EMO: Pretraining Mixture of Experts for Emergent Modularity

Ryan Wang[α,β] **Akshita Bhagia**[β] **Sewon Min**[α,β]

[α]University of California, Berkeley [β]Allen Institute for AI

ryanyxw@berkeley.edu akshitab@allenai.org sewonm@berkeley.edu

🤗	**Model**	hf.co/collections/allenai/emo
	Code	github.com/allenai/EMO
	Blog	allenai.org/blog/emo
	Visualization	https://emovisualization.netlify.app

Abstract

Large language models are typically deployed as monolithic systems, requiring the full model even when applications need only a narrow subset of capabilities, e.g., code, math, or domain-specific knowledge. Mixture-of-Experts (MoEs) seemingly offer a potential alternative by activating only a subset of experts per input, but in practice, restricting inference to a subset of experts for a given domain leads to severe performance degradation. This limits their practicality in memory-constrained settings, especially as models grow larger and sparser. We introduce **EMO**, an MoE designed for modularity—the independent use and composition of expert subsets—without requiring human-defined priors. Our key idea is to encourage tokens from similar domains to rely on similar experts. Since tokens within a document often share a domain, EMO restricts them to select experts from a shared pool, while allowing different documents to use different pools. This simple constraint enables coherent expert groupings to emerge during pretraining using document boundaries alone. We pretrain a 1B-active, 14B-total EMO on 1T tokens. As a full model, it matches standard MoE performance. Crucially, it enables selective expert use: retaining only 25% (12.5%) of experts incurs just a 1% (3%) absolute drop, whereas standard MoEs break under the same setting. We further find that expert subsets in EMO specialize at semantic levels (e.g., domains such as math or code), in contrast to the low-level syntactic specialization observed in standard MoEs. Altogether, our results demonstrate a path toward modular, memory-efficient deployment of large, sparse models and open new opportunities for composable architectures.

1 Introduction

Large language models (LLMs) are typically trained and deployed as monolithic systems: a single model is pretrained, finetuned, and served as one unified entity [1, 2, 3]. While effective, this paradigm becomes increasingly restrictive as models scale. In many deployment settings, applications require only a narrow subset of capabilities—such as code generation, mathematical reasoning, or domain-specific knowledge—but must still serve the full model, incurring unnecessary computational cost and memory use. Moreover, the monolithic design prevents isolating, updating, or improving specific capabilities without retraining and redeploying the entire system.

Mixture-of-Experts (MoE) models appear to offer a natural path toward relaxing this constraint, as they consist of many small FFNs (*experts*), of which only a small subset is activated for each input

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

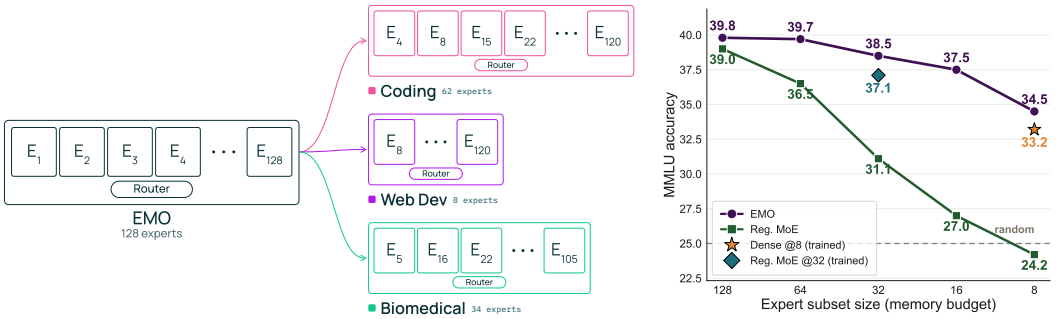

Figure 1: **(Left)** EMO is an MoE trained with modularity as a first-class objective. For a given domain (e.g., math, code, biomedical), users can select a small subset of experts of any size and retain near full-model performance. This turns a single model into a composable architecture, enabling flexible deployment with improved memory-accuracy tradeoffs for large, sparse MoEs. **(Right)** Averaged performance over 16 MMLU categories across different memory budgets. EMO (purple) and Reg. MoE (green) are single models evaluated with expert subsets of different sizes. EMO expert subsets push the Pareto frontier in memory-accuracy trade-off, outperforming standard MoEs and even fixed-budget models trained from scratch.

token [2, 4]. However, existing MoEs still require the full model for any task: tokens within the same input activate different experts, causing most or all experts to be used over the course of a task. As we show, this behavior—partially driven by experts specializing in low-level lexical patterns (e.g., prepositions, punctuation)—prevents subsets of the model from being usable independently, limiting the deployability of MoEs in memory-constrained settings, an issue that becomes increasingly important as models grow larger and sparser [5, 2, 3].

We instead seek to train MoE models in which experts organize into coherent groups that can be selectively used and composed. Concretely, we train an MoE model to be *modular*, i.e., to support (1) the independent use of expert subsets and (2) their composition into a strong general-purpose model. Achieving this in practice, however, is challenging. Prior work has explored partitioning training data into predefined domains (e.g., math, coding) and training separate experts [6, 7], but this is too restricted for model's learning and limits the model's overall performance.

In this work, we propose to train MoE models in which modular structure emerges directly from the data, without relying on human-defined prior, and introduce **EMO**, an MoE that follows this approach. Our key intuition is that tokens from the similar domains should activate similar subsets of experts. Assuming that tokens within a document tend to share a domain, we enforce this structure by restricting all tokens in a document to select their active experts from a shared pool. For example, in an MoE with 128 total and 8 active experts, all tokens from a document select their active subset from a shared pool of 32 experts. Different documents may use different expert pools, allowing the model to learn recurring expert subsets across the training corpus. Importantly, EMO does not require predefined task or domain labels: expert subsets emerge in a self-supervised way, using document boundaries as the only grouping signal.

We train a 1B-active, 14B-total parameter EMO model on 1 trillion tokens. As a full model, EMO matches the overall performance of a standard MoE. More importantly, however, it enables effective composition of expert subsets, which standard MoEs fail to support. Across domain-specific subsets of MMLU and MMLU-Pro (e.g., math, physics, biology, social sciences), identifying and deploying only the most relevant experts largely preserve performance, e.g., 1% absolute performance drop when retaining 25% of experts, and 3% when retaining 12.5%. This is in contrast to standard MoEs that see severe degradation under the same constraint, e.g., 10% and 15% drops, respectively. These results show that EMO makes MoEs significantly more practical and accessible: instead of loading the full model, one can serve only a small subset of experts relevant to a given task or domain (Figure 1), which has important implications for deployment in memory-constrained settings [8, 9, 10].

We further analyze routing patterns and find that expert subsets specialize at higher-level semantic granularity, such as domains and topics (e.g., math, code), which is in contrast to experts in standard MoEs that specialize in lower-level syntactic patterns (e.g., prepositions, punctuation). This difference suggests that expert specialization in EMO is qualitatively distinct and underlies its modularity.

Together, these results demonstrate that modularity can be built into large language models, opening a path for broader functionalities, such as targeted extension training or more interpretable and debuggable components to better regulate model behavior. We release both EMO and a matched baseline trained on the same data to support reproducibility and further study.

2 Related Work

Mixture-of-Experts as Scalable Architectures. Mixture-of-Experts (MoE) architectures introduce sparsity into Transformers by activating only a subset of experts per input, enabling efficient scaling to very large models [11, 12, 13]. Recent systems push this paradigm further by increasing both the number of experts and the degree of sparsity—for example, DeepSeek-V3 [2] employs hundreds of experts per layer while activating only a small subset per token—allowing models to reach scales of hundreds of billions of parameters.

As MoEs grow larger and sparser, memory bottlenecks become a central challenge: even inactive experts need to reside in VRAM at inference time. This has motivated a line of work such as memory-constrained scaling laws [14], memory-efficient serving [8, 9, 10], and expert pruning for a general purpose model that removes redundant experts [15]. This work introduces an MoE that enables selective use of expert subsets for a given downstream task. Among its benefits, this provides a new way to alleviate memory bottlenecks in large, sparse MoEs, complementary to prior approaches.

Specialization and Modularity of Existing MoEs. A growing body of work studies the extent to which specialization emerges in MoE models. Prior work finds that specialization is often driven by surface-level patterns (e.g., token ID that is context-independent) or low-level lexical cues (e.g., prepositions, punctuations) [16, 17], while other works find that specialization is confined to only a tiny subset of experts [18]. Other work suggests that apparent expert specialization can reflect geometric properties of the representation space that is difficult to interpret [19]. In parallel, several works attempt to exploit these patterns for efficiency, for example by pruning experts for a given task [20, 21, 15, 22, 23].

In this work, we show that standard MoEs trained with conventional objectives do not support meaningful use of small expert subsets for downstream domains, and instead advocate for training an MoE with modularity as a first-class objective. When training accordingly, MoEs naturally support selective use of expert subsets, and this behavior is robust across different subset selection methods.

Training MoEs with Structured or Specialized Experts. Prior work has explored training MoEs with more structured or specialized experts. One line of work promotes interpretability or diversity across experts, primarily to reduce redundancy [24, 25, 26, 27], but such approaches do not ensure that expert subsets are usable in isolation. Another line of work explicitly partitions training data into predefined domains (e.g., math, biomedical), train separate experts, and merge them into a single MoE [7, 6, 28]. While this enables standalone use of expert subsets, it relies on fixed, human-defined priors, which restricts flexibility and limits overall model performance. In contrast, we train an MoE end-to-end with modularity as a first-class objective, allowing expert structure to emerge without requiring predefined domains or human priors.

The closest line of work is ModuleFormer [29], which shares our goal of training a modular MoE that supports standalone use of expert subsets. It introduces an objective that maximizes mutual information between tokens and experts. However, they evaluate only against dense models, without standard MoEs. We attempted to reproduce ModuleFormer and found that they do not perform better than standard MoEs (see §B.7). Furthermore, the original ModuleFormer paper already reports that performance degrades significantly when less than 40% of experts are retained. In contrast, EMO largely shares the motivation with ModuleFormer but proposed a more effective training objective that significantly outperforms standard MoEs and other parameter-matched and memory-matched baselines during selective expert deployment, showing minimal degradation even with an expert subset size of just 12.5%.

3 Modular Mixture of Experts (EMO)

The goal of EMO is to pre-train an MoE with modularity as the first-class objective, i.e., (1) expert subsets should be usable in isolation for a particular downstream domain, and (2) their composition—the full model—remains a strong general-purpose model.

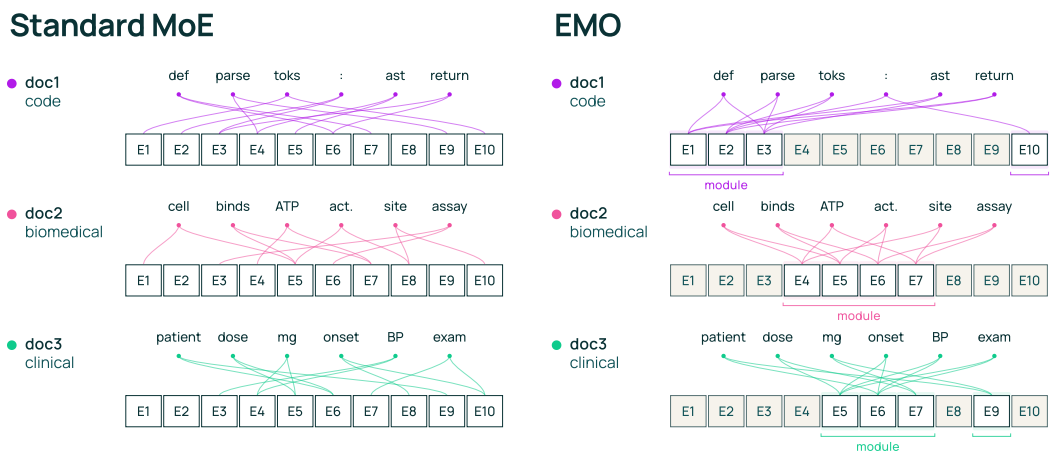

Figure 2: Comparison of training of a standard MoE and EMO ($k = 2, n = 10$, shared experts omitted for simplicity). **(Left)** In a standard MoE, each token independently selects its top-k experts. **(Right)** In EMO, the router first selects a subset of experts for each document, and all tokens are constrained to route within this subset. This enforces consistent expert usage across the document, encouraging groups of experts to form domain specialization.

Naive Approach. A straightforward approach to develop modularity is to enforce expert specialization in MoEs by routing tokens to experts based on predefined semantic domains (e.g., math, biology, code). Methods such as FlexOlmo [7] and BTX [6] instantiate this idea. However, this formulation requires domain labels across pretraining data, which can be ambigious, difficult to obtain, and injects human biases. Having fixed domains also restricts flexibility, making it difficult for the model to be applied to new domains during inference.

EMO's Approach. Instead, we induce modular structure without explicit domain labels (Figure 2). Our key observation is that tokens within the same document usually come from the same domain. We therefore treat *document boundaries* as a *weak supervisory signal*: for each document, the router selects a shared expert pool, and all tokens in that document choose their active experts only from this pool. Different documents can use different pools, allowing modular expert subsets to emerge directly from the training data.

In the rest of the section, we first describe the standard MoE architecture and objective (§3.1), then describe EMO's training objective (§3.2).

3.1 Preliminary: Mixture of Experts Architecture

Mixture-of-Experts (MoE) models are decoder-only Transformer language models [30] in which the feedforward sublayer is replaced by a sparse mixture of expert networks. Let the model contain n total experts, consisting of n_r routed experts and n_s shared experts ($n = n_r + n_s$). Routed experts are selected dynamically on a per-token basis, while shared experts are always active.

Given the hidden state x_t at token position t, a router produces logits over the routed experts,

$$r(x_t) \in \mathbb{R}^{n_r}, \qquad p_t = \mathrm{softmax}(r(x_t)).$$

Let $\mathcal{K}_t = \text{Top-}K(p_t, k) \subseteq \{1, \ldots, n_r\}$ denote the indices of the top-k routed experts selected for token t. The MoE feedforward output is then

$$\mathrm{FFN}_{\mathrm{out}}(x_t) = \sum_{i \in \mathcal{K}_t} (p_t)_i \, E_i(x_t) \; + \; \sum_{j=1}^{n_s} E_j^{(s)}(x_t),$$

where E_i denotes the i-th routed expert and $E_j^{(s)}$ denotes the j-th shared expert.

The resulting $\mathrm{FFN}_{\mathrm{out}}(x_t)$ is used throughout the forward pass of the model to compute token probabilities. We train the model using the standard autoregressive language modeling objective:

$$\mathcal{L}_{\mathrm{CE}} = -\sum_{t=1}^{T} \log P(x_t \mid x_{<t}),$$

where the conditional probabilities $P(x_t \mid x_{<t})$ are computed using the MoE layer defined above.

In addition to the cross entropy, MoE training includes auxiliary losses such as the load balancing loss $\mathcal{L}_{\text{LB}}$ to encourage uniform expert utilization:

$$\mathcal{L}_{\text{LB}} = n_r \sum_{i=1}^{n_r} \bar{f}_i \cdot \bar{P}_i,$$

where $\bar{f}_i$ is the fraction of tokens routed to expert i and $\bar{P}_i$ is the average routing probability of expert i across all tokens. The full objective is

$$\mathcal{L} = \mathcal{L}_{\text{CE}} + \alpha \mathcal{L}_{\text{LB}} + \beta \mathcal{L}_{\text{RZ}},$$

where $\mathcal{L}_{\text{RZ}}$[1] regularizes router logits, and α and β control auxiliary loss weights.

3.2 EMO: An Objective to Induce Modularity

The goal of EMO is to induce modularity by leveraging *document boundaries* as a weak supervisory signal. EMO achieves this by selecting a *document expert pool* for each document and constrains all tokens in the document to route within this pool during training (Figure 2).

Formulation. Recall that $p_t = \text{softmax}(r(x_t)) \in \mathbb{R}^{n_r}$ denotes the routing distribution for token t. For each document, we define the document expert pool $\mathcal{D}$ based on the average routing distribution across tokens in that document:

$$\mathcal{D} = \text{Top-}K\left(\frac{1}{T}\sum_{t=1}^{T} p_t,\ d\right) \subseteq \{1, \ldots, n_r\}.$$

Note that for different documents, a different $\mathcal{D}$ is chosen. Routing for each document is then restricted to their corresponding $\mathcal{D}$ via a masked and renormalized distribution

$$\hat{p}_t(i) = \begin{cases} \dfrac{p_t(i)}{\sum_{j \in \mathcal{D}} p_t(j)} & \text{if } i \in \mathcal{D}, \\ 0 & \text{otherwise.} \end{cases}$$

The routed experts are then

$$\mathcal{R}_t = \text{Top-}K(\hat{p}_t, k).$$

The resulting feedforward output is

$$\text{FFN}_{\text{out}}(x_t) = \sum_{i \in \mathcal{R}_t} (\hat{p}_t)_i \, E_i(x_t) \ + \ \sum_{j=1}^{n_s} E_j^{(s)}(x_t),$$

where E_i denotes routed experts and $E_j^{(s)}$ denotes shared experts.

The hyperparameter d controls subset granularity: smaller d enforces highly specialized expert subsets with limited expressivity (e.g., $d = k$ forces all tokens in a document to use the same experts), while larger d increases flexibility at the cost of weaker modular structure (e.g., $d = n_r$ recovers the standard MoE). Despite the additional document-level routing constraint, EMO incurs negligible training overhead compared to standard MoEs (see §A.7).

3.3 Key Technical Considerations

Several technical choices were important for effective training of EMO (see §A for details).

Consideration 1. Load Balancing. A central challenge is that load balancing and document-level routing appear to impose opposing pressures. This conflict arises under standard micro-batch load balancing, where the load-balancing loss is computed over only a few documents. While this local implementation reduces cross-device communication and simplifies distributed training, it also

[1] $\mathcal{L}_{rz} = \frac{1}{T}\sum_{t=1}^{T} (log \sum_{i=1}^{n_r} exp(r(x_t)_i)))^2$, which we take from OLMoE [17]

encourages tokens from the same document to spread across many experts, directly opposing the shared-pool constraint and causing unstable training.

We address this by adopting global load balancing [31], aggregating routing statistics across data-parallel groups. Applied over a larger and more diverse set of documents, load balancing encourages uniform utilization of experts *across* documents, while our routing constraint enforces expert consistency *within* each document, making the two objectives largely complementary. Empirically, this is important for stable training: see Figure 6 in §A.

Consideration 2. Choosing Expert Pool Size. Fixing a single expert pool size d works well during training but limits inference-time flexibility. The model "overfits" only to expert sets of size d and performs poorly when deployed as expert subsets that isn't of size d.

To enable the model to support expert subsets of all sizes, we treat d as a random variable and sample it independently for each document during pretraining:

$$d \sim \mathcal{U}\{k, \ldots, n_r\}.$$

where k is the number of active experts per token and n_r is the total number of routable experts. This exposes the model to a range of expert pool sizes during training, enabling it to support expert subsets of varying capacities for selective expert use.

4 Experimental Setup

4.1 Architecture & Training Details

We consider an MoE with 1B active and 14B total parameters, consisting of $n = 128$ experts ($n_r = 127$ routed, $n_s = 1$ shared), with $k = 8$ experts activated per token. The baseline MoE and EMO share the same architecture; they differ only in their training objectives, as described in §3.2.

We train both the baseline MoE and EMO from scratch on 1 trillion tokens from the OLMoE pretraining corpus [17], followed by a 50B-token linear annealing phase. For ablations, we additionally train models on 130B tokens and include comparison to dense baselines and smaller MoEs.

Our architecture largely follows that of OLMoE [17], with several key improvements: (1) adding a shared expert, (2) using pre-norm instead of post-norm, and (3) removing QK-norm; see §A for details on the improvements introduced by these changes. These modifications make our baseline MoEs significantly more competitive: as shown in §5.1, our baseline MoE trained on 1T tokens consistently outperforms OLMoE trained on 5T tokens despite being trained on the same data.

4.2 Evaluation

We evaluate our models under two settings: (1) **full-model evaluation**, reflecting the standard use case in which a pretrained model is deployed for a broad set of tasks, and (2) **selective expert use**, where only a task-specific subset of experts is activated for a particular task or domain. Additional details on evaluation tasks and settings are provided in §D.

Full-model Evaluation. We first evaluate the full model under zero-shot settings. We report results on five evaluation suites: (1) **MC9**, an average over nine multiple-choice benchmarks including ARC-Easy [32], ARC-Challenge [32], BoolQ [33], CSQA [34], HellaSwag [35], OpenBookQA [36], PIQA [37], SocialIQa [38], and WinoGrande [39]; (2) **Gen5**, an average over five generative tasks including CoQA [40], SQuAD [41], Natural Questions [42], TriviaQA [43], and DROP [44]; (3) **MMLU** [45][2]; (4) **MMLU-Pro** [46][2]; and (5) **GSM8K** [47].

Selective Expert Use. We next evaluate whether models can be deployed using only a subset of experts for each downstream domain (Figure 1). We consider coarse-grained domain grouping of MMLU and MMLU-Pro, e.g., math, physics, health, philosophy, history, which contain 16^2 and 13^2 domains, respectively, as well as GSM8K.

For each domain, we assume access to a small validation set to identify relevant experts. In §B.2, we show that this validation set can be extremely small: even a single few-shot example is sufficient

[2]Aggregated results exclude the "other" category; see §D and B.3 for details.

	# train tokens	MC9	Gen5	MMLU	MMLU Pro	GSM8K
OLMoE†	5T	63.5	57.6	42.8	18.7	13.7
Reg. MoE	1T	63.9	59.7	42.4	19.3	13.9
EMO (Ours)	1T	63.1	57.9	42.8	18.5	12.0
Dense	130B	54.1	41.5	33.0	12.2	2.7
Reg. MoE	130B	60.1	51.0	37.5	15.8	5.2
EMO (Ours)	130B	59.1	49.2	38.1	15.5	4.2

Table 1: **Full-model Evaluation (§5.1)**. All models are trained on the same data mixture, and activate the same number of parameters (1B). EMO matches the performance of a standard MoE. †: Use the outdated architecture (no pre-norm, use QK-norm, no shared expert, micro-batch load balancing) and has 64 total experts instead of 128.

to select an effective expert subset. We consider two selection methods: (1) a simple approach that aggregates routing probabilities across tokens and ranks experts by their average routing probability, and (2) Easy-EP [21], a more computationally expensive, state-of-the-art expert selection method. We then retain the top-d experts in each layer and discard the rest, producing a domain-specific subset of experts that can be used as a standalone model. We vary d to measure how performance changes as fewer experts are retained. We report both zero-shot performance and performance after finetuning. More evaluation details can be found in §D.

5 Results and Analysis

5.1 Full-Model Evaluation

Table 1 reports full-model performance for models trained on the same data with the same number of active parameters (1B). First, our baseline MoE is competitive, outperforming OLMOE [17] trained on 5x tokens. Nonetheless, EMO matches the performance of this standard MoE. The trend holds in the 130B-token setting: both our baseline MoE and EMO significantly outperform a dense model with matched active parameters, and EMO remains comparable to the standard MoE.

5.2 Selective Expert Use

We evaluate whether expert subsets in EMO can retain full-model performance for a given domain (Figure 3 for 1T tokens, Figure 10 for 130B tokens).

Standard MoEs (Green) Degrade Sharply. Restricting to expert subsets leads to large performance drops—for instance, over 10% when retaining 25% of experts (128→32), and below a dense model with matched active parameters (Figure 10). This trend is consistent with and without fine-tuning. These results show that standard MoEs do not support modular use: even when only a narrow set of capabilities is required, restricting to expert subsets causes performance to break down.

EMO (Purple) Enables Modular Use. In contrast, EMO retains performance under subset deployment. Performance drops are minimal, e.g., about 1% at 25% expert retention and 3% at 12.5%, and the model continues to outperform dense baselines, even in an extreme case where only 6.2% of experts are retained. This trend persists after fine-tuning, e.g., notably, on GSM8K, subsets with up to 12.5% of experts perfectly recover full-model performance. These results indicate that EMO supports modular use by identifying domain-relevant expert subsets, enabling significant memory savings by avoiding the need to load the full model (see §A.6).

Notably, we show in §B.2 that selecting relevant expert subsets is sample efficient, needing as few as five examples. We also provide examples of GSM8K generations in §B.5, demonstrating qualitative differences in generation quality between expert subsets of EMO and those for standard MoEs.

Expert Subsets of EMO Outperform Memory-matched Models Trained from Scratch. Figure 1 (right) compares EMO expert subsets against memory-matched models trained from scratch, including a standard MoE with 32 experts and a dense model. A full set of results can be found in §B.1. Despite using only a subset of the experts from a larger pretrained model, the 32-expert and 8-expert subsets

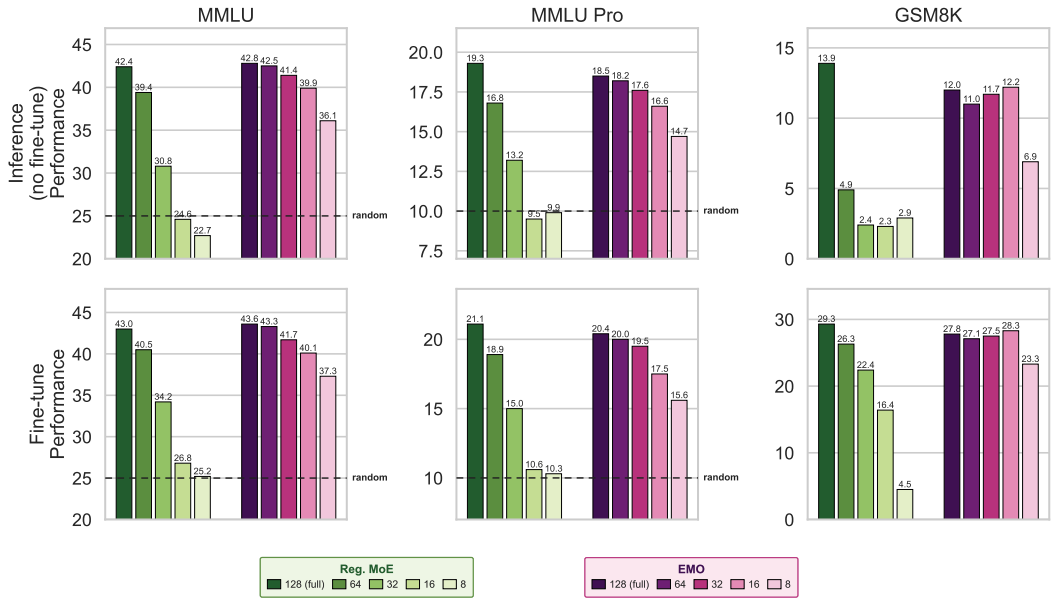

Figure 3: **Selective Expert Use** of MoEs trained on ***1T*** tokens (§5.2). Results are shown both without fine-tuning (*top*) and with fine-tuning (*bottom*). For MMLU and MMLU-Pro, each domain selects an expert subset as described in §4.2, and we report macro-averaged results across domains (16 for MMLU and 13 for MMLU-Pro). The baseline MoE degrades sharply under subset restriction, whereas EMO remains robust, with $\approx 1\%$ drop at 25% parameters and $\approx 3\%$ drop at 12.5%.

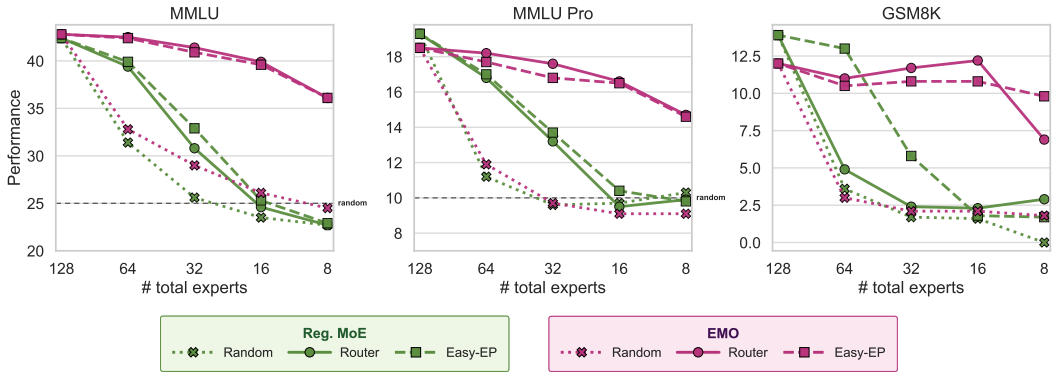

Figure 4: **Expert Selection Methods** in selective expert use of MoEs trained on *1T* tokens (§5.2). Results are without fine-tuning. For MMLU and MMLU-Pro, each domain selects a corresponding expert subset as described in §4.2. EMO expert subsets maintain high performance compared to regular MoEs across both router-based and Easy-EP expert-selection strategies. Random expert selection converges quickly to random performance as expert subsets shrink.

of EMO match or outperform these memory-matched baselines. These results suggest that expert subsets from a single EMO model offer a stronger memory-accuracy trade-off than models trained from scratch under fixed memory budgets, forming a new Pareto frontier across memory regimes.

EMO is Robust to Expert Selection Schemes. In addition to router-based selection, we evaluate EASY-EP [21], the state-of-the-art expert pruning method as an alternative selection strategy (Figure 4). As a sanity check, we also compare with random selection.

For a standard MoE, Easy-EP consistently outperforms router-based selection when larger subsets are retained, although performance still degrades sharply as the subset size decreases, consistent with observations from the original paper [21]. This suggests that even state-of-the-art selection methods cannot overcome the lack of localized domain-specific capabilities.

In contrast, EMO achieves strong performance under both router-based and Easy-EP selection methods. Performance is largely insensitive to the choice of selection scheme, and remains robust



Figure 5: **Token Clusters** of pretraining data on MoEs trained on *1T* tokens, clustered according to the process described in §5.3. Claude Code was used to assign a representative short description for each cluster. EMO clusters correspond to semantically meaningful domains, with tokens from the same document largely grouped together. Standard MoE training produces clusters of surface-level or syntactic features, with document tokens dispersed across multiple clusters.

even with small expert subsets. This highlights that modularity must be learned during training, rather than recovered through post hoc expert selection.

In §B.4, we also test whether modularity can emerge after pre-training by annealing a standard MoE with the document-level expert pool objective (§3.2); while training EMO from scratch performs best, the annealed model shows signs of modularity, which we leave for future work to investigate.

5.3 Semantic Specialization Emerge in EMO

We analyze how functional modularity emerges in EMO by examining expert specialization. We find a qualitative shift in behavior: while standard MoEs specialize at the lexical level, EMO induces specialization at the level of domains and topics (See §C for more analysis).

To study this, we cluster pretraining tokens based on their routing behavior. Specifically, we sample the first 100 tokens from 12K documents and extract routing probabilities across experts. We project these representations using PCA (retaining 95% variance), apply L2 normalization, and cluster them using spherical k-means with 32 clusters.

EMO Clusters Align with Semantic Themes. An interactive visualization is available at `emovisualization.netlify.app`;

Figure 5 shows representative cluster descriptions, assigned by Claude Code. First, in a standard MoE, clusters correspond to low-level lexical categories, such as "prepositions", "proper names", "copula verbs", or "definite articles", consistent with observations from prior work [16, 17]. As a result, tokens within a single document are dispersed across many clusters.

In contrast, EMO produces clusters aligned with high-level semantics and domains, such as "film, music, TV & book reviews", "health, medical & wellness", "news reporting", and "US. politics & elections". As a result, tokens within the same document are typically assigned to the same cluster, indicating consistent expert use.

This reveals two key insights. First, domain-level specialization emerges in EMO, even with no explicit supervision. Second, this specialization directly enables modularity: tokens from the same domain share routing patterns, allowing computation to be localized to a small, coherent subset of experts. As a result, expert subsets can be used effectively for downstream domains.

6 Future Directions

EMO is among the first MoE models with emergent modularity. While this work primarily focuses on modularity for efficient deployment, it also points to several broader opportunities.

Accessible Deployment of Large Sparse MoEs. As MoE models scale to trillions of parameters [48, 49], deploying or adapting them becomes increasingly resource-intensive. Prior work addresses this through memory-constrained scaling and optimized serving [14, 8, 9, 10]. Modularity offers an orthogonal path: selectively using small subsets of experts for a given domain, enabling more accessible deployment and adaptation, particularly well-suited for large, highly sparse models.

Fine-grained Control. Modularity can enable finer-grained control at inference time. Since EMO organizes experts along semantic domains, subsets could be selectively enabled or disabled depending on the application. For example, clusters associated with spam, gambling, or adult content (Figure 5) can be excluded in child-facing applications. Similarly, specialized domains, e.g., biomedical knowledge, may be valuable for benign use but risky in misuse scenarios, and could be conditionally exposed. This suggests a potential alternative to dataset-level filtering: isolating and managing capabilities at inference time depending on scenarios.

Modular Development and Maintenance. Modularity can also open up a new paradigm for model updates. Current language models are usually trained and maintained as one large system, requiring the full model, data, and compute to be available at once. A modular model could instead support modular pretraining: training task or domain-specific subset of experts and then reintegrating those experts into the full model.

As a preliminary test, we finetune a 32-expert subset from EMO, then inserted it back into the original 128-expert model by replacing the corresponding experts. The resulting model improves over the original full model, though it does not yet match the performance of the standalone subset. This provides early evidence that expert subsets can be trained independently and later integrated, which we leave to future work.

Higher Degrees of Monitorability. Finally, modularity can also make models easier to monitor and audit. Expert activations provide a structured signal of which parts of the model are being used for a given input. For example, if a model answers a math question while strongly activating an expert subset associated with creative writing or low-quality web content, that mismatch may warrant closer inspection. This gives model developers a more structured interface for understanding and debugging model behavior.

7 Conclusion

We introduced EMO, a mixture-of-experts model designed to make modularity emerge during pretraining. By constraining tokens within the same document to route through a shared expert pool, EMO induces expert subsets that specialize to high-level tasks and capabilities without relying on human-defined domains or task labels. Our results show that this structure does not come at the cost of general performance: as a full model, EMO matches standard MoEs, while its extracted expert subset remain effective even when only a small fraction of experts are retained. Beyond efficient deployment, our analyses show that EMO learns expert subsets aligned with semantic domains rather than surface-level token patterns, suggesting a qualitatively different form of specialization. Together, these results demonstrate that large language models need not remain monolithic systems. Modularity can be built into pretraining itself, opening a path toward models that are easier to deploy, adapt, inspect, and compose.

Acknowledgement

We thank Prasann Singhal, Gustavo Lucas Carvalho, Weijia Shi, Jagdeep Bhatia, Colin Raffel, Berkeley AI Research members, the Sky computing lab members, and Ai2 members for valuable discussion and feedback.

This research was supported in part by ONR (N00014-26-1-2233), the NVIDIA Academic Grant Program, and gifts from Ai2 and Apple. Ryan Wang was supported by the National Science Foundation Graduate Research Fellowship Program.

References

[1] Team Olmo, :, Allyson Ettinger, Amanda Bertsch, Bailey Kuehl, David Graham, David Heineman, Dirk Groeneveld, Faeze Brahman, Finbarr Timbers, Hamish Ivison, Jacob Morrison, Jake Poznanski, Kyle Lo, Luca Soldaini, Matt Jordan, Mayee Chen, Michael Noukhovitch, Nathan Lambert, Pete Walsh, Pradeep Dasigi, Robert Berry, Saumya Malik, Saurabh Shah, Scott Geng, Shane Arora, Shashank Gupta, Taira Anderson, Teng Xiao, Tyler Murray, Tyler Romero, Victoria Graf, Akari Asai, Akshita Bhagia, Alexander Wettig, Alisa Liu, Aman Rangapur, Chloe Anastasiades, Costa Huang, Dustin Schwenk, Harsh Trivedi, Ian Magnusson, Jaron Lochner, Jiacheng Liu, Lester James V. Miranda, Maarten Sap, Malia Morgan, Michael Schmitz, Michal Guerquin, Michael Wilson, Regan Huff, Ronan Le Bras, Rui Xin, Rulin Shao, Sam Skjonsberg, Shannon Zejiang Shen, Shuyue Stella Li, Tucker Wilde, Valentina Pyatkin, Will Merrill, Yapei Chang, Yuling Gu, Zhiyuan Zeng, Ashish Sabharwal, Luke Zettlemoyer, Pang Wei Koh, Ali Farhadi, Noah A. Smith, and Hannaneh Hajishirzi. Olmo 3, 2026.

[2] DeepSeek-AI, Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Daya Guo, Dejian Yang, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Han Bao, Hanwei Xu, Haocheng Wang, Haowei Zhang, Honghui Ding, Huajian Xin, Huazuo Gao, Hui Li, Hui Qu, J. L. Cai, Jian Liang, Jianzhong Guo, Jiaqi Ni, Jiashi Li, Jiawei Wang, Jin Chen, Jingchang Chen, Jingyang Yuan, Junjie Qiu, Junlong Li, Junxiao Song, Kai Dong, Kai Hu, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Lei Xu, Leyi Xia, Liang Zhao, Litong Wang, Liyue Zhang, Meng Li, Miaojun Wang, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Mingming Li, Ning Tian, Panpan Huang, Peiyi Wang, Peng Zhang, Qiancheng Wang, Qihao Zhu, Qinyu Chen, Qiushi Du, R. J. Chen, R. L. Jin, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, Runxin Xu, Ruoyu Zhang, Ruyi Chen, S. S. Li, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shaoqing Wu, Shengfeng Ye, Shengfeng Ye, Shirong Ma, Shiyu Wang, Shuang Zhou, Shuiping Yu, Shunfeng Zhou, Shuting Pan, T. Wang, Tao Yun, Tian Pei, Tianyu Sun, W. L. Xiao, Wangding Zeng, Wanjia Zhao, Wei An, Wen Liu, Wenfeng Liang, Wenjun Gao, Wenqin Yu, Wentao Zhang, X. Q. Li, Xiangyue Jin, Xianzu Wang, Xiao Bi, Xiaodong Liu, Xiaohan Wang, Xiaojin Shen, Xiaokang Chen, Xiaokang Zhang, Xiaosha Chen, Xiaotao Nie, Xiaowen Sun, Xiaoxiang Wang, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xingkai Yu, Xinnan Song, Xinxia Shan, Xinyi Zhou, Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, Y. K. Li, Y. Q. Wang, Y. X. Wei, Y. X. Zhu, Yang Zhang, Yanhong Xu, Yanhong Xu, Yanping Huang, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Li, Yaohui Wang, Yi Yu, Yi Zheng, Yichao Zhang, Yifan Shi, Yiliang Xiong, Ying He, Ying Tang, Yishi Piao, Yisong Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Yu Wu, Yuan Ou, Yuchen Zhu, Yuduan Wang, Yue Gong, Yuheng Zou, Yujia He, Yukun Zha, Yunfan Xiong, Yunxian Ma, Yuting Yan, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Z. F. Wu, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhen Huang, Zhen Zhang, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhibin Gou, Zhicheng Ma, Zhigang Yan, Zhihong Shao, Zhipeng Xu, Zhiyu Wu, Zhongyu Zhang, Zhuoshu Li, Zihui Gu, Zijia Zhu, Zijun Liu, Zilin Li, Ziwei Xie, Ziyang Song, Ziyi Gao, and Zizheng Pan. Deepseek-v3 technical report, 2025.

[3] 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, 2025.

[4] DeepSeek-AI, Aixin Liu, Bei Feng, Bin Wang, Bingxuan Wang, Bo Liu, Chenggang Zhao, Chengqi Dengr, Chong Ruan, Damai Dai, Daya Guo, Dejian Yang, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Hanwei Xu, Hao Yang, Haowei Zhang, Honghui Ding, Huajian Xin, Huazuo Gao, Hui Li, Hui Qu, J. L. Cai, Jian Liang, Jianzhong Guo, Jiaqi Ni, Jiashi Li, Jin Chen, Jingyang Yuan, Junjie Qiu, Junxiao Song, Kai Dong, Kaige Gao, Kang Guan, Lean Wang, Lecong Zhang, Lei Xu, Leyi Xia, Liang Zhao, Liyue Zhang, Meng Li, Miaojun Wang, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Mingming Li, Ning Tian, Panpan Huang, Peiyi Wang, Peng Zhang, Qihao Zhu, Qinyu Chen, Qiushi Du, R. J. Chen, R. L. Jin, Ruiqi Ge, Ruizhe Pan, Runxin Xu, Ruyi Chen, S. S. Li, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shaoqing Wu, Shengfeng Ye, Shirong Ma, Shiyu Wang, Shuang Zhou, Shuiping Yu, Shunfeng Zhou, Size Zheng, T. Wang, Tian Pei, Tian Yuan, Tianyu Sun, W. L. Xiao, Wangding Zeng, Wei An, Wen Liu, Wenfeng Liang, Wenjun Gao, Wentao Zhang, X. Q. Li, Xiangyue Jin, Xianzu Wang, Xiao Bi, Xiaodong Liu, Xiaohan Wang, Xiaojin Shen, Xiaokang Chen, Xiaosha Chen, Xiaotao Nie, Xiaowen Sun, Xiaoxiang Wang, Xin Liu, Xin Xie, Xingkai Yu, Xinnan Song, Xinyi Zhou, Xinyu Yang, Xuan Lu, Xuecheng Su, Y. Wu, Y. K. Li, Y. X. Wei, Y. X. Zhu, Yanhong Xu, Yanping Huang, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Li, Yaohui Wang, Yi Zheng, Yichao Zhang, Yiliang Xiong, Yilong Zhao, Ying He, Ying Tang, Yishi Piao, Yixin Dong, Yixuan Tan, Yiyuan Liu, Yongji Wang, Yongqiang Guo, Yuchen Zhu, Yuduan Wang, Yuheng Zou, Yukun Zha, Yunxian Ma, Yuting Yan, Yuxiang You, Yuxuan Liu, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhen Huang, Zhen Zhang, Zhenda Xie, Zhewen Hao, Zhihong Shao, Zhiniu Wen, Zhipeng Xu, Zhongyu Zhang, Zhuoshu Li, Zihan Wang, Zihui Gu, Zilin Li, and Ziwei Xie. Deepseek-v2: A strong, economical, and efficient mixture-of-experts language model, 2024.

[5] Damai Dai, Chengqi Deng, Chenggang Zhao, R.x. Xu, Huazuo Gao, Deli Chen, Jiashi Li, Wangding Zeng, Xingkai Yu, Y. Wu, Zhenda Xie, Y.k. Li, Panpan Huang, Fuli Luo, Chong Ruan, Zhifang Sui, and Wenfeng Liang. DeepSeekMoE: Towards ultimate expert specialization in mixture-of-experts language models. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar, editors, *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1280–1297, Bangkok, Thailand, August 2024. Association for Computational Linguistics.

[6] Sainbayar Sukhbaatar, Olga Golovneva, Vasu Sharma, Hu Xu, Xi Victoria Lin, Baptiste Rozière, Jacob Kahn, Daniel Li, Wen-tau Yih, Jason Weston, and Xian Li. Branch-train-MiX: Mixing expert LLMs into a mixture-of-experts LLM. In *Conference on Language Modeling*, 2024.

[7] Weijia Shi, Akshita Bhagia, Kevin Farhat, Niklas Muennighoff, Pete Walsh, Jacob Morrison, Dustin Schwenk, Shayne Longpre, Jake Poznanski, Allyson Ettinger, Daogao Liu, Margaret Li, Dirk Groeneveld, Mike Lewis, Wen-tau Yih, Luca Soldaini, Kyle Lo, Noah A. Smith, Luke Zettlemoyer, Pang Wei Koh, Hannaneh Hajishirzi, Ali Farhadi, and Sewon Min. FlexOlmo: Open language models for flexible data use. In *Proceedings of Advances in Neural Information Processing Systems*, 2025.

[8] Chenyang Song, Weilin Zhao, Xu Han, Chaojun Xiao, Yingfa Chen, Yuxuan Li, Zhiyuan Liu, and Maosong Sun. Blockffn: Towards end-side acceleration-friendly mixture-of-experts with chunk-level activation sparsity, 2025.

[9] Zeyu Shen and Peter Henderson. Temporally extended mixture-of-experts models. *arXiv preprint arXiv:2604.20156*, 2026.

[10] Suraiya Tairin, Shohaib Mahmud, Haiying Shen, and Anand Iyer. emoe: Task-aware memory efficient mixture-of-experts-based (moe) model inference. *arXiv preprint arXiv:2503.06823*, 2025.

[11] Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. In *Proceedings of the International Conference on Learning Representations*, 2017.

[12] 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. In *Proceedings of the International Conference on Learning Representations*, 2021.

[13] 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.

[14] Houyi Li, Ka Man Lo, Ziqi Wang, Zili Wang, Wenzhen Zheng, Shuigeng Zhou, Xiangyu Zhang, and Daxin Jiang. Can mixture-of-experts surpass dense llms under strictly equal resources? In *ICLR*, 2026.

[15] Xudong Lu, Qi Liu, Yuhui Xu, Aojun Zhou, Siyuan Huang, Bo Zhang, Junchi Yan, and Hongsheng Li. Not all experts are equal: Efficient expert pruning and skipping for mixture-of-experts large language models. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar, editors, *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6159–6172, Bangkok, Thailand, August 2024. Association for Computational Linguistics.

[16] Albert Q Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, et al. Mixtral of experts. *arXiv preprint arXiv:2401.04088*, 2024.

[17] Niklas Muennighoff, Luca Soldaini, Dirk Groeneveld, Kyle Lo, Jacob Morrison, Sewon Min, Weijia Shi, Pete Walsh, Oyvind Tafjord, Nathan Lambert, et al. Olmoe: Open mixture-of-experts language models. *Proceedings of the International Conference on Learning Representations*, 2025.

[18] Marmik Chaudhari, Idhant Gulati, Nishkal Hundia, Pranav Karra, and Shivam Raval. Moe lens – an expert is all you need, 2026.

[19] Xi Wang, Soufiane Hayou, and Eric Nalisnick. The myth of expert specialization in moes: Why routing reflects geometry, not necessarily domain expertise. *arXiv preprint arXiv:2604.09780*, 2026.

[20] jie hu, Jiahui Hou, and Xiangyang Li. Quantifying expert specialization for effective pruning in mixture-of-experts models, 2025.

[21] Zican Dong, Han Peng, Peiyu Liu, Wayne Xin Zhao, Dong Wu, Feng Xiao, and Zhifeng Wang. Domain-specific pruning of large mixture-of-experts models with few-shot demonstrations. In *Proceedings of Advances in Neural Information Processing Systems*, 2025.

[22] Tianyu Chen, Shaohan Huang, Yuan Xie, Binxing Jiao, Daxin Jiang, Haoyi Zhou, Jianxin Li, and Furu Wei. Task-specific expert pruning for sparse mixture-of-experts, 2022.

[23] Weizhong Huang, Yuxin Zhang, Xiawu Zheng, Fei Chao, Rongrong Ji, and Liujuan Cao. Discovering important experts for mixture-of-experts models pruning through a theoretical perspective. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2026.

[24] Xingyi Yang, Constantin Venhoff, Ashkan Khakzar, Christian Schroeder de Witt, Puneet K. Dokania, Adel Bibi, and Philip Torr. Mixture of experts made intrinsically interpretable. In *Proceedings of the International Conference of Machine Learning*, 2025. Poster.

[25] Jungwoo Park, Young Jin Ahn, Kee-Eung Kim, and Jaewoo Kang. Monet: Mixture of monosemantic experts for transformers. *Proceedings of the International Conference on Learning Representations*, 2025.

[26] Rizhen Hu, Yuan Cao, Boao Kong, Mou Sun, and Kun Yuan. Improving moe performance and efficiency with plug-and-play intra-layer specialization and cross-layer coupling losses, 2026.

[27] Hongcan Guo, Haolang Lu, Guoshun Nan, Bolun Chu, Jialin Zhuang, Yuan Yang, Wenhao Che, Xinye Cao, Sicong Leng, Qimei Cui, and Xudong Jiang. Advancing expert specialization for better MoE. In *Advances in Neural Information Processing Systems*, 2025.

[28] Margaret Li, Suchin Gururangan, Tim Dettmers, Mike Lewis, Tim Althoff, Noah A. Smith, and Luke Zettlemoyer. Branch-train-merge: Embarrassingly parallel training of expert language models, 2022.

[29] Yikang Shen, Zheyu Zhang, Tianyou Cao, Shawn Tan, Zhenfang Chen, and Chuang Gan. Moduleformer: Modularity emerges from mixture-of-experts, 2023.

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

[31] Zihan Qiu, Zeyu Huang, Bo Zheng, Kaiyue Wen, Zekun Wang, Rui Men, Ivan Titov, Dayiheng Liu, Jingren Zhou, and Junyang Lin. Demons in the detail: On implementing load balancing loss for training specialized mixture-of-expert models. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5005–5018, 2025.

[32] Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv:1803.05457v1*, 2018.

[33] Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. BoolQ: Exploring the surprising difficulty of natural yes/no questions. In Jill Burstein, Christy Doran, and Thamar Solorio, editors, *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2924–2936, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.

[34] Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. CommonsenseQA: A question answering challenge targeting commonsense knowledge. In Jill Burstein, Christy Doran, and Thamar Solorio, editors, *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4149–4158, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.

[35] Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. HellaSwag: Can a machine really finish your sentence? In Anna Korhonen, David Traum, and Lluís Màrquez, editors, *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4791–4800, Florence, Italy, July 2019. Association for Computational Linguistics.

[36] Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. Can a suit of armor conduct electricity? a new dataset for open book question answering. In *EMNLP*, 2018.

[37] Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. PIQA: Reasoning about physical commonsense in natural language. In *Association for the Advancement of Artificial Intelligence*, 2020.

[38] Maarten Sap, Hannah Rashkin, Derek Chen, Ronan Le Bras, and Yejin Choi. SocialIQA: Commonsense reasoning about social interactions. In *Proceedings of Empirical Methods in Natural Language Processing*, 2019.

[39] Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. WinoGrande: An adversarial winograd schema challenge at scale. In *Association for the Advancement of Artificial Intelligence*, 2020.

[40] Siva Reddy, Danqi Chen, and Christopher D. Manning. CoQA: A conversational question answering challenge. *Transactions of the Association for Computational Linguistics*, 7:249–266, 2019.

[41] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. SQuAD: 100,000+ questions for machine comprehension of text. In Jian Su, Kevin Duh, and Xavier Carreras, editors, *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, Austin, Texas, November 2016. Association for Computational Linguistics.

[42] Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew Kelcey, Ming-Wei Chang, Andrew M. Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. Natural questions: A benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:452–466, 2019.

[43] Mandar Joshi, Eunsol Choi, Daniel Weld, and Luke Zettlemoyer. TriviaQA: A large scale distantly supervised challenge dataset for reading comprehension. In Regina Barzilay and Min-Yen Kan, editors, *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1601–1611, Vancouver, Canada, July 2017. Association for Computational Linguistics.

[44] Dheeru Dua, Yizhong Wang, Pradeep Dasigi, Gabriel Stanovsky, Sameer Singh, and Matt Gardner. DROP: A reading comprehension benchmark requiring discrete reasoning over paragraphs. In Jill Burstein, Christy Doran, and Thamar Solorio, editors, *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2368–2378, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.

[45] Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. *Proceedings of the International Conference on Learning Representations (ICLR)*, 2021.

[46] Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, et al. Mmlu-pro: A more robust and challenging multi-task language understanding benchmark. *Advances in Neural Information Processing Systems*, 37:95266–95290, 2024.

[47] 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, 2021.

[48] DeepSeek-AI. Deepseek-v4: Towards highly efficient million-token context intelligence. Technical report, 2026. Available at Hugging Face.

[49] Kimi Team, Yifan Bai, Yiping Bao, Y. Charles, Cheng Chen, Guanduo Chen, Haiting Chen, Huarong Chen, Jiahao Chen, Ningxin Chen, Ruijue Chen, Yanru Chen, Yuankun Chen, Yutian Chen, Zhuofu Chen, Jialei Cui, Hao Ding, Mengnan Dong, Angang Du, Chenzhuang Du, Dikang Du, Yulun Du, Yu Fan, Yichen Feng, Kelin Fu, Bofei Gao, Chenxiao Gao, Hongcheng Gao, Peizhong Gao, Tong Gao, Yuyao Ge, Shangyi Geng, Qizheng Gu, Xinran Gu, Longyu Guan, Haiqing Guo, Jianhang Guo, Xiaoru Hao, Tianhong He, Weiran He, Wenyang He, Yunjia He, Chao Hong, Hao Hu, Yangyang Hu, Zhenxing Hu, Weixiao Huang, Zhiqi Huang, Zihao Huang, Tao Jiang, Zhejun Jiang, Xinyi Jin, Yongsheng Kang, Guokun Lai, Cheng Li, Fang Li, Haoyang Li, Ming Li, Wentao Li, Yang Li, Yanhao Li, Yiwei Li, Zhaowei Li, Zheming Li, Hongzhan Lin, Xiaohan Lin, Zongyu Lin, Chengyin Liu, Chenyu Liu, Hongzhang Liu, Jingyuan Liu, Junqi Liu, Liang Liu, Shaowei Liu, T. Y. Liu, Tianwei Liu, Weizhou Liu, Yangyang Liu, Yibo Liu, Yiping Liu, Yue Liu, Zhengying Liu, Enzhe Lu, Haoyu Lu, Lijun Lu, Yashuo Luo, Shengling Ma, Xinyu Ma, Yingwei Ma, Shaoguang Mao, Jie Mei, Xin Men, Yibo Miao, Siyuan Pan, Yebo Peng, Ruoyu Qin, Zeyu Qin, Bowen Qu, Zeyu Shang, Lidong Shi, Shengyuan Shi, Feifan Song, Jianlin Su, Zhengyuan Su, Lin Sui, Xinjie Sun, Flood Sung, Yunpeng Tai, Heyi Tang, Jiawen Tao, Qifeng Teng, Chaoran Tian, Chensi Wang, Dinglu Wang, Feng Wang, Hailong Wang, Haiming Wang, Jianzhou Wang, Jiaxing Wang, Jinhong Wang, Shengjie Wang, Shuyi Wang, Si Wang, Xinyuan Wang, Yao Wang, Yejie Wang, Yiqin Wang, Yuxin Wang, Yuzhi Wang, Zhaoji Wang, Zhengtao Wang, Zhengtao Wang, Zhexu Wang, Chu Wei, Qianqian Wei, Haoning Wu, Wenhao Wu, Xingzhe Wu, Yuxin Wu, Chenjun Xiao, Jin Xie, Xiaotong Xie, Weimin Xiong, Boyu Xu, Jinjing Xu, L. H. Xu, Lin Xu, Suting Xu, Weixin Xu, Xinran Xu,

Yangchuan Xu, Ziyao Xu, Jing Xu, Jing Xu, Junjie Yan, Yuzi Yan, Hao Yang, Xiaofei Yang, Yi Yang, Ying Yang, Zhen Yang, Zhilin Yang, Zonghan Yang, Haotian Yao, Xingcheng Yao, Wenjie Ye, Zhuorui Ye, Bohong Yin, Longhui Yu, Enming Yuan, Hongbang Yuan, Mengjie Yuan, Siyu Yuan, Haobing Zhan, Dehao Zhang, Hao Zhang, Wanlu Zhang, Xiaobin Zhang, Yadong Zhang, Yangkun Zhang, Yichi Zhang, Yizhi Zhang, Yongting Zhang, Yu Zhang, Yutao Zhang, Yutong Zhang, Zheng Zhang, Haotian Zhao, Yikai Zhao, Zijia Zhao, Huabin Zheng, Shaojie Zheng, Longguang Zhong, Jianren Zhou, Xinyu Zhou, Zaida Zhou, Jinguo Zhu, Zhen Zhu, Weiyu Zhuang, and Xinxing Zu. Kimi k2: Open agentic intelligence, 2026.

[50] Alexander Wettig, Kyle Lo, Sewon Min, Hannaneh Hajishirzi, Danqi Chen, and Luca Soldaini. Organize the web: Constructing domains enhances pre-training data curation. In *Proceedings of the International Conference of Machine Learning*, 2025.

A Architectural & Training Details and Ablations

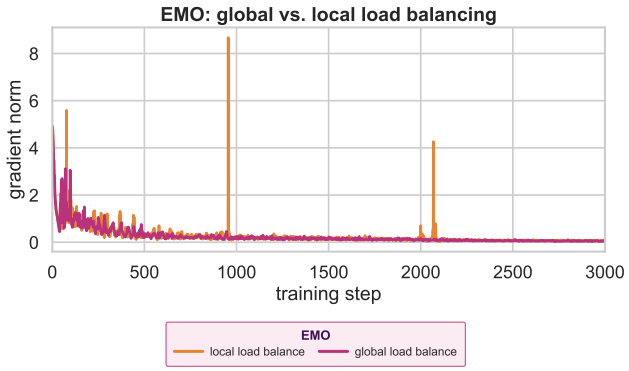

Figure 6: **Global vs Local Load Balancing** and its effects on training stability (Appendix A.1). Using global load balancing leads to more stable pre-training runs with less gradient norm spikes.

A.1 Load Balancing

We begin by making explicit how the simplified formulation of load balancing as described in Section 3.1 is implemented in practice.

Under standard implementations, the statistics $\bar{f}_i$ and $\bar{P}_i$ are computed independently within each data parallel group. Let there be n_p data parallel groups[3]. For each group j, let f_i^j denote the fraction of tokens in its micro-batch routed to expert i, and let P_i^j denote the average routing probability assigned to expert i across tokens in that micro-batch. The load balancing loss is then defined as

$$\mathcal{L}_{\text{LB}} = \frac{1}{n_p} \sum_{j=1}^{n_p} \left[n_r \sum_{i=1}^{n_r} f_i^j \cdot P_i^j \right],$$

which corresponds to computing the simplified objective separately within each group and averaging the results.

In this paper, we instead modify the load balancing objective to operate over aggregated routing statistics across data parallel groups. Specifically, we perform an all-reduce over data parallel groups to obtain the aggregated routing frequency

$$\bar{f}_i = \sum_{j=1}^{n_p} f_i^j,$$

while retaining the per-group routing probabilities P_i^j. The load balancing loss is then computed as

$$\mathcal{L}_{\text{LB}} = \frac{1}{n_p} \sum_{j=1}^{n_p} \left[n_r \sum_{i=1}^{n_r} \bar{f}_i \cdot P_i^j \right].$$

This formulation, proposed and used in Qwen 3 [3, 31], replaces local (micro-batch-level) estimates of $\bar{f}_i$ with aggregated statistics across data parallel groups, yielding a closer approximation to global routing behavior. In our setting, where training uses data parallelism only, this corresponds to computing load balancing over a larger set of sequences (i.e., the global batch up to gradient accumulation).

We find that global load balancing is critical for stable training in EMO. As shown in Figure 6, models trained with standard micro-batch-level load balancing exhibit unstable behavior, while global load balancing leads to consistent and reliable training dynamics. This difference arises because micro-batch-level load balancing conflicts with our document-level routing constraint by enforcing

[3] In this work, we pre-train with data parallelism only.

uniform expert usage within each micro-batch. In contrast, global load balancing aggregates routing statistics across data-parallel groups, allowing expert utilization to diversify across documents while preserving consistent routing within each document.

Configuration			Inference					Fine-tuning				
n_r	n_s	d	**8**	**16**	**32**	**64**	**128**	**8**	**16**	**32**	**64**	**128**
128	0	32	30.2	35.3	37.2	36.0	31.9	31.7	36.2	37.8	38.4	37.3
127	1	32	29.6	34.4	36.6	35.6	33.6	31.7	36.0	38.3	38.5	37.4
127	1	$U(8, 128)$	33.7	36.4	37.0	37.7	38.1	34.5	37.5	38.5	39.7	39.8

Table 2: **Ablating Shared Experts and Document Pool Size**. Evaluation on MMLU across different architectural configurations, comparing the effects of shared experts and dynamic expert subset sizes. We compare models without a shared expert ($n_s = 0$), with a shared expert and fixed expert subset size ($d = 32$), and with a shared expert and dynamic expert subset size ($U(8, 128)$). Using a shared expert improves performance, while dynamic expert-subset-size training enables more flexible expert selection across different budgets.

A.2 How to Choose d

We find that varying the size d during training is important for enabling flexible selective expert use usage at inference time. As shown in Table 2, models trained with a fixed d perform well at that specific expert subset size but degrade when evaluated at other subset sizes. In contrast, training with a distribution over d yields robust performance across a wide range of expert subset sizes.

A.3 Shared Experts

In Table 2, we find that incorporating shared experts improves performance in EMO. This is consistent with prior observations in DeepSeek-MoE [5].

A.4 Tuning LR and LB

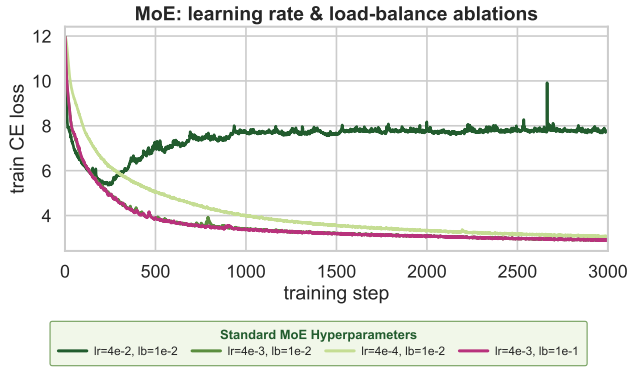

Figure 7: **Standard MoE LR and LB Ablations**. We ablate standard MoEs across learning rates and load balancing (Appendix A.4). We identify the best configuration as lr = $4e-3$ and lb = $1e-1$. For load balancing coefficient, we do not observe significant differences between $1e-1$ and $1e-2$, and choose the former because it had slightly higher training stability.

Due to limited compute budget, we perform ablations first on learning rate, then on load balancing in a sequential manner. Furthermore, we ablate the coefficients of each in increments of 10x.

Standard MoE Hyperparameter Ablations. In Figure 7, we initialize with the default lr = $4e-4$, lb = $1e-2$ configurations following OLMoE [17]. We first ablate the learning rate by increasing it to $4e-3$ and $4e-2$ while keeping the load balancing coefficient fixed. In our results, using a learning rate of $4e-3$ led to the best results. We then ablated the load balancing coefficient from $1e-2$ to

$1e-1$, which helped training stablility. We ended up selecting the hyperparameter configurations of lr = $4e-3$ and lb = $1e-1$.

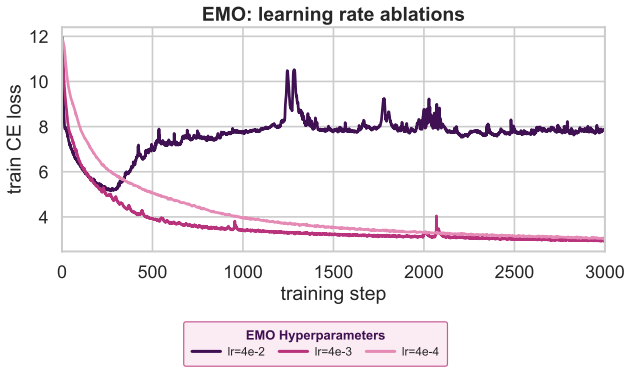

Figure 8: **EMO ablations over LR**. Ablations of EMO across learning rates. Due to limited compute resources, we fix lb = $1e-1$ and only ablate the learning rate, finding that lr = $4e-3$ offered the best training loss. Minor training loss spikes in lr = $4e-3$ were resolved by implementing load balancing over global batches.

EMO Hyperparameter Ablations. In Figure 8, we train EMO immediately using lb = $1e-1$ and ablated learning rate across lr = $4e-4$ (OLMoE default), $4e-3$, and $4e-2$. We noticed that lr = $4e-3$ led to the strongest performance by 3000 training steps and decided to move forwards with the final configuration of lr = $4e-3$ and lb = $1e-1$. We noticed small spikes in the loss during training for lr = $4e-3$, which were reduced when we used a global version of load balancing, see A.1.

A.5 Prenorm vs ReorderedNorm

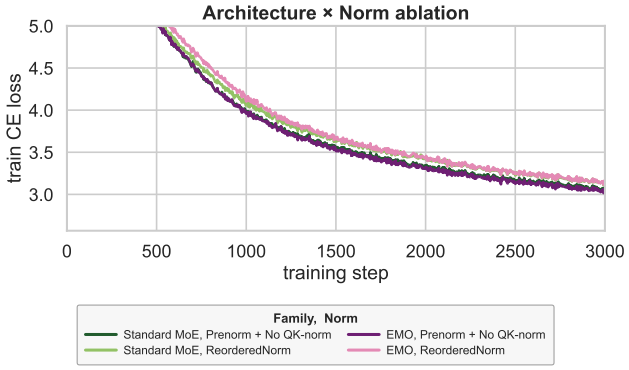

Figure 9: **Prenorm w. No QK Norm Ablations**. Ablations on ReorderedNorm from [17] versus Prenorm with removed QK-norm (Appendix A.5). On both standard MoEs and EMO, using Prenorm with removed QK-norm achieves lower loss than ReorderedNorm. These experiments were conducted without applying global load balancing, shared experts, and dynamic d.

Both EMO and the standard MoE in this work implemented Prenorm with removed QK-norm instead of the default ReorderedNorm implementation in OLMoE [17]. We ran ablations that consistently showed that our new implementation achieves a lower training loss, see Figure 9.

A.6 Measured GPU Memory Footprint

# Experts	Parameters (B)	GPU Weights (GiB)	Peak Forward Memory (GiB)
128 (full)	13.57	25.3	26.8
64	7.12	13.3	14.4
32	3.90	7.3	8.2
16	2.29	4.3	5.1
8	1.49	2.8	3.6

Table 3: **Measured GPU Memory Footprint.** Memory usage of EMO subsets with different numbers of retained experts. Peak forward memory is measured on a 4,096-token input.

We measure the memory footprint of domain-specific EMO subsets (Table 3). Retaining only 32 experts reduces the model weights from 25.3 to 7.3 GiB and peak forward-pass memory from 26.8 to 8.2 GiB. With eight experts, these quantities decrease to 2.8 and 3.6 GiB, respectively.

A.7 Training Throughput Overhead

Document-level routing adds little computational overhead to standard MoE training. After computing the usual token-level router probabilities, EMO aggregates probabilities within each document using a vectorized `scatter_add`, selects the top-d document experts, and masks and renormalizes each token's routing distribution. These operations require no additional router or expert forward passes. In our measurements, EMO and the standard MoE achieve 20.5k and 20.4k tokens/s, respectively; this difference is within normal measurement variability.

B Selective Expert Use Details

B.1 130B Token Experiments

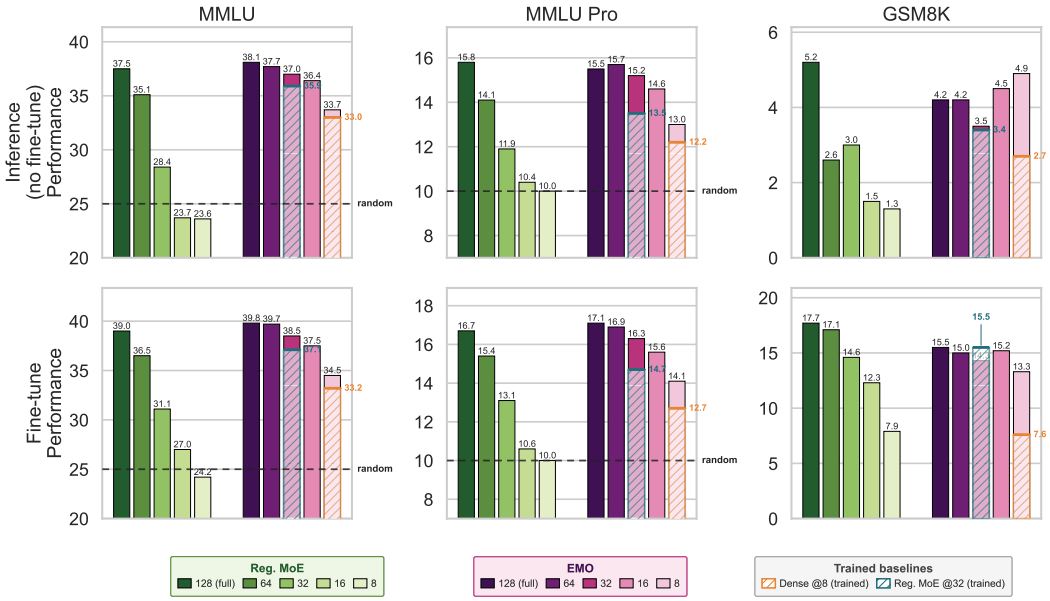

Figure 10: **Selective Expert Use** of MoEs trained on ***130B*** tokens (§5.2). We report performance before fine-tuning (*top*) and after fine-tuning (*bottom*). For MMLU and MMLU-Pro, each domain selects a corresponding expert subset as described in §4.2, and we report macro-averaged results across domains (17 for MMLU and 14 for MMLU-Pro). "Trained baseline @k" denotes a model trained from scratch with a parameter count matched to a k-expert subset. Across all tasks, the EMO 32-expert subset and 8-expert subset match or outperform the corresponding Reg. MoE @ 32 and Dense @ 8 trained models, with the 8-expert subset of EMO nearly doubling the performance of the Dense @ 8 on GSM8k, both before and after fine-tuning.

Following §5.2, we show the results for selective expert useacross MMLU, MMLU Pro, and GSM8K for models trained on 130B tokens. Additionally, we compare against memory-matched models traine from scratch, including a standard MoE with 32 experts and a dense model. A subset of these results are presented in Figure 1 (right). EMO match or outperform all baselines in selective expert use, pushing the pareto-frontier in memory-accuracy trade-off.

B.2 Ablations on expert subset initialization

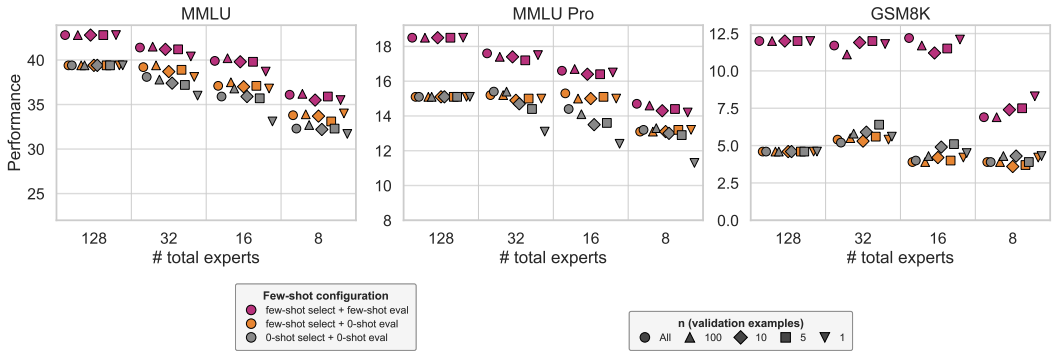

Figure 11: Effects of **validation data quantity** (n) and the presence of **few-shot demonstrations** (on both the data used to select experts and the actual test-set queries) on EMO expert subset performance, without any subsequent fine-tuning. By default MMLU/MMLU Pro have 5-shot demonstrations and GSM8K has 8-shot demonstrations. For GSM8K, we run with three random seeds for n=1, 5, and 10. EMO is sample-efficient in expert selection, and using few-shot demonstrations during both expert selection and evaluation brings large performance gains.

We now investigate how validation data affects expert selection. To study this, we vary three factors in Figure 11: (1) the number of validation examples used for expert selection, (2) validation set data format (few-shot vs. zero-shot prompts), and (3) the test-set data format (few-shot vs. zero-shot prompts).

EMO is Sample-efficient in Expert-selection. We first consider the default setting used in this work, when both validation and test set use few-shot prompts. In this setting, EMO shows little degradation as validation set size decreases—even down to a single example (red in Figure 11. We hypothesize this robustness arises because of the presence of few-shot demonstrations in each validation datapoint, which may provide sufficient token-level signals.

We then investigate using zero-shot prompts for validation and evaluation (gray), performance degrades slightly as the validation set size decreases, but the drop remains modest even with only 5 validation examples. Overall, this demonstrations that EMO remains effective even in highly data-constrained settings.

EMO Depends on Validation Data. While EMO is sample-efficient, the relationship between validation data and expert subset performance is nuanced and task-dependent. On GSM8K, for example, performance improves as the validation set size decreases. One possible explanation is that smaller validation sets produce more focused estimates of expert relevance, whereas aggregating across multiple examples can smooth these signals and yield less specialized expert subsets.

We also find that validation format sometimes matters independently of evaluation format: selecting experts using few-shot prompts can outperform zero-shot prompts, even when test examples are evaluated using zero-shot formats. This suggests that both the content and structure of validation data play a key role in shaping expert selection, an interaction we leave to future work.

B.3 Selective Expert Use on "Other" Category in MMLU and MMLU Pro

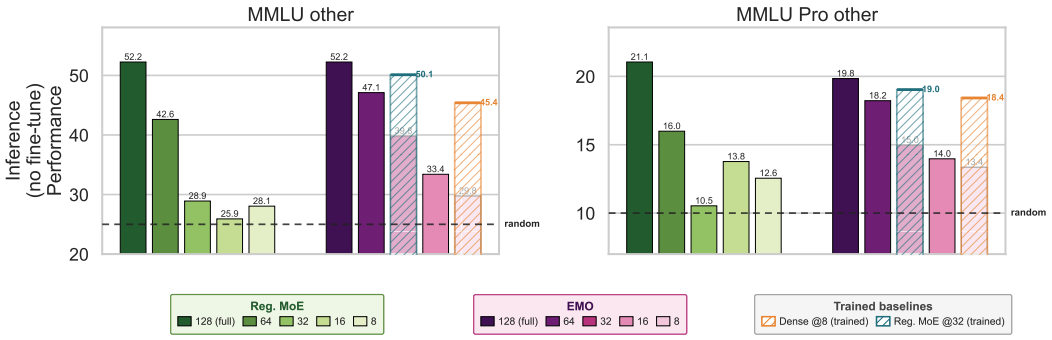

Figure 12: **Selective expert use on Other category** on EMO and standard MoEs trained on *130B* tokens (Appendix B.3). Results are without fine-tuning. "Trained baseline @k" denotes a model trained from scratch with a parameter count matched to a k-expert subset. On tasks that are general, EMO expert subsets of sizes 32 and 8 performs worse compared to "Reg MoE @32" and "Dense @8" that are trained from scratch.

When the deployment task is general (e.g MMLU other and MMLU Pro categories, which serve as a "catch-all" for MMLU subjects), EMO expert subsets of size 32 and 8 experts struggle to match the Reg MoE @ 32 and Dense @8 baseline models trained from scratch (Figure 12). We view this phenomenon as a property of modular models, and believe it provides concrete evidence that EMO works in selective expert use because it has groups of experts that have localize capabilities. When reporting aggregate metrics of MMLU and MMLU Pro, we intentionally exclude including the "other" category, as it is not an example of selective expert use on a specific task.

B.4 Annealing Standard MoEs to be Modular.

	# Experts	Inference			Fine-tuning		
		MMLU	**MMLU-Pro**	**GSM8K**	**MMLU**	**MMLU-Pro**	**GSM8K**
EMO-anneal	8	32.1	13.1	7.3	33.5	14.2	22.6
	16	35.4	14.8	9.9	37.3	16.3	25.3
	32	38.8	16.5	11.3	39.9	18.3	26.8
	64	41.3	17.4	12.8	42.7	19.5	27.7
	128 (trained)	42.3	18.2	13.0	43.7	20.1	27.2
EMO	8	36.1	14.7	6.9	37.3	15.6	23.3
	16	39.9	16.6	12.2	40.1	17.5	28.3
	32	41.4	17.6	11.7	41.7	19.5	27.5
	64	42.5	18.2	11.0	43.3	20.0	27.1
	128 (trained)	42.8	18.5	12.0	43.6	20.4	27.8

Table 4: **Inducing Modularity during Annealing Only.** We compare EMO, trained from scratch on 1T tokens and annealed on 50B tokens with document-level expert pool constraint (§3.2), against EMO-anneal, a model trained on 1T tokens as a standard MoE, but annealed on 50B tokens with the document-level expert pool constraint. Across most tasks and expert subset sizes, EMO improves over EMO-anneal, indicating that pre-training from scratch is important in realizing gains during selective expert use.

We investigate whether modularity requires applying the document-level expert pool objective (§3.2) throughout pre-training, or whether a standard MoE can be made modular after pre-training. To test this, we take a standard MoE pretrained on 1T tokens and anneal it using the document-level expert

pool objective instead of the standard MoE training objective. Denoted as EMO-anneal in Table 4, this model underperforms EMO on most benchmarks across most expert subset sizes. However, EMO-anneal still trains successfully and exhibits signs of modularity, suggesting that post-training applications of the document-level expert pool objective may be a promising direction for future work.

B.5 Generations from Selective Expert Use on GSM8K

We now demonstrate how small expert subsets of EMO is qualitatively better than that of regular MoEs. We give examples of GSM8K generations of EMO and Regular MoE trained on 1T tokens under selective expert use across different expert subset sizes. No finetuning was performed (expert subsets are evaluated zero-shot). We note that 8-expert subsets of EMO consistently produces coherent outputs while regular MoEs subsets deteriorate.

Q. Harry slept 9 hours last night. His friend James slept only 2/3 of what Harry slept. How many more hours did Harry sleep than James?
Gold: 3

Reg. MoE, 8-expert subset ✗

Olivia $2005 2005 2005 2005 2005 2005 2005 [+162 more] 200

EMO, 8-expert subset ✓

Harry slept 9 hours. His friend James slept 2/3 of 9. So 2/3 of 9 is 6. So the answer is 6 more hours. 9 - 6 = 3. So the answer is 3.

Reg. MoE, 32-expert subset ✗

Harry slept 9 hours last night. So the answer is 9 - 2 = 7. So the answer is 7.

EMO, 32-expert subset ✓

Harry slept 9 hours. James slept 2/3 of 9 hours. So James slept 9 * 2/3 = 6 hours. So Harry slept 9 - 6 = 3 hours more than James.

Reg. MoE, 128-expert subset ✓

Harry slept 9 hours. James slept 2/3 of that. So James slept 9 * 2/3 = 6 hours. So Harry slept 9 - 6 = 3 hours more than James.

EMO, 128-expert subset ✓

Harry slept 9 hours. James slept 2/3 of 9 hours. So James slept 9 * 2/3 = 6 hours. So Harry slept 9 - 6 = 3 hours more than James.

Q. Tom gets 4 car washes a month. If each car wash costs $15 how much does he pay in a year?
Gold: 720

Reg. MoE, 8-expert subset ✗

3 4 ? 1000 3 = 9, 5 5 4 + 4 = 9 so 1000 9 = 3 4 = 4 = 5 4 = 5 4 = 5

Q. Raymond and Samantha are cousins. Raymond was born 6 years before Samantha. Raymond had a son at the age of 23. If Samantha is now 31, how many years ago was Raymond's son born?
Gold: 14

Reg. MoE, 8-expert subset ✗

Olivia 3 5 = 4 9 4 Xe ? 5 4 3 Xe ? 5 5 4 = 9 so 20v 5 5 5 5 5 5 [+213 more]

EMO, 8-expert subset ✗

Raymond is 6 years older than Samantha. So Samantha is 31 - 6 = 25. Raymond is 6 years older than Samantha. So Raymond is 31 - 6 = 25 years old. So the answer is 25.

Reg. MoE, 32-expert subset ✗

Raymond was born 6 years before Samantha. So the answer is 6.

EMO, 32-expert subset ✗

Raymond was born 6 years before Samantha. So Raymond was born 6 years before Samantha was born. 6 years before Samantha was born was 6 years before 31. So Raymond's son was born 6 ...

Reg. MoE, 128-expert subset ✗

Raymond was born 6 years before Samantha. So he was 6 years older than Samantha. So he was 6 + 31 = 37 years old when his son was born. So the answer is 37.

EMO, 128-expert subset ✗

Raymond was born 6 years before Samantha. So he was born 6 years before Samantha was 31. So he was born 6 years before 31 = 25 years ago. So the answer is 25 years ago.

B.6 Overlap Across Domain-Specific Expert Subsets

	# Experts	Metric	Layer			
			0	5	10	15
EMO	8	Pairwise intersection	2.33	3.34	2.51	2.11
		Union	48	37	47	54
	32	Pairwise intersection	15.92	18.67	17.19	14.98
		Union	97	98	99	105
	64	Pairwise intersection	42.95	46.68	45.20	43.54
		Union	121	117	116	115
Reg. MoE	8	Pairwise intersection	4.79	3.53	4.61	4.33
		Union	24	28	26	25
	32	Pairwise intersection	28.46	21.54	18.26	17.71
		Union	48	73	84	89
	64	Pairwise intersection	55.93	49.60	45.61	46.16
		Union	99	111	114	115

Table 5: **Expert Overlap Across MMLU Categories.** Mean pairwise intersection and union of domain-specific expert sets over 17 MMLU categories, for different subset sizes and layers. Union counts are out of 128 total experts.

We quantify the overlap among expert subsets selected independently for the 17 MMLU categories. At each layer, we measure (i) the mean pairwise intersection of selected expert sets, averaged over the 136 category pairs, and (ii) their union, the total number of distinct experts selected by at least one category. Table 5 reports both measures for several expert budgets.

At small expert budgets, EMO subsets have less overlap across categories and collectively cover more distinct experts than regular MoE subsets.

B.7 Best-Effort Reproduction of ModuleFormer

Model	Loss ↓
ModuleFormer 1B/14B (our implementation)	2.70
Dense 1B	2.69
EMO 1B/14B	2.49
Standard MoE 1B/14B	2.48

Table 6: **Best-Effort ModuleFormer Reproduction.** Full-model loss for ModuleFormer and comparison models, trained on 130B tokens. This reproduction is specific to our best-effort implementation of ModuleFormer.

ModuleFormer is closely related to our goal of learning modular expert subsets. We made a best-effort reproduction of its mutual-information-based modularity objective using a modern MoE training setup. As shown in Table 6, this implementation yields a loss of 2.70 after training on 130B tokens, worse than both the matched standard MoE (2.48) and a 1B-parameter dense model (2.69). Note that the EMO comparison in table 6 is an earlier suboptimal EMO configuration with a fixed document pool ($d = 32$) and no shared expert. We would also like to emphasize that these results is specific to our best-effort attempts at implementation of ModuleFormer and does not rule out stronger ModuleFormer reproductions.

Given that our best-effort attempt to reproduce ModuleFormer does not achieve competitive full-model performance, we did not examine its selective-expert-use evaluations and used selective expert use results in the original paper as a point of reference.

C Token Clustering Details

EMO forms a Distinct Cluster for the First Token of Each Document. We observe a dedicated expert cluster consistently activated for the first token in each document (See clustering visualization). This behavior is intuitive, as the model processes the first token without any context. Interestingly, this cluster does not extend to the second token: in most cases after observing just one token, the model transitions into a specific expert activation pattern that often remains stable throughout the rest of the document. This suggests that EMO rapidly commits to a document-level routing pattern after minimal context, with the first token serving as a distinct initialization phase before more specialized processing begins.

EMO Expert Activation Patterns across Domains are Distinct and Matches Human Intuition. We next ask whether domain-level expert activation patterns reflect human-interpretable domain similarity. We find that EMO groups conceptually related domains while separating unrelated ones, a structure much less pronounced in standard MoEs.

To measure this, we use a random sample of 20 million documents from WebOrganizer [50], which assigns each document to one of 24 human-labeled domains. For each domain, we construct a domain-level expert activation vector by first averaging router activations across tokens within each document, and then averaging these document-level vectors across all documents assigned to that domain. We measure similarity between domains using cosine similarity over the resulting domain-level expert activation vectors.

As shown in Figure 13, EMO produces domain-level similarity patterns that better match semantic relationships between human-labeled domains. Related domains exhibit higher expert-activation similarity, while unrelated domains are more clearly separated. In contrast, the standard MoE shows more diffuse similarities across domains, suggesting that its expert activations are less aligned with human-interpretable domain structure.

Across both models, early layers show limited domain structure, while later layers exhibit clearer alignment with human-labeled domains. This suggests that domain-level expert specialization emerges progressively in deeper layers, potentially motivating future work on inducing modularity in a layer-wise manner.

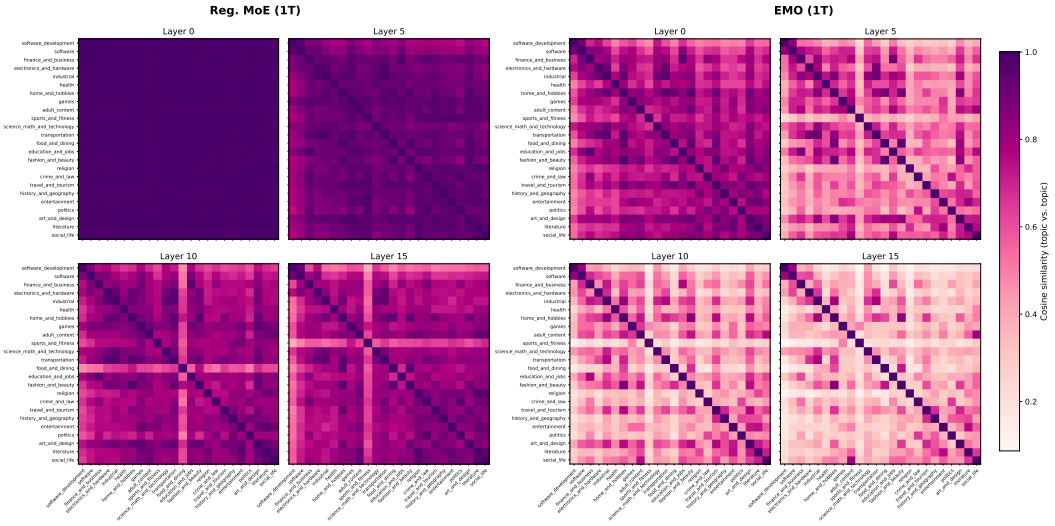

Figure 13: **Domain Similarity** of WebOrganizer documents on MoEs trained on *1T* tokens (§5.3). Expert utilization is highly similar (similarity > 0.6 for most domain pairs) in regular MoEs, while they are much more distinguishable (similarity < 0.4) in EMO, especially in later layers.

D Evaluation Details

MMLU and MMLU-Pro Setup. For MMLU and MMLU-Pro, we randomly sample 40% of examples for the validation/training set and use the remaining 60% for evaluation. For MMLU, we group the original 57 subjects into 17 broader categories following [45] to ensure sufficient examples per category to train and evaluate with. Expert selection is performed at the category level: all subjects within the same category are processed by a single group of experts. When reporting aggregate MMLU and MMLU-Pro results, we exclude the "other" category, which is discussed in Appendix B.3.

Evaluation and Finetuning Protocol. We provide the full list of evaluation categories and subjects in Table 7. For multiple-choice benchmarks, we score each answer choice by its log-likelihood and select the highest-scoring option, reporting raw accuracy ("acc-raw"). For generation benchmarks, we report recall on Gen5 and exact match on GSM8K.

During selective expert use, we use the same set of examples as both the validation data for expert selection and training data for finetuning. For tasks other than MMLU and MMLU-Pro, we merge the original train and validation splits and use the combined set for both expert selection and finetuning. When finetuning is performed, we mask the input portion of the prompt and optimize only over output tokens. Unless otherwise specified, finetuning follows standard Hugging Face settings with one epoch, batch size 32, and learning rate 5×10^{-5}.

Task	Shots	Train+Val	Test	Primary metric
MMLU-Pro (14 tasks)				
Math	5	601	750	acc/raw
Health	5	388	430	acc/raw
Physics	5	580	719	acc/raw
Business	5	376	413	acc/raw
Biology	5	347	370	acc/raw
Chemistry	5	513	619	acc/raw
Computer Science	5	224	186	acc/raw
Economics	5	398	446	acc/raw
Engineering	5	448	521	acc/raw
Philosophy	5	260	239	acc/raw
Other	5	430	494	acc/raw
History	5	213	168	acc/raw
Psychology	5	380	418	acc/raw
Law	5	501	600	acc/raw
MMLU (17 tasks)				
Biology	5	320	182	acc/raw
Business	5	308	176	acc/raw
Chemistry	5	211	122	acc/raw
Computer Science	5	289	165	acc/raw
Culture	5	232	134	acc/raw
Economics	5	525	298	acc/raw
Engineering	5	103	58	acc/raw
Geography	5	140	80	acc/raw
Health	5	1 162	659	acc/raw
History	5	658	373	acc/raw
Law	5	1 250	707	acc/raw
Math	5	752	427	acc/raw
Other	5	825	467	acc/raw
Philosophy	5	1 427	808	acc/raw
Physics	5	453	257	acc/raw
Politics	5	459	260	acc/raw
Psychology	5	823	463	acc/raw
MC9 (9 tasks)				
ARC-Easy	5	2 821	2 376	acc/raw
ARC-Challenge	5	1 418	1 172	acc/raw
BoolQ	5	9 427	3 270	acc/raw
HellaSwag	5	39 905	10 042	acc/raw
CSQA	5	9 741	1 221	acc/raw
OpenBookQA	5	5 457	500	acc/raw
PIQA	5	16 113	1 838	acc/raw
SocialIQA	5	33 410	1 954	acc/raw
WinoGrande	5	40 398	1 267	acc/raw
Gen5 (5 tasks)				
SQuAD	5	87 599	10 570	F1
CoQA	0	108 647	7 983	F1
NaturalQS	5	87 925	3 610	F1
TriviaQA	5	61 888	7 993	F1
DROP	5	77 409	9 536	F1
GSM8K (1 task)				
GSM8K	8	7 473	1 319	EM

Table 7: Evaluation tasks grouped by **MMLU-Pro** (14 categories), **MMLU** (17 categories), **MC9** (9 multiple-choice), **Gen5** (5 generation), and **GSM8K**. *Train+Val* reports the size of the dataset used during expert selection and finetuning.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract claims that EMO exhibits emerging properties of modularity, and can be used as expert subsets. The rest of the paper addresses this, with Figure 3 supporting claims such as an $\approx 1\%$ drop with 25% of experts and a $\approx 3\%$ drop with just 12.5% of experts.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: A limitation section exists, and discusses some fundamental assumptions (e.g that selective expert useassumes the model will be deployed for task-specific applications) and compute limitations that prevents error bars.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [N/A]

Justification: The paper has no formal theorems.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: The paper will fully open-source its development code. All data used is publicly available. We will also release model artifacts to aid in reproducibility. The entire paper pipeline is fully reproducible.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: The data, code, and all model artifacts will be completely open-sourced and will have detailed instructions on reproducing each figure and table.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: We specify hyperparameters (including their tuning) in the Appendix and the Experimental Setup section. For hyperparameters that are not critical to understand the results, we usually source them from prior work and will fully open-source our training and eval codes for maximum transparency and reproducibility.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No]

   Justification: Pretraining experiments are compute-intensive, and it is infeasible to run multiple runs with different seeds across billions of tokens. We instead capture evaluation noise through testing on a diverse range of downstream domains.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.

- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Compute resources stated in 4.1.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The research conforms in every respect.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [N/A]

Justification: This paper focuses on building language models that are modular. The paper mainly investigates whether subsets of experts can be independently used and whether expert utilization exhibit domain-level specialization patterns.

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The paper poses no such risk.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: Yes, data and models used were properly cited.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.

- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: Yes, there are new model artifacts and they will be properly documented on huggingface and github.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: There are no crowdsourcing experiments nor research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: Does not apply.

    Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: The core methods in this research does not involve the use of LLMs. Some analysis involved using LLMs for labeling clusters for visualization (not core to paper, mainly for human readability), which is also clearly stated in the paper.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.