.7	73.0

Q.2 How to Choose Condenser Experts

In this subsection, we conduct an ablation study to investigate how experts should be selected as shared experts during post-training. Table 14 reports the results of comparing two selection strategies: (i) choosing high-bias experts and (ii) choosing low-bias experts. Across both **DeepSeek-V2-Lite (16B)** and **Qwen1.5-MoE (14B)**, we observe that selecting *low-bias experts* consistently leads to stronger downstream performance on math reasoning benchmarks. For example, in the **math7k** setting, low-bias experts achieve higher average accuracy (73.1 vs. 72.4 for DeepSeek-V2-Lite and 63.4 vs. 61.8 for Qwen1.5-MoE). These results suggest that low-bias experts encode more generalizable knowledge, making them more effective as shared experts in MoE post-training.

Proposition (Low-Bias Experts as Effective Aggregators). Let $\tilde{s}_i(x) = s_i(x) + b_i$ denote the biased routing logits under auxiliary-free routing, and let $p_i = \mathbb{P}(i \in S(x))$ denote the empirical activation probability of expert i. While bias b_i influences routing, activation frequency p_i is an empirical property that emerges from both routing logits and data distribution. Therefore, low-bias experts are not strictly equivalent to low-activation experts.

Under standard sparse routing, experts with lower activation probabilities p_i receive fewer gradient updates due to their infrequent selection. Promoting such experts to condenser experts (i.e., enforcing $i \in S(x)$ for all x) removes this sparsity factor and increases their expected gradient exposure.

As a result, low-activation experts can transition from under-trained specialists to persistent aggregation nodes that accumulate gradient signals across diverse inputs. In contrast, high-activation experts already receive frequent updates, so promoting them provides limited additional benefit. Randomly selected experts fall between these two extremes, providing moderate but less targeted aggregation.

To validate this, we compare five selection strategies for choosing condenser experts: (i) low-bias, (ii) high-bias, (iii) low-activation, (iv) high-activation, and (v) random selection.

Table 14: Evaluation of post-trained models (Zero-Shot results) on downstream Math Reasoning datasets to conduct ablation study on expert selection strategies, including bias-based, activation-based, and random selection.

Dataset	Model	Model Size	#Param (Experts)	Selection Type	GSM8k	SingleEq	SVAMP	MultiArith	AddSub	AQuA	mawps	AVG
math7k	**DeepSeek-V2-Lite**	**16B**	**2.4B**	high-bias experts	60.1	90.2	70.4	90.2	74.2	37.0	84.5	72.4
				low-bias experts	59.4	92.5	69.1	91.5	79.5	36.1	83.6	73.1
				low-activation experts	59.2	91.8	69.7	90.9	78.6	35.8	83.9	72.8
				high-activation experts	59.8	90.5	70.0	90.6	75.1	36.7	84.0	72.4
				random experts	59.2	91.0	70.6	91.0	79.2	36.2	83.2	72.9
	Qwen1.5-MoE	**14B**	**2.7B**	high-bias experts	54.7	71.0	53.8	84.6	58.8	32.8	76.8	61.8
				low-bias experts	57.2	74.6	55.7	86.0	61.8	33.1	75.6	63.4
				low-activation experts	56.6	73.9	55.1	86.4	62.0	32.7	75.0	63.0
				high-activation experts	55.0	73.8	54.2	84.9	60.4	32.5	76.2	62.4
				random experts	56.1	72.8	55.0	85.2	60.3	32.9	75.4	62.5

Q.3 Gated versus Ungated Condenser Experts

The ablation in § 5.4 varies whether a persistent path exists, not whether that path is gated, so it cannot separate the always-active capacity from the per-token weighting applied to it. We isolate the two by changing a single term of Eq. 3: the condenser contribution $g_j(x_t)\, E_j(x_t)$ becomes $c_j\, E_j(x_t)$ with c_j=1, which is exactly an ungated shared expert. Table 15 reports the control on Qwen1.5-MoE-A2.7B / `math7k` with r=2, one epoch, and identical initialization and hyperparameters across rows. The ungated variant averages 59.3, matching the 59.4 of *aux-free+bias*, which has no persistent path at all; gating the same path raises the average to 63.4. Always-active capacity therefore yields no gain on its own—the improvement comes from keeping the persistent path token-dependent, as argued in § 3.1.

Table 15: Gating control on Qwen1.5-MoE-A2.7B / `math7k` (r=2, 1 epoch). The last two rows differ only in the condenser weighting: a fixed c_j=1 versus the router score $g_j(x_t)$. The *aux-free+bias* and gated rows are the corresponding rows of Table 6.

Variant	GSM8k	SingleEq	SVAMP	MultiArith	AddSub	AQuA	mawps	AVG
SFT	45.8	70.2	53.6	76.2	53.3	27.6	67.8	56.4
aux-free+bias	48.2	76.6	52.7	80.3	59.2	26.8	71.8	59.4
+ ungated condenser (c_j=1)	52.7	71.4	50.4	82.3	60.3	26.8	71.5	59.3
+ gated condenser ($g_j(x_t)$)	**57.2**	74.6	**55.7**	**86.0**	**61.8**	**33.1**	**75.6**	**63.4**

R Training Stability Analysis of ExpertCondenser

Figure 10 presents the training dynamics of EXPERTCONDENSER compared with the ESFT and DenseMixer baselines when post-training the GPT-OSS model on the *math7K* dataset. The left panel plots the training-loss curves, while the right panel reports the corresponding gradient norms over the full optimization trajectory.

Across training, EXPERTCONDENSER displays markedly improved stability and convergence efficiency relative to the baselines. Its loss curve decreases rapidly during early optimization and consistently settles at a lower final value than both ESFT and DenseMixer. This indicates more effective optimization dynamics and better alignment with the target reasoning distribution. ESFT converges to a higher asymptotic loss, whereas DenseMixer converges slower.

The gradient-norm behavior further highlights this stability advantage. After the initial warm-up phase, EXPERTCONDENSER maintains smooth and well-bounded gradients, free of the high-magnitude spikes observed in DenseMixer and the early instability exhibited by ESFT. The absence of gradient bursts suggests that EXPERTCONDENSER operates over a more stable gradient landscape, which contributes to improved convergence and reduced optimization volatility.

Taken together, these results demonstrate that EXPERTCONDENSER not only delivers better downstream accuracy but also enhances the stability and reliability of the training process compared to existing post-training approaches.

Figure 10: Training stability comparison among EXPERTCONDENSER, ESFT, and DenseMixer. (a) Training loss curves, where EXPERTCONDENSER converges more quickly and reaches a lower final loss. (b) Gradient-norm evolution, showing that EXPERTCONDENSER maintains smooth, stable gradients without the spikes observed in DenseMixer and ESFT.

S Formalizing the Motivation Behind Condenser Experts

Mixture-of-Experts (MoE) architectures rely on two forms of expert capacity: (i) *sparse routed experts*, selected dynamically via top-k routing, and (ii) *static shared experts*, which are always active and contribute a fixed residual transformation.

The static shared experts contribute

$$\sum_{k=1}^{\hat{n}} E_k(\mathbf{x}_t),$$

independent of routing decisions. While this provides persistent capacity, these experts are ungated and therefore cannot adapt their contribution to the input or interact with routing dynamics.

To address this limitation, we introduce **condenser experts**, which are *always selected* yet remain *gated per token*. Let $J \subset \{1, \ldots, n\}$ denote the set of condenser experts with $|J| = r$. For each token $\mathbf{x}_t$, the active expert set is defined as

$$S(\mathbf{x}_t) = J \cup \mathrm{TopK}_{i \notin J}\left(\tilde{s}_i(\mathbf{x}_t),\, k - r\right).$$

Following Eq. 3, the MoE output is

$$h'_t = \underbrace{\sum_{j \in J} g_j(\mathbf{x}_t)\, E_j(\mathbf{x}_t)}_{\text{gated condenser experts}} + \underbrace{\sum_{i \in S(\mathbf{x}_t) \setminus J} g_i(\mathbf{x}_t)\, E_i(\mathbf{x}_t)}_{\text{selected sparse experts}} + \underbrace{\sum_{k=1}^{\hat{n}} E_k(\mathbf{x}_t)}_{\text{static shared experts}}.$$

This formulation highlights that condenser experts define a *persistent but input-dependent computation path*, in contrast to static shared experts which are input-independent.

S.1 Router Stability and Anti-Collapse Behavior

Condenser experts also improve routing stability. The gradient with respect to router parameters ϕ is

$$\nabla_\phi \mathcal{L}(\mathbf{x}) = \sum_{i \in S(\mathbf{x})} \nabla_\phi g_i(\mathbf{x})\, E_i(\mathbf{x}),$$

where $J \subset S(\mathbf{x})$ guarantees that part of this gradient is always present.

This acts as a structural regularizer by: (i) reducing routing variance, (ii) ensuring consistent gradient flow, and (iii) mitigating collapse of long-tailed experts.

Unlike static shared experts, which bypass routing, condenser experts propagate both forward and backward signals through the router.

S.2 Capacity Constraints and Variance Reduction

By enforcing $J \subset S(\mathbf{x}_t)$, the MoE layer incorporates a structured capacity constraint:

$$|S(\mathbf{x}_t)| = k, \quad J \subset S(\mathbf{x}_t).$$

Under stochastic routing, the output variance can be decomposed as

$$\mathrm{Var}[h'_t] = \mathrm{Var}\left[\sum_{i \in S(\mathbf{x}_t) \setminus J} g_i(\mathbf{x}_t)\, E_i(\mathbf{x}_t)\right] + \mathrm{Var}\left[\sum_{j \in J} g_j(\mathbf{x}_t)\, E_j(\mathbf{x}_t)\right].$$

The second term is invariant to routing selection randomness, since condenser experts are always active. This reduces stochasticity induced by dynamic routing and leads to more stable optimization.

T Hyper-parameters

T.1 Math7K Dataset and Math14K Dataset

In this subsection, we perform an ablation study to verify whether the number of fine-tuning epochs is sufficient for convergence. Since training efficiency and stability are critical in post-training large MoE models, it is important to ensure that extending training does not yield further improvements or lead to overfitting. We therefore evaluate the performance of **DeepSeek-V2-Lite (16B)** and **OLMoE (7B)** under the **math7k** and `math14k` benchmark with **ESFT** fine-tuning for 1, 2, and 3 epochs. The results are reported in Table 17 and 16.

Overall, these findings confirm that our main experiments are conducted with models that have already converged, and that increasing the number of training epochs does not lead to meaningful gains.

Table 16: Evaluation of SFT model Zero-Shot Results on downstream math reasoning tasks after fine-tuning with *Math-14K*, including SingleEQ, MultiArith, AddSub, GSM8K, SVAMP, and AQuA.

Model	Model Size	#Param (Experts)	Distill Type	GSM8k	SingleEq	SVAMP	MultiArith	AddSub	AQuA	mawps	AVG
OLMoE	**7B**	**1B**	SFT-1epoch	55.7	76.2	58.8	71.2	62.8	28.3	64.3	59.6
			SFT-2epoch	52.8	78.3	59.1	71.8	63.3	29.1	68.9	60.5
			SFT-3epoch	52.6	77.2	57.8	72.7	64.6	31.9	70.1	60.9

Table 17: Evaluation of SFT model Zero-Shot Results on downstream math reasoning tasks after fine-tuning with *Math-7K*, including SingleEQ, MultiArith, AddSub, GSM8K, SVAMP, and AQuA.

Model	Model Size	#Param (Experts)	Distill Type	GSM8k	SingleEq	SVAMP	MultiArith	AddSub	AQuA	mawps	AVG
DeepSeek-V2-Lite	**16B**	**2.4B**	ESFT-1epoch	54.1	88.0	65.3	83.7	72.7	26.8	79.4	67.1
			ESFT-2epoch	58.6	80.9	65.8	90.7	62.3	27.6	76.1	66.0
			ESFT-3epoch	58.2	75.8	65.2	89.0	56.5	29.5	73.5	64.0
OLMoE	**7B**	**1B**	SFT-1epoch	57.0	78.5	58.6	72.0	64.3	28.3	76.1	62.1
			SFT-2epoch	53.8	70.9	55.7	65.0	61.5	31.5	69.7	58.3
			SFT-3epoch	50.3	69.3	49.5	54.5	59.2	27.6	59.2	52.8

T.2 Gamma for Bias Update

Table 18: Evaluation of different post-trained Qwen1.5 model Zero-Shot Results on downstream math reasoning tasks with different gamma settings after fine-tuning with *Math-7K*, including SingleEQ, MultiArith, AddSub, GSM8K, SVAMP, and AQuA.

Model Gamma	**Model Size**	**Distill Type**	**#Param (Experts)**	**GSM8k**	**SingleEq**	**SVAMP**	**MultiArith**	**AddSub**	**AQuA**	**mawps**	**AVG**
G-1e-6	**14B**	aux-free-loss	**2.7B**	48.6	76.8	50.0	81.8	59.6	29.6	70.6	59.6
G-1e-5	**14B**	aux-free+bias	**2.7B**	48.8	76.4	50.2	81.6	59.8	29.7	70.8	59.6
G-5e-5	**14B**	aux-free+bias	**2.7B**	47.8	76.6	50.2	81.8	60.3	29.9	70.6	59.6
G-1e-4	**14B**	aux-free+bias	**2.7B**	48.2	76.6	52.7	80.3	59.2	26.8	71.8	59.4
G-1e-3	**14B**	aux-free+bias	**2.7B**	47.7	74.2	50.6	83.2	59.0	30.7	67.2	58.9
G-3e-3	**14B**	aux-free+bias	**2.7B**	31.8	58.5	37.9	65.2	39.5	28.3	56.7	45.4
G-5e-3	**14B**	aux-free+bias	**2.7B**	15.3	32.7	26.0	47.5	29.4	21.7	33.2	29.4
G-1e-2	**14B**	aux-free+bias	**2.7B**	7.2	17.3	14.7	28.2	15.7	17.7	18.5	17.0

Table 19: Evaluation of different post-trained Deepseek-v2-lite model Zero-Shot Results on downstream math reasoning tasks with different gamma settings after fine-tuning with *Math-7K*, including SingleEQ, MultiArith, AddSub, GSM8K, SVAMP, and AQuA.

Model Gamma	Model Size	Distill Type	#Param (Experts)	GSM8k	SingleEq	SVAMP	MultiArith	AddSub	AQuA	mawps	AVG
G-1e-6	**14B**	aux-free-loss	**2.7B**	56.2	89.4	68.9	86.2	73.2	35.2	79.0	69.8
G-1e-5	**14B**	aux-free-loss	**2.7B**	56.7	89.6	69.8	87.8	74.0	35.8	79.6	70.5
G-5e-5	**14B**	aux-free-loss	**2.7B**	58.6	90.6	70.2	88.6	74.4	36.2	80.6	71.3
G-1e-4	**14B**	aux-free-loss	**2.7B**	58.8	90.7	69.3	88.7	74.2	36.1	80.3	71.2
G-1e-3	**14B**	aux-free-loss	**2.7B**	43.4	78.7	62.8	80.5	60.5	25.2	71.0	60.3