THE MYTH OF EXPERT SPECIALIZATION IN MOES: WHY ROUTING REFLECTS GEOMETRY, NOT NECESSARILY DOMAIN EXPERTISE.

Anonymous authors
Paper under double-blind review

ABSTRACT

Mixture of Experts (MoEs) are now ubiquitous in large language models, yet the mechanisms behind their "expert specialization" remain poorly understood. We show that, since MoE routers are linear maps, hidden state similarity is both necessary and sufficient to explain expert usage similarity, where inputs from similar domains receive similar routing decisions, i.e. expert specialization. We confirm this at both token and sequence level across five pre-trained models. We additionally prove that load-balancing loss suppresses shared hidden state directions to maintain routing diversity, which might provide a theoretical explanation for specialization collapse under less diverse data, e.g. small batch. Despite this clean mechanistic account, we find that specialization patterns in pre-trained MoEs resist human interpretation: expert overlap between different models answering the same question is no higher than between entirely different questions ($\sim$60%); prompt-level routing does not predict rollout-level routing; and deeper layers exhibit near-identical expert activation across semantically unrelated inputs in certain models due to specific output structure. We conclude that, while the efficiency perspective of MoEs is well understood, understanding expert specialization is at least as hard as understanding LLM hidden state geometry, a long-standing open problem in the literature.

1 INTRODUCTION

Mixture of Experts (MoEs, Fedus et al., 2022) have become a standard architecture for large language models (LLMs). Compared with decoder-only LLMs (Radford et al., 2018), MoEs replace the dense feed-forward network (FFN) at each transformer block with a sparse mixture of expert blocks, where each token activates only a sparse subset of FFNs, keeping per-token compute manageable while scaling total model capacity.

An *expert specialization* behavior has been widely observed in pre-trained MoEs: sequences from different domains tend to activate markedly different expert subsets (e.g., Sec. 5 in Muennighoff et al. (2024), Fig. 9 in Liu et al. (2024)). Yet despite its importance as both a signal of meaningful feature learning and a prerequisite for parameter efficiency, its mechanistic origins remain largely unexplained. This paper asks: (1) *what mechanism produces expert specialization?* and (2) *to what extent are the resulting patterns humanly interpretable?*

The key observation is simple: the MoE router is a linear projection from token hidden states to expert logits, therefore tokens with similar hidden states must activate similar experts. We formalize this as Prop. 1, a tight data-geometry-aware upper bound on logit distance, verified empirically across five pre-trained MoEs (Fig. 1). Specialization is therefore a consequence of how the model organizes its representation space, not a property of the routing mechanism itself. We further show (Prop. 2) that when using a load-balancing loss to train MoEs, the router is forced to "suppress" shared directions in correlated hidden states, which might provide a theoretical explanation to the specialization collapse observed by Qiu et al. (2025) under small-batch training.

Despite this clean mechanistic picture, specialization patterns frequently defy human intuition. We demonstrate three cases: (1) different models solving the same math question activate expert patterns as dissimilar as those for entirely different questions ($\sim$60% overlap, Fig. 6); (2) semantically different

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

prompts activate nearly identical experts during prefilling, but diverge during generation (Fig. 7); (3) in deeper layers, sequences from unrelated domains can collapse to identical expert usage during prefilling (Figs.10, 11), yet the suppressed experts remain consequential once generation begins.

Together, these findings reframe expert specialization as a consequence of the geometry of LLM hidden states, rather than as a standalone interpretable phenomenon. Understanding it is therefore at least as difficult as understanding geometry, which remains an open question even in dense models (Lad et al., 2024; Skean et al., 2025). It may even be harder, since tokens with different hidden states can still activate similar experts.

From a practical perspective (Sec. 5), the geometric view suggests that: Depth scaling or zero-initialized attention mitigates routing imbalance from rank collapse at initialization (Fig. 8); During training, routers should be monitored jointly with the data they read; Lastly, to have actual modular, domain-aligned experts may require explicit training objectives.

2 Related work

Identification of domain-specialized experts Several works study expert specialization from an interpretability perspective, aiming to verify its existence and locate domain-specific experts in pre-trained MoEs. Chaudhari et al. (2025) introduces a LogitLens-based framework for analyzing MoE specialization patterns, Hu et al. (2026) proposes a causal-effect method to identify which experts are domain-specific while Wang et al. (2026b) propose a CommitteeAudit framework. Overall, these works share similar observations: There exists a compact, domain-invariant expert that captures the majority of routing mass across all domains, with some peripheral experts showing domain sensitivity. Our findings offer a mechanistic explanation for this pattern: shared committee experts likely reflect shared information in the hidden state space, potentially arising from shared syntax structure in the datasets they studied or simply from LLM internal computation. Our work differs from these in two key ways. First, rather than cataloguing *which* experts are domain-specific, we identify *why* expert usage may or may not correlate with domain / semantic at all, grounding the phenomenon in hidden state geometry. Second, these works rely on short benchmark inputs (MMLU: $<$100 tokens; GSM8K: $<$200 tokens), whereas we study much longer sequences, including multi-thousand-token reasoning chains, where the gap between prefilling and generation behavior becomes critical.

Applications of expert specialization Beyond interpretability, expert specialization has found practical applications in several domains. In multilingual settings, Chen et al. (2026) and Bandarkar et al. (2026) leverage routing patterns to identify language-specific experts and improve MoE performance on low-resource languages. From a different angle, recent work in the security community (Ding et al., 2025; Nuriyev & Kulp, 2026) independently arrives at a perspective closely related to ours: because expert usage patterns are a deterministic function of input hidden states, they constitute a side channel that can leak information about user queries, raising significant privacy concerns.

Expert specialization insights from pre-trained MoE reports Many open-sourced MoEs' reports have discussed expert specialization, typically by comparing expert activation patterns across corpora drawn from different domains, e.g., Appendix C in DeepSeek V3 (Liu et al., 2024), Sec 6 in dots.llm1 (Huo et al., 2025), and Figure 22 in OLMoE (Muennighoff et al., 2024). A notable exception is Mistral (Jiang et al., 2024), whose experts show little specialization, likely a consequence of sparse upcycling during training. It is worth noting that the goal of monitoring specialization during pre-training is primarily practical rather than interpretive: the goal is to prevent expert homogenization, i.e. the failure mode in which all experts converge to redundant representations, wasting model capacity. DeepSeek V3 (Liu et al., 2024) operationalizes this by studying the sensitivity of removing top experts, arguing that MoE with better specialization shows higher sensitivity to expert removal, indicating less redundancy.

Layer-wise behavior of hidden states in dense models A key implication of our work is that expert specialization can vary across depth because hidden state geometry evolves across layers. A growing body of work studies this evolution in dense transformers. Lad et al. (2024) identifies four functional stages of inference — detokenization, feature engineering, prediction ensembling, and residual sharpening — to explain non-uniform performance degradation under layer removal. Skean et al. (2025) shows that intermediate layers consistently produce richer representations than final layers

for downstream tasks. Acevedo et al. (2026) further finds that syntactic and semantic information are encoded differently across layers. These results translate directly to expert specialization: what a layer's hidden states encode determines which experts it activates, explaining why specialization patterns differ between early and late layers.

3 BACKGROUND: MOE

Decoder only LLM For a decoder only LLM with L layers, let $\boldsymbol{X}^\ell = \{\boldsymbol{x}_t^\ell\}_{t=1}^T \in \mathbb{R}^{T\times D}$ denote the input to layer ℓ, where T is the sequence length and D is the residual-stream dimension. A standard transformer block first applies self-attention and then an MLP sublayer, each wrapped with layer normalization and a residual connection:

$$\boldsymbol{H}^\ell = \boldsymbol{X}^\ell + \text{SelfAttention}(\text{LN}(\boldsymbol{X}^\ell)), \qquad \boldsymbol{Y}^\ell = \boldsymbol{H}^\ell + \text{FFN}(\text{LN}(\boldsymbol{H}^\ell)), \tag{1}$$

where we assume a pre-norm formulation, $\text{LN}(\cdot)$ denotes layernorm, $\boldsymbol{H}^\ell$ is the intermediate hidden representation after attention and $\boldsymbol{Y}^\ell$ is the output of layer ℓ.

Token-choice sparse MoE Recent models replace the dense MLP with a sparse mixture-of-experts (MoE) module, so that only a small subset of parameters is activated for each token while maintaining the total model capacity unchanged. Concretely, for a token representation $\boldsymbol{h}_t^\ell \in \mathbb{R}^D$, i.e. a row from $\text{LN}(\boldsymbol{H}^\ell)$, the MoE router computes routing logits

$$\boldsymbol{g}_t^\ell = P^\ell \boldsymbol{h}_t^\ell \in \mathbb{R}^E, \tag{2}$$

where $P^\ell \in \mathbb{R}^{E\times D}$ is the router weight matrix and E is the number of experts. Based on these logits, the router selects a small subset $\mathcal{S}_t^\ell \subseteq \{1,\dots,E\}$ of size k, typically the top-k experts with the largest logit value. Let $s_{t,e}^\ell = \sigma(\boldsymbol{g}_t^\ell)_e$ denote the routing weight assigned to expert e, usually obtained by setting $\sigma(\cdot)$ to be a softmax or sigmoid-and-normalize. The MoE output is then computed as

$$\text{MoE}(\boldsymbol{h}_t^\ell) = \Sigma_{e\in\mathcal{S}_t^\ell} s_{t,e}^\ell \,\text{FFN}_e^\ell(\boldsymbol{h}_t^\ell), \tag{3}$$

where $\text{FFN}_e^\ell(\cdot)$ is the feed-forward network corresponding to expert e. Thus, different tokens may be processed by different fixed-size subsets of experts, enabling the model to increase parameter count while keeping per-token computation relatively small. This token-choice routing formulation covers the most recent sparse MoE models. Variations of model structures usually occur in including extra shared experts, whether a jitter noise is added to $\boldsymbol{h}$ before routing, different sparsity ratios $\frac{k}{E}$, or the design of σ.

4 EXPERT "SPECIALIZATION" ARISES FROM HIDDEN STATE SIMILARITY

4.1 HIDDEN STATES, EXPERT USAGE, AND THE ROLE OF AUXILIARY LOSS

To understand the expert activation pattern, we begin by looking at the relationship between the tokens' hidden state ($\boldsymbol{h}$) and corresponding router outputs ($\boldsymbol{g}$). The idea is simple: since the router Eq. (2) is a linear projection, we expect tokens with similar $\boldsymbol{h}$ to show similar $\boldsymbol{g}$. To formalize this intuition, we measure the similarity between the hidden states using the Euclidean distance in a data-dependent space, and obtain a tight upperbound on how router outputs behave as a function of hidden states.

Proposition 1. *Let $H \in \mathbb{R}^{N\times D}$ be a data matrix whose rows are $\boldsymbol{h}_1^\top,\dots,\boldsymbol{h}_N^\top$, and let $H = U\Sigma V^\top$ be its singular value decomposition. Denote by $m = \min\{N, D\}$ the rank of H. Fix $r \in \{1,2,\dots,m\}$, and let $V_r \in \mathbb{R}^{D\times r}$ be the matrix formed by the top r right singular vectors of H. Define the orthogonal projector onto the corresponding principal subspace by $\Pi_r := V_r V_r^\top \in \mathbb{R}^{D\times D}$. Let $P \in \mathbb{R}^{E\times D}$ be the router weight matrix. Then, for any $i, j \in \{1,\dots,N\}$,*

$$\underbrace{\|P\boldsymbol{h}_i - P\boldsymbol{h}_j\|_2}_{\textit{Logits similarity}} \le \underbrace{\|P\Pi_r\|_2}_{\textit{Router-data alignment}} \underbrace{\|\boldsymbol{h}_i - \boldsymbol{h}_j\|_{\Pi_r}}_{\textit{Hidden state similarity}} + \underbrace{\|P(I-\Pi_r)(\boldsymbol{h}_i - \boldsymbol{h}_j)\|_2}_{\textit{Residual}}. \tag{4}$$

$\|\cdot\|_2$ denotes operator norm (maximum singular value) for matrix input, and $\|z\|_{\Pi_r} = \|\Pi_r z\|_2$ denotes the norm in the space corresponding to the projector Π_r. The full proof is provided in App. C, which first projects each $\boldsymbol{h}_i$ on Π_r and then applies the triangle inequality.

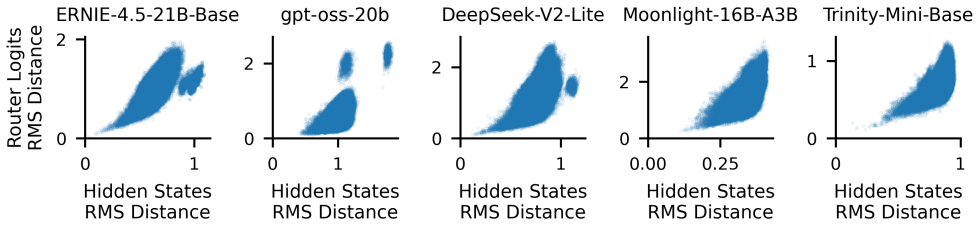


Figure 1: Empirical verification of Prop. 1. On the middle layer of 5 models, using 16 sequences from OpenWebText, we compare the RMS distance (Eq. (7)) of all token pairs' hidden state v.s. their router logits. **Bottom left corners:** Across all models, similar hidden states induce similar router logits, i.e. similar expert usage. **Middle right:** When the hidden states are different, the router logits can be similar or different. **Outlier dots** are from distances to the massive activation of the first token, i.e. attention sink (Sun et al., 2024).

Triangle shape of hidden states vs logit distances Under the assumption that the data variance is mostly concentrated in the top r directions (right-most columns in Fig. 18,19,20,21), the residual term is negligible compared to the first term. Proposition 1 suggests that when two tokens' hidden states are similar in the sense of norm $\|.\|_{\Pi_r}$, the upper bound collapses; their corresponding logits, i.e., the activated experts, must be similar. This is a much sharper bound than the trivial bound $\|P\|_2\|h_j - h_j\|_2$ (verified in Fig. 15, Appendix). However, when the hidden states differ, their router logits can be either similar or dissimilar. Fig. 1 empirically verifies this using OpenWebText on 5 different MoEs, where we observe the dots to demonstrate a *triangle* shape with small / large hidden state distance regions being the triangle's angle / side.

Impact of load-balancing From Proposition 1, if two hidden states are highly correlated (and there norms are roughly of the same order), router outputs will be similar. This could happen for several reasons. One structural reason is depth: as depth grows, correlation between hidden states increases between token hidden states, leading to increasing similarity as we will see below. In the next result, we show that load-balancing loss helps mitigate this problem by forcing the router to ignore directions shared by hidden states to achieve balanced expert usage.

Proposition 2 (Informal)**.** *We consider a simple hidden state correlation model where the hidden states are represented by*

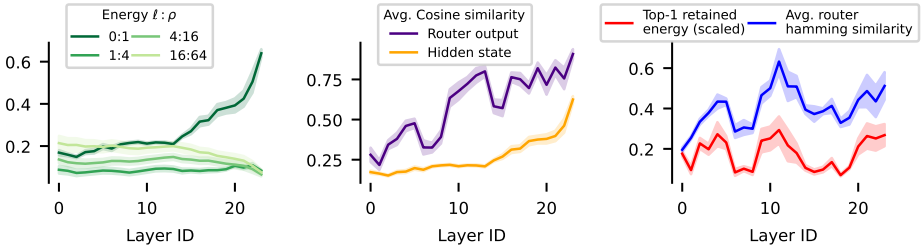
$$\boldsymbol{h}_i = \mu + \xi_i, \quad i = 1, \ldots N,$$

where $\mu \in \mathbb{R}^D$ is a shared direction and ξ_i are terms that differentiate each hidden state. We assume that ξ_i are centered and i.i.d. Consider a simplified load balancing loss for router $P \in \mathbb{R}^{E\times D}$

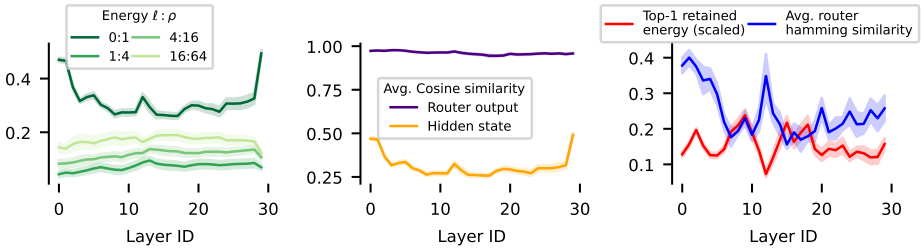
$$\mathcal{L}_{bal}(P) := \left\| \frac{1}{N}\sum_{i=1}^{N} \mathrm{softmax}\left(P\boldsymbol{h}_i\right) - \frac{1}{E}\mathbf{1} \right\|^2 .$$

Then, a near-minimizer of $\mathcal{L}_{bal}(P)$ must satisfy $P\mu \approx \overline{P\mu}\,\mathbf{1}$, where $\overline{P\mu} := \frac{1}{E}\mathbf{1}^\top P\mu$. That is, the shared direction does not affect the routing as softmax and top-K are invariant to constant shift.

The proof is presented in App. C.1, where we assume the small-logit regime under which softmax can be linearized. For sigmoid router, similar proof techniques and conclusions can also be applied (Lemma. 3). Proposition 2 shows that the load-balancing objective forces the router to suppress the shared direction μ, which is expected since the shared direction pushes the hidden states to use similar experts. Realistically, load-balancing is *added* to the pre-training objective, which means that we should only partially observe this effect. We believe this result could provide a theoretical explanation for the observation in Qiu et al. (2025), which suggests that when load-balancing is applied on small micro-batches (rather than the large global-batches), the resulting model loses any form of expert specialization, showing same expert usage patterns for inputs of different topics. Intuitively, small batches are not diverse enough, and therefore shared directions within small batches can significantly vary from one batch to another. This constant change in shared directions from one batch to another prevents the router from finding the informative directions. Similarly, Fig. 5 in Wang et al. (2024) showed that auxiliary loss-free approach works better with larger batch size.



(a) gpt-oss-20b, trained with AdamW and auxiliary load balancing loss.

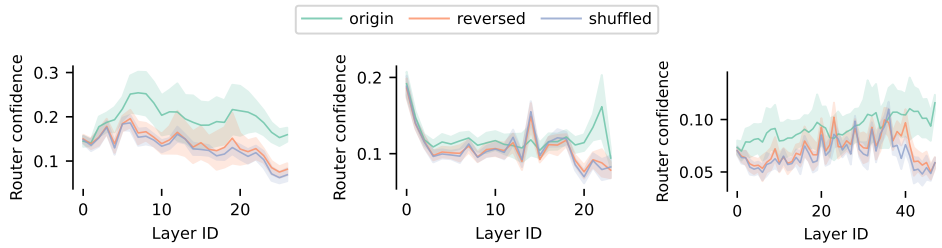


(b) Trinity-Mini-Base, trained with Muon and auxiliary-free load balancing.

Figure 2: On gpt-oss-20b and Trinity-Mini-Base, we studied tokens in 16 random sequences from OpenWebText. Lines / bars denote the mean / 1-std. over 16 sequences. Two models use different training configurations, resulting in distinct patterns across depth. **Top Left**: Token hidden states become more correlated with depth, indicated by stronger energy (Eq. (9)) from the shared direction (first right singular vector). **Top Middle**: After layer 11, the router output does not show a significant increase in cosine similarity, while hidden states do (Eq. (12)). **Top Right**: Looking at the router, it learns to suppress the shared direction to maintain diversity of router outputs: The trace of router's projection (averaged over rows) onto the shared direction (red line, Eq. (10)) aligns with the shape of router output similarity (blue lines, Eq. (13)) **Bottom left**: Under Muon trained model, token hidden states does not show increased correlation with depth. **Bottom middle and right**: When no load balancing loss is used, the router does not show evidence of suppressing shared direction.

Auxiliary loss pushes router to de-correlate hidden states The upper bound in Eq. (4) involves the term $\|P\Pi_r\|$ which measures router $\leftrightarrow$ hidden-state alignment. Intuitively, if this term vanishes, the logits would fail to capture the differences in the hidden states, effectively ignoring the data. Such a scenario could happen during MoE pre-training when the data is not diverse or the load balancing loss coefficient is set too large[1], which we empirically verified on a small-scale pre-training experiments (Fig. 12, App. C.2). We did not observe such extreme "dead router" patterns in open-sourced MoEs, as these models often show suboptimal performance and would not be released to the public. However, a related behavior can be observed in deeper layers of MoEs. In particular, LLM's hidden states across tokens inside a sequence tend to show increasing similarity with depth (left Fig. 2 for GPT-OSS, and Fig. 14 for dense models), which effectively *decreases* the diversity of expert usage. This is consistent with the *rank-collapse* phenomenon that occurs with depth (Schoenholz et al., 2017; Hayou et al., 2019; 2021; Noci et al., 2022). The router, driven by auxiliary load-balancing loss, would learn to suppress the shared direction (first singular vector) to maintain diverse router outputs. We confirm such suppression by comparing the router's retained energy on the shared direction with the expert usage similarity (right-most Fig. 2, or by looking at the router – singular-direction alignments, where the router shows less alignment with the first / shared direction compared with the second / third one (Middle columns, Fig. 18,20). In contrast, in models that are "auxiliary loss free", such as Moonlight 16B (Liu et al., 2025) or Trinity Mini (Singh et al., 2026), we do not observe such patterns (Fig. 17c,17d, 21, 19).

[1]Regularizing the router to orthogonal (Omi et al., 2025; Baidu-ERNIE-Team, 2025) cannot alleviate this, as the router can be an orthogonal matrix with all rows orthogonal to the data principal directions.



(a) ERNIE-4.5-21B (b) GPT-OSS-20B (c) Qwen-3-30B

Figure 3: We shuffled and reversed 16 sequences from OpenWebText to simulate OOD inputs. Router outputs show less confidence (measured by maximum softmax probability, Eq. (14)) on OOD inputs compared with the original ones (green line), due to the reduced alignment data router alignment (Eq. (4), verified in Fig. 25).

Router confidence detects OOD inputs Lastly, the router in pre-trained MoE is not a random matrix; it shows alignment with the data subspace during pre-training (left-most columns in Fig. 18,19,20,21). This suggests that when the input sequence is out-of-distribution (OOD), its hidden states could show less alignment with the router, reducing the upper bound and outputting more uniform logits. We simulate OOD by reversing and shuffling tokens in the sequences. Indeed, in Fig. 3 we see the confidence, measured by maximum softmax probability (Hendrycks & Gimpel, 2016), is lower for OOD inputs compared with ordinary inputs. In Fig. 25 in Appendix, we further verified the reduced alignment between router and reversed inputs compared with standard inputs.

4.2 Expert activation patterns are context dependen

In the previous section, we argued that for reasonably trained routers (non-collapsed), the token-wise expert usage is determined by its hidden state, whose value depends on: a) the token ID; b) the context. In earlier layers, we expect the hidden state to be mainly dependent on the token ID as it is closer to the input embedding layer. In deeper layers, strong correlation between hidden states for different tokens appears because depth and self-attention induce more interaction with the other tokens, and therefore the hidden states and the expert usage become more context dependent.

We empirically verify expert usage's context dependency by putting the same token(s) under different contexts on `gpt-oss-20b`. In Fig. 4 (top), we plot the activated experts for "apple" in daily context v.s. in coding context. Broadly, in first layer, there exists a high overlap across context (more purple blocks); but in later layers, the overlap starts to reduce. Fig. 4 (bottom) performs a similar study but for a 6-token sentence in daily v.s. medical context, where we look at expert activation frequency for the 6 tokens. Note that our observations go against the "Context-Independent Specialization" behavior observed by OpenMoe (Xue et al., 2024), which we believe arises from insufficient training.

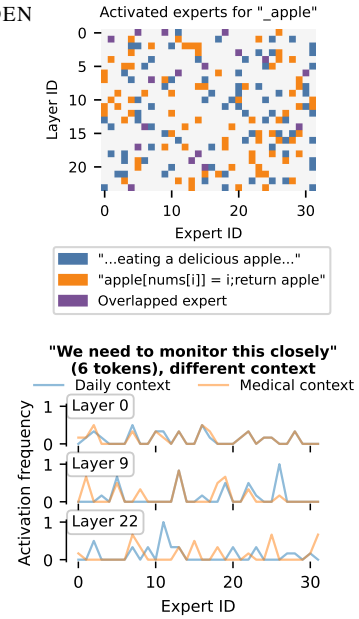


Figure 4: Same token (top) / sentence (bottom), different context, the expert usage is context dependent in deep layers.

4.3 Sequence and dataset-level expert specialization

The preceding analysis examined expert usage at the token level, showing that tokens with similar hidden states tend to route to similar experts. We now extend this view to the sequence level by comparing the cosine similarity of pooled hidden states (Eq. (19)) against the cosine similarity of sequence-level expert activation frequencies (Eq. (16)). Fig. 5 reports pairwise comparisons of 100 sequences from `OpenWebText` and 100 from `Nemotron-CC-Math` (Mahabadi et al., 2025),

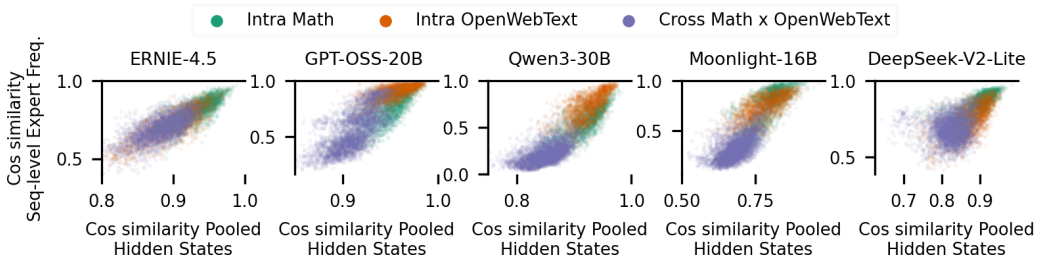


Figure 5: Pairwise comparisons of inputs drawn from Nemotron-CC-Math-v1, OpenWebText, and across these two datasets, plotting pooled hidden-state similarity (Eq. (19)) against expert-usage frequency similarity (Eq. (16)). Consistent with the token-level findings in Fig. 1, more similar sequences tend to activate more similar experts.

evaluated both within and across datasets at the middle layer of five models (full results appear in Fig. 22,24,23 in Appendix).

Two patterns emerge. First, sequence similarity strongly correlates with expert-usage similarity across all models. Second, some models exhibit a triangular point cloud: while similar sequences consistently activate similar experts, dissimilar sequences can activate either similar *or* different experts, a many-to-one relationship that already hints at interpretive difficulty. Notably, cross-dataset pairs cluster at lower hidden-state similarity than intra-dataset pairs, offering a straightforward explanation for the specialist behavior observed in prior work: inputs from different domains tend to have different hidden-state distributions, and therefore engage different sets of experts.

So far, the picture seems tractable: understand the hidden states, and expert usage follows. But this view holds only at a coarse, dataset level. As soon as we ask finer-grained questions, e.g. do two models solving the same problem use the same experts? Does the prompt predict the rollout's routing behaviour? The tractability breaks down. This leads us to a key tension:

Understanding expert specialization is at least as hard as understanding hidden-state geometry, and potentially harder: semantically unrelated inputs can activate the same experts, and inputs whose prompts route identically can diverge once generation begins.

4.4 When expert usage departs from semantics

The dataset-level correlation in Fig. 5 provides a simple picture: inputs from the same domain use the same experts. The four examples below test this reading at finer granularity. In each, routing is consistent with hidden-state geometry but not with the semantic notion of "domain" a human would apply. Full setups are in App. B.

Same problem, different generator, different experts. We pass reasoning traces from 60 models produced for 30 HMMT February 2025 problems (Balunović et al., 2025) through three MoEs (Qwen3-30B-A3B, ERNIE-4.5-21B-A3B, gpt-oss-20b) and measure expert overlap at top-p=0.8 (Eq. (21)). For a fixed MoE, traces of the same problem from two different generators share only $\sim$60% of their most-used experts. This is as dissimilar as traces of *different* problems from the same generator, and well below the $\sim$80–85% shared by two seeds of the same generator on the same problem (Fig. 6). The gap holds across all layers and all three MoEs (other p in Fig. 27). Therefore, which model wrote a solution moves routing nearly as much as which problem it solves.

Prompt-phase routing does not predict rollout routing. On the instruction-tuned Ling-mini (Team, 2025), we track expert-usage similarity for three prompt pairs from prefill through generation (Fig. 7): two concepts explained in the same style (GAN vs. VAE), one concept explained in plain vs. mathematical language (No-U-Turn Sampler, NUTS), and one AIME problem solved in Chinese vs. French. All three pairs route almost identically during prefill. Once generation begins, the GAN/VAE pair stays aligned while the other two diverge sharply. This suggests that the actual experts involved in answering a query are generally *not* visible in prompt-phase routing.

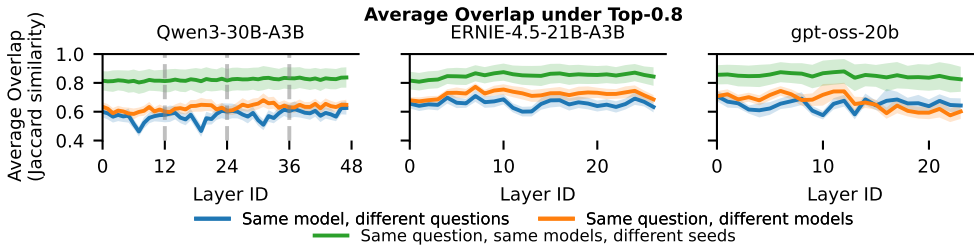


Figure 6: Expert overlap (Jaccard similarity at Top-p=0.8, Eq. (21)) between rollout pairs on 30 HMMT February 2025 questions, averaged over 4 seeds and three MoEs. Different models solving the same question (orange) share only ∼60% of their most-used experts, on par with the same model solving different questions (blue), and well below the same-question, same-model baseline (green). The gap is consistent across all layers.

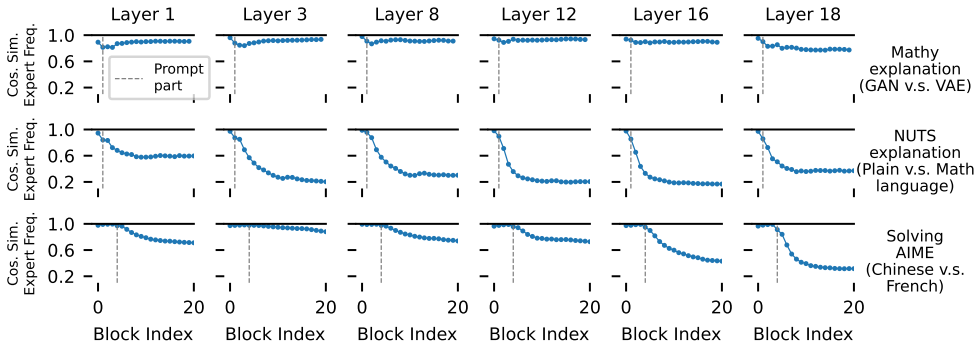


Figure 7: Expert-usage similarity between paired queries on `Ling-mini`, tracked from prompt through rollout. The dashed line marks the prompt boundary. Across all three query pairs, prompt-phase similarity is near-identical; once generation begins, similarity either holds or drops sharply depending on the query and is not predictable from the prompt alone.

Unrelated inputs, identical experts. Semantically unrelated inputs can share experts almost exactly. In the deeper layers of gpt-oss-20b, an HLE question (Phan et al., 2025) and a Wikipedia passage on the history of Wimbledon activate nearly identical experts during prefill (Fig. 10, appendix). Once generation begins, their routing diverges, and a pruning test shows the difference is functional. We keep only the 12 (of 32) experts most used on a single hand-written paragraph in the last two layers, which raises NLL by under 10% on prompts from *four unrelated domains*, but substantially on both on- and off-policy answer rollouts (Fig. 9). The effect is model-specific: it weakens without gpt-oss's chat template (top figure in Fig. 29) and is absent in the non-reasoning ERNIE-4.5. This suggests that the model's structured output format, not input semantics, can drive routing, especially in deep layers.

Changing geometry without changing meaning. Finally, we intervene on geometry directly. Repeating an input k times (k=1 to 12) leaves its content unchanged but strengthens the shared direction in its hidden states (Fig. 28). Across Ling-mini-base, gpt-oss-20b, and ERNIE-4.5, cross-domain expert-usage similarity rises with k toward the intra-domain baseline (Fig. 11). Semantics stay fixed; routing follows the geometry.

5 Practical Takeaways

Since routing inherits hidden-state geometry, several practical questions about MoEs reduce to questions about that geometry.

Reducing rank collapse provides better balancing at initialization Hidden state similarities are not necessarily induced by semantic similarity; in particular, at initialization the hidden states of tokens within a sentence tend to show increased correlation simply with depth regardless of input

embedding differences, known as rank collapse (Noci et al., 2022). Since the router is a random matrix at initialization, increasingly correlated representations lead to similar expert allocation and imbalance. In particular, at layer ℓ, tracking the residual update for the ith token,

$$\boldsymbol{h}_i^{\ell+1} \;=\; \boldsymbol{h}_i^{\ell} + \alpha_\ell \, \boldsymbol{\Delta}_i^{\ell}, \tag{5}$$

where $\boldsymbol{\Delta}_i^{\ell}$ is the update from the attention and MoE block output, and we introduce α_ℓ as a tunable depth-scaling parameter. The expected cross-token inner product evolves as

$$\mathbb{E}\left[\langle \boldsymbol{h}_i^{\ell+1}, \boldsymbol{h}_j^{\ell+1} \rangle\right] \;=\; \mathbb{E}\left[\langle \boldsymbol{h}_i^{\ell}, \boldsymbol{h}_j^{\ell} \rangle\right] + \alpha_\ell^2 \, \mathbb{E}\left[\langle \boldsymbol{\Delta}_i^{\ell}, \boldsymbol{\Delta}_j^{\ell} \rangle\right]. \tag{6}$$

Therefore as ℓ increases, the accumulation of $\langle \boldsymbol{\Delta}_i^{\ell}, \boldsymbol{\Delta}_j^{\ell} \rangle$ tends to make hidden states more correlated, see App. D for a detailed derivation.

To resolve the routing collapse, we need to reduce rank collapse: One solution is to utilize depth scaling, which sets $\alpha_\ell \propto 1/\sqrt{L}$ (Depth-μP, Yang et al. (2023)) to keep the total introduced correlation bounded; An alternative solution is to initialize the value or output projection as zero, which limits the magnitude of $\mathbb{E}\left[\langle \boldsymbol{\Delta}_i^{\ell}, \boldsymbol{\Delta}_j^{\ell} \rangle\right]$ by removing attention's contribution. The two recipes are also adopted in large-scale MoE pre-training: Trinity (Singh et al., 2026) initializes the residual stream's post-norm's scaling with $1/\sqrt{L}$, and the MAI-thinking (Microsoft AI Team, 2026) report explicitly states the importance of attention zero initialization for balanced routing. Notably, attention residual (Kimi Team et al., 2026b) used by recent Kimi models (Kimi Team et al., 2026a) can be viewed as setting α_ℓ adaptively, which we believe also alleviate the rank collapse problem.

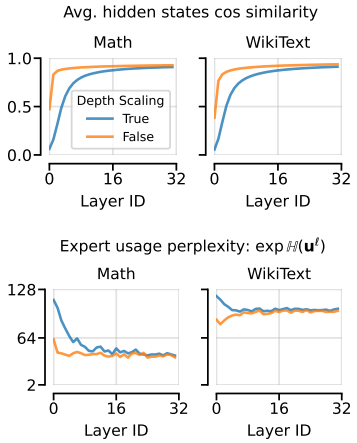


Figure 8: At initialization, depth scaling (blue, $\alpha_\ell = 0.2/\sqrt{L}$) delays the rise in cross-token hidden-state similarity (top) and yields more balanced expert usage in early layers (bottom; higher perplexity is more balanced). More diverse data (WikiText vs. Nemotron-CC-Math (Mahabadi et al., 2025)) also yields more balanced usage.

MoE monitoring requires jointly looking at the data and router During training, the hidden states and the router are not independent. The gradient tends to align them. Therefore, it is worth monitoring how the router is responsive to the data subspace, which is particularly useful when auxiliary loss is applied to the router, as we showed in Fig. 2 and in our diagnostic in Fig. 12, which also helps understand the robustness of the routing decision, i.e. whether the router is highly aligned with the off-principal directions in the data.

A real modular MoE requires explicit supervision objectives A fully modular MoE, where queries for different tasks activate *a disentangled set of experts*, can have many practical advantages, including adaptive expert loading at inference time or selective fine-tuning. However, our empirical results demonstrate that current pre-trained MoEs are not modular: While certain similarity patterns in the hidden states align with the input semantic (Fig. 5), the interpretable pattern lacks universality: There are many unpredictable and model-dependent exceptions (Sec. 4.4).

To actually enable the experts to be modular, or align with human bias, supervised objectives should be introduced, such as encouraging the same consistent set of experts to be used for processing the query (Shen & Henderson, 2026) or using document source as a domain label for encouraging disentangled expert activation (Li et al., 2026; Wang et al., 2026a).

6 Conclusion

Expert specialization is crucial for MoE's practical performance: First, it confirms that the model is performing proper feature learning, producing diverse hidden states for different inputs. Secondly, from the MoE perspective, expert specialization is a necessary condition for parameter efficiency: If the router fails to route different inputs to different experts, all experts will end up being duplicates of each other, effectively wasting parameters. However, our observation indicates that, unless an explicit objective is introduced, we should not expect the hidden state and router logit space to have a metric system aligned with human standards.

AI USE STATEMENT

In this work, we used generative AI tools for paper polishing and experiment implementation. We have not used generative AI tools for proposing hypotheses and refining research ideas. We have reviewed all AI-assisted work, where proof read the paper manually. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REFERENCES

Santiago Acevedo, Alessandro Laio, and Marco Baroni. Differential syntactic and semantic encoding in llms. *arXiv preprint arXiv:2601.04765*, 2026.

Baidu-ERNIE-Team. Ernie 4.5 technical report. https://ernie.baidu.com/blog/publication/ERNIE_Technical_Report.pdf, 2025.

Mislav Balunović, Jasper Dekoninck, Ivo Petrov, Nikola Jovanović, and Martin Vechev. Matharena: Evaluating llms on uncontaminated math competitions, February 2025. URL https://matharena.ai/.

Lucas Bandarkar, Chenyuan Yang, Mohsen Fayyaz, Junlin Hu, and Nanyun Peng. Multilingual routing in mixture-of-experts. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=ZoZR0x7tTD.

Marmik Chaudhari, Idhant Gulati, Nishkal Hundia, Pranav Karra, and Shivam Raval. Moe lens-an expert is all you need. In *Sparsity in LLMs (SLLM): Deep Dive into Mixture of Experts, Quantization, Hardware, and Inference*, 2025.

Yuxin Chen, Zhengzhou Cai, Xiangtian Ji, Weixiang Zhao, An Zhang, Xiang Wang, and Tat-Seng Chua. Understanding multilingualism in mixture-of-experts llms: Routing mechanism, expert specialization, and layerwise steering. *arXiv preprint arXiv:2601.14050*, 2026.

Ruyi Ding, Tianhong Xu, Xinyi Shen, Aidong Adam Ding, and Yunsi Fei. Moecho: Exploiting side-channel attacks to compromise user privacy in mixture-of-experts llms. In *Proceedings of the 2025 ACM SIGSAC Conference on Computer and Communications Security*, pp. 2159–2173, 2025.

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.

Soufiane Hayou, Arnaud Doucet, and Judith Rousseau. On the impact of the activation function on deep neural networks training, 2019. URL https://arxiv.org/abs/1902.06853.

Soufiane Hayou, Eugenio Clerico, Bobby He, George Deligiannidis, Arnaud Doucet, and Judith Rousseau. Stable resnet, 2021. URL https://arxiv.org/abs/2010.12859.

Dan Hendrycks and Kevin Gimpel. A baseline for detecting misclassified and out-of-distribution examples in neural networks. *arXiv preprint arXiv:1610.02136*, 2016.

Guimin Hu, Meng Li, Qiwei Peng, Lijie Hu, Boyan Xu, and Ruichu Cai. What gets activated: Uncovering domain and driver experts in moe language models. *arXiv preprint arXiv:2601.10159*, 2026.

Bi Huo, Bin Tu, Cheng Qin, Da Zheng, Debing Zhang, Dongjie Zhang, En Li, Fu Guo, Jian Yao, Jie Lou, et al. dots. llm1 technical report. *arXiv preprint arXiv:2506.05767*, 2025.

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.

Team Kimi Team, Tongtong Bai, Yifan Bai, Yiping Bao, Jianfeng Cai, Xinyuan Cai, Peizhou Cao, Yuxuan Cao, Ziwei Chai, Y Charles, et al. Kimi k3: Open frontier intelligence. *arXiv preprint arXiv:2607.24653*, 2026a.

Team Kimi Team, Guangyu Chen, Yu Zhang, Jianlin Su, Weixin Xu, Siyuan Pan, Yaoyu Wang, Yucheng Wang, Guanduo Chen, Bohong Yin, Yutian Chen, Junjie Yan, Ming Wei, Y. Zhang, Fanqing Meng, Chao Hong, Xiaotong Xie, Shaowei Liu, Enzhe Lu, Yunpeng Tai, Yanru Chen, Xin Men, Haiqing Guo, Y. Charles, Haoyu Lu, Lin Sui, Jinguo Zhu, Zaida Zhou, Weiran He, Weixiao Huang, Xinran Xu, Yuzhi Wang, Guokun Lai, Yulun Du, Yuxin Wu, Zhilin Yang, and Xinyu Zhou. Attention residuals, 2026b.

Vedang Lad, Jin Hwa Lee, Wes Gurnee, and Max Tegmark. The remarkable robustness of llms: Stages of inference? *arXiv preprint arXiv:2406.19384*, 2024.

Jiaang Li, Haibin Chen, Langming Liu, Yujin Yuan, Yadao Wang, Yizhen Zhang, Chengting Yu, Xin Tong, Weidong Zhang, Shilei Liu, et al. Expert divergence learning for moe-based language models. *arXiv preprint arXiv:2603.00054*, 2026.

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

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

Rabeeh Karimi Mahabadi, Sanjeev Satheesh, Shrimai Prabhumoye, Mostofa Patwary, Mohammad Shoeybi, and Bryan Catanzaro. Nemotron-cc-math: A 133 billion-token-scale high quality math pretraining dataset. 2025. URL https://arxiv.org/abs/2508.15096.

Team Microsoft AI Team. Mai-thinking-1: Building a hill-climbing machine. Technical report, Microsoft, June 2026. URL https://microsoft.ai/pdf/mai-thinking-1.pdf.

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. *arXiv preprint arXiv:2409.02060*, 2024.

Lorenzo Noci, Sotiris Anagnostidis, Luca Biggio, Antonio Orvieto, Sidak Pal Singh, and Aurelien Lucchi. Signal propagation in transformers: Theoretical perspectives and the role of rank collapse, 2022. URL https://arxiv.org/abs/2206.03126.

Amir Nuriyev and Gabriel Kulp. Expert selections in moe models reveal (almost) as much as text. *arXiv preprint arXiv:2602.04105*, 2026.

Nabil Omi, Siddhartha Sen, and Ali Farhadi. Load balancing mixture of experts with similarity preserving routers. *arXiv preprint arXiv:2506.14038*, 2025.

Long Phan, Alice Gatti, Ziwen Han, Nathaniel Li, Josephina Hu, Hugh Zhang, Chen Bo Calvin Zhang, Mohamed Shaaban, John Ling, Sean Shi, et al. Humanity's last exam. *arXiv preprint arXiv:2501.14249*, 2025.

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)*, pp. 5005–5018, 2025.

Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. Improving language understanding by generative pre-training. 2018.

Samuel S. Schoenholz, Justin Gilmer, Surya Ganguli, and Jascha Sohl-Dickstein. Deep information propagation, 2017. URL https://arxiv.org/abs/1611.01232.

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

Varun Singh, Lucas Krauss, Sami Jaghouar, Matej Sirovatka, Charles Goddard, Fares Obied, Jack Min Ong, Jannik Straube, Aria Harley, Conner Stewart, et al. Arcee trinity large technical report. *arXiv preprint arXiv:2602.17004*, 2026.

Oscar Skean, Md Rifat Arefin, Dan Zhao, Niket Patel, Jalal Naghiyev, Yann LeCun, and Ravid Shwartz-Ziv. Layer by layer: Uncovering hidden representations in language models. *arXiv preprint arXiv:2502.02013*, 2025.

Mingjie Sun, Xinlei Chen, J Zico Kolter, and Zhuang Liu. Massive activations in large language models. *arXiv preprint arXiv:2402.17762*, 2024.

Ling Team. Every flop counts: Scaling a 300b mixture-of-experts ling llm without premium gpus. *arXiv preprint arXiv:2503.05139*, 2025.

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

Ryan Wang, Akshita Bhagia, and Sewon Min. Emo: Pretraining mixture of experts for emergent modularity. *arXiv preprint arXiv:2605.06663*, 2026a.

Yan Wang, Yitao Xu, Nanhan Shen, Jinyan Su, Jimin Huang, and Zining Zhu. The illusion of specialization: Unveiling the domain-invariant" standing committee" in mixture-of-experts models. *arXiv preprint arXiv:2601.03425*, 2026b.

Fuzhao Xue, Zian Zheng, Yao Fu, Jinjie Ni, Zangwei Zheng, Wangchunshu Zhou, and Yang You. Openmoe: An early effort on open mixture-of-experts language models. *arXiv preprint arXiv:2402.01739*, 2024.

Greg Yang, Dingli Yu, Chen Zhu, and Soufiane Hayou. Tensor programs vi: Feature learning in infinite-depth neural networks, 2023. URL https://arxiv.org/abs/2310.02244.

A SUMMARIZATION OF ANALYSIS METRIC

RMS Distance In Fig. 1, we compared tokens' hidden states RMS distance with their logits' RMS distance. Given two vectors $\boldsymbol{x}_i, \boldsymbol{x}_j \in \mathbb{R}^D$, their RMS distance is defined as

$$\mathrm{RMSDist}(\boldsymbol{x}_i, \boldsymbol{x}_j) = \sqrt{\frac{1}{D}\sum_{d=1}^{D}\big(x_{i,d} - x_{j,d}\big)^2}, \tag{7}$$

where $x_{i,d}$ denotes the d-th coordinate of $\boldsymbol{x}_i$.

Directional energy and retained energy In the left most and right most columns of Fig. 2,17b,17c,17d, we studied the directional energy $\mathrm{Energy}_{\ell:\rho}$ and the retained energy for the first directions across layers, averaged over 16 sequences. For a T-token sequence, at a particular layer, let $H \in \mathbb{R}^{T\times D}$ be a data matrix whose rows are $\boldsymbol{h}_1^\top, \ldots, \boldsymbol{h}_T^\top$, i.e. the hidden state of the tth token and let

$$H = \boldsymbol{U}\mathrm{diag}\left(s_1, \ldots, s_D\right)\boldsymbol{V}^\top, \tag{8}$$

denotes its SVD. The directional energy is defined as

$$\mathrm{Energy}_{\ell:\rho} = \frac{\sum_{i=\ell}^{\rho} s_i^2}{\sum_{j=1}^{r} s_j^2}, \tag{9}$$

which measures how much of the total feature energy is captured by singular directions $\ell, \ldots, \rho$. Then, given a router P and the ith right singular direction $\boldsymbol{v}_i$, we define

$$\mathrm{RetainedEnergy}_i = \frac{1}{T}\sum_{t=1}^{T}\frac{\|P\boldsymbol{v}_i \odot \boldsymbol{m}_t\|_2^2}{\|P\|_F^2}. \tag{10}$$

where $\boldsymbol{m}_t \in \{0,1\}^E$ is a binary vector denoting whether token t usesd the eth vector. Since

$$\|P\boldsymbol{v}_i \odot \boldsymbol{m}_t\|_2^2 = \sum_{e=1}^{E}\boldsymbol{m}_t^e\langle r_e, \boldsymbol{v}_i\rangle^2, \tag{11}$$

this quantity measures how strongly the router preserves direction $\boldsymbol{v}_i$. A small value of $\mathrm{RetainedEnergy}_i$ for dominant shared directions (e.g., $\boldsymbol{v}_1$) suggests that the router suppresses those directions, which may help maintain balanced expert usage.

Average cosine similarity (inter-sequence) Given a sequence of T tokens, denote the hidden states / router logits at a particular layer as $\{\boldsymbol{x}_t\}_{t=1}^T$, average cosine similarity computes the average over the lower triangular of the kernel matrix, defined as:

$$\overline{\cos}(\{\boldsymbol{x}_t\}) = \frac{2}{T(T-1)}\sum_{i=2}^{T}\sum_{j=1}^{i-1}\frac{\boldsymbol{x}_i^\top\boldsymbol{x}_j}{\|\boldsymbol{x}_i\|\,\|\boldsymbol{x}_j\|}. \tag{12}$$

Router Hamming Similarity Given a sequence of T tokens, denote every token's expert usage at a particular layer as $\{\boldsymbol{m}_t\}_{t=1}^T$, where $\boldsymbol{m}_t \in \{0,1\}^E$ is a binary vector denoting whether token t used the e-th expert. The router Hamming similarity is computed as

$$\overline{\mathrm{Hamming}}(\{\boldsymbol{m}_t\}) = \frac{2}{T(T-1)}\sum_{i=2}^{T}\sum_{j=1}^{i-1}\frac{\boldsymbol{m}_i^\top\boldsymbol{m}_j}{E}, \tag{13}$$

where we average over all pairwise Hamming similarity. Compared with Eq. (12), the hamming similarity measures expert usage similarity more directly.

Router confidence In Fig. 3, we looked at the router confidence averaged over 16 sequences. For each sequence of T tokens, at a particular layer, the router confidence is defined as the average maximum softmax probability over all tokens:

$$\frac{1}{T}\sum_{t=1}^{T}\max_e \mathrm{softmax}(P\boldsymbol{h}_t)_e, \tag{14}$$

where P denotes the router and $\boldsymbol{h}_t$ denotes the hidden state.

Sequence-level expert frequency cosine similarity In Figs. 5, 7, 11, we compared two different sequences' expert usage frequency. In particular, for each sequence of T tokens, we count the frequency of token visits for each expert and normalize it into a probability vector:

$$\boldsymbol{p}^{(s)} = \frac{1}{TK} \sum_{t=1}^{T} \boldsymbol{m}_t^{(s)} \in \Delta^{E-1}, \tag{15}$$

where K denotes the number of activated experts and $\boldsymbol{m}_t^{(s)} \in \{0,1\}^E$ is the expert-usage vector of token t in sequence s, and Δ^{E-1} denotes the $(E-1)$-simplex. Then we compute the cosine similarity between two frequency vectors $\boldsymbol{p}^{(s)}$ and $\boldsymbol{p}^{(s')}$,

$$\mathrm{FreqSim}(s, s') = \frac{\boldsymbol{p}^{(s)\top} \boldsymbol{p}^{(s')}}{\|\boldsymbol{p}^{(s)}\| \, \|\boldsymbol{p}^{(s')}\|}, \tag{16}$$

as a measure of sequence-level expert usage difference.

Pooled hidden states cosine similarity In Fig. 5, the x-axis denotes the cosine similarity between two sequences' pooled hidden states. Given two sequences i and j of length L_i and L_j respectively, we first normalise each token's hidden state at layer ℓ:

$$\tilde{\boldsymbol{h}}_t^{\ell} = \frac{\boldsymbol{h}_t^{\ell}}{\|\boldsymbol{h}_t^{\ell}\|_2 + \varepsilon}, \tag{17}$$

and pool across tokens to obtain a sequence-level representation:

$$\bar{\boldsymbol{h}}_i^{\ell} = \frac{1}{L_i} \sum_{t=1}^{L_i} \tilde{\boldsymbol{h}}_t^{\ell}. \tag{18}$$

The pooled cosine similarity between sequences i and j is then:

$$d_{i,j}^{\ell} = \cos\left(\bar{\boldsymbol{h}}_i^{\ell}, \bar{\boldsymbol{h}}_j^{\ell}\right), \tag{19}$$

where $\varepsilon > 0$ is a small constant for numerical stability.

Overlap under top-P Given two sequences s and s', we look at the overlap ratio of their top-P expert sets. Using the empirical expert frequency vector $\boldsymbol{p}^{(s)}$ defined above, we define the top-P expert set of sequence s as the smallest set of experts (sorted by decreasing frequency) whose cumulative probability mass exceeds P:

$$\mathcal{E}_P^{(s)} = \arg\min_{S \subseteq [E]} |S| \quad \text{s.t.} \quad \sum_{e \in S} p_e^{(s)} \geq P, \quad p_e^{(s)} \geq p_{e'}^{(s)} \; \forall e \in S, e' \notin S, \tag{20}$$

and the overlap ratio between sequences s and s' is then their Jaccard similarity:

$$\mathrm{Overlap}_P(s, s') = \frac{|\mathcal{E}_P^{(s)} \cap \mathcal{E}_P^{(s')}|}{|\mathcal{E}_P^{(s)} \cup \mathcal{E}_P^{(s')}|}. \tag{21}$$

Compared with cosine similarity, Eq. (16), $\mathrm{Overlap}_P(s, s')$ measures expert usage similarity more directly, and the value is more interpretable.

B Details of the expert-usage examples

Here we provide setup details and supporting results for the four examples in Sec. 4.4

B.1 Same problem, different generator

Data. We use the reasoning traces released by MathArena (Balunović et al., 2025) for the 30 problems of HMMT February 2025, produced by 60 reasoning models with 4 samples (seeds) per problem. Each trace (problem and solution) is passed through each of three MoEs: Qwen3-30B-A3B, ERNIE-4.5-21B-A3B, and gpt-oss-20b. Throughout, "model" refers to the model that *generated* a trace. Overlaps are always computed between two traces processed by the same MoE.

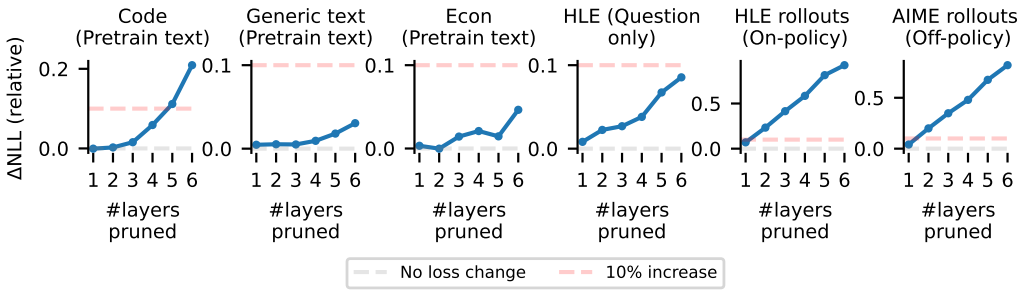


Figure 9: On gpt-oss-20b, we keep the 12 most-used experts (out of 32) in the last k layers, using usage frequency from a single hand-written paragraph. **First four columns:** Pruning up to 2 layers has a negligible effect (<10% NLL increase) on prompt-only inputs, indicating router collapse during prefilling. **Last two columns:** Adding completions makes the pruned experts consequential, with NLL increasing substantially for on- or off-policy rollout.

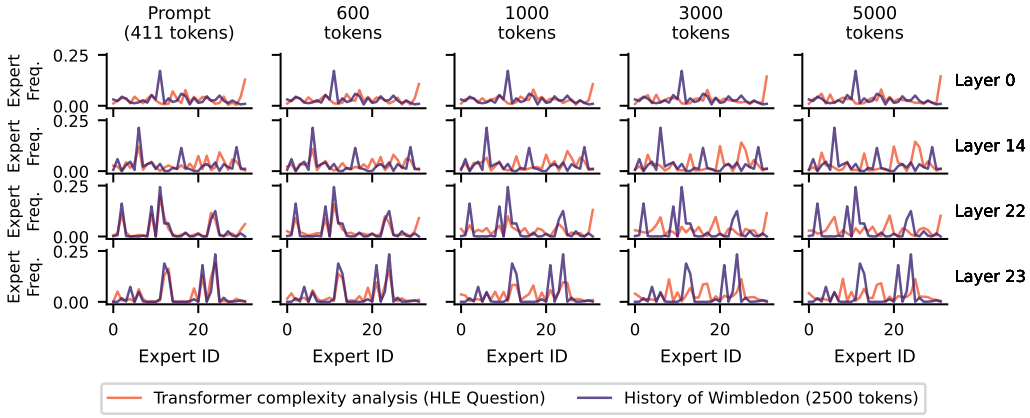


Figure 10: Per-expert activation frequencies for an HLE question (orange, 411 tokens) and a Wimbledon Wikipedia passage (purple, 2500 tokens) across four layers and five generation lengths for answering the HLE question. During prefilling, the two unrelated sequences activate nearly identical experts in deeper layers. As the generation proceeds, their activation profiles diverge, with the split emerging earlier in later layers.

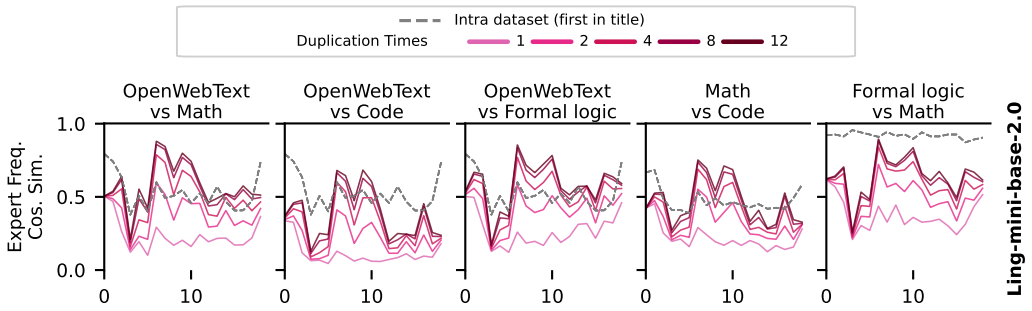
Comparisons. We index each trace by `(question, model, seed)` and form pairs that fix two of the three axes: (i) same model, different questions; (ii) same question, different models; (iii) same question and model, different seeds. For each trace and layer, we take its top-p expert set: the smallest set of experts accounting for a fraction p of its expert activations. We then compute the Jaccard similarity between the two traces' sets (Eq. (21)), averaged over pairs. Fig. 6 uses $p = 0.8$; Fig. 27 repeats the analysis for p from 0.6 to 0.99. For every $p \leq 0.9$, comparison (iii) sits well above (i) and (ii), which stay close to each other. At $p = 0.99$ the top-p sets contain most experts and the gap narrows; for ERNIE, the three curves nearly coincide.

B.2 Rollout content dominates expert usage

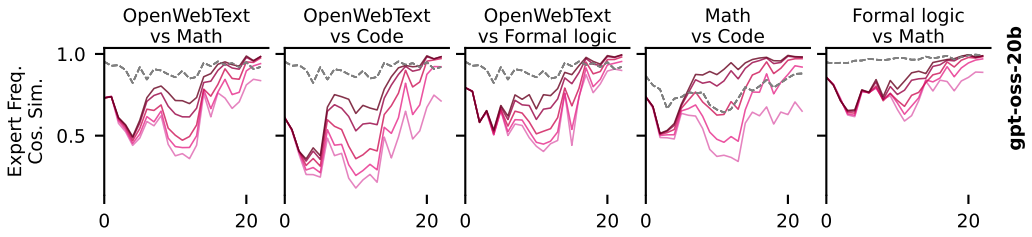
We prompt an instruction-tuned MoE (`Ling-mini` Team, 2025) with query pairs that are matched along one axis but differ along another: (i) different concepts explained in the same style; (ii) the same concept explained in plain v.s mathematical language; (iii) AIME problems prompted to be solved in different languages. We then track expert-usage similarity between paired queries' evolution from the prompt tokens through progressively longer rollouts.

B.3 Prefill-stage router collapse in gpt-oss

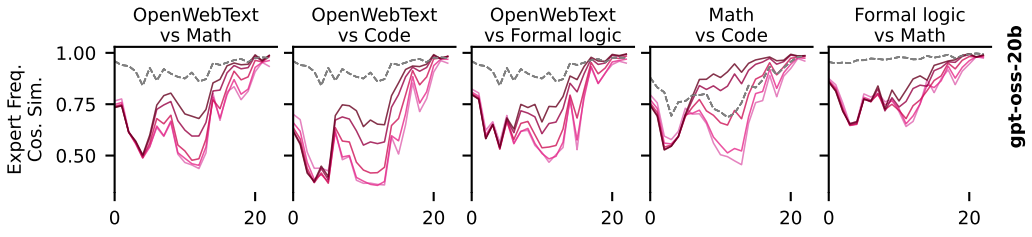
The reasoning model's output structure affects expert usage, which can lead to model-dependent counterintuitive observations: during prefilling, semantically unrelated sequences can activate exactly the same experts.



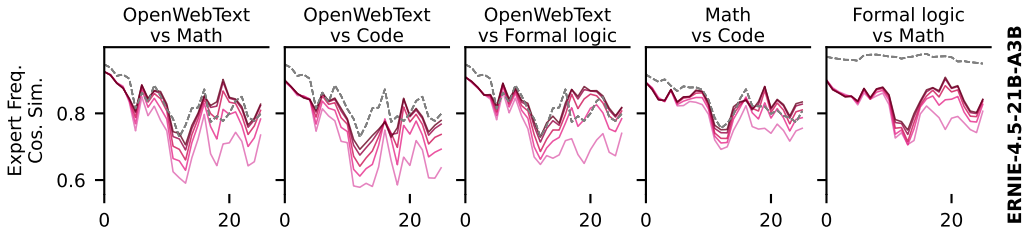
(a) Ling-mini-base, no chat template



(b) gpt-oss-20b, no chat template



(c) gpt-oss-20b, with chat template



(d) ERNIE-4.5-21B-A3B-Base, with chat template

Figure 11: Cross-domain expert-usage similarity under increasing input duplication ($1\times$ to $12\times$) across models; dashed grey shows intra-domain similarity. For models except gpt-oss with chat template (11c), hidden state similarity alone does not produce router collapse under no duplication. Duplication amplifies the dominant hidden-state direction, progressively collapsing cross-domain expert usage toward the intra-domain baseline.

As an illustrative example, the bottom left of Fig. 10 shows that in the later layers of `gpt-oss-20b`, a question from the Humanity's Last Exam (HLE, Phan et al., 2025) and a passage on the history of Wimbledon engage an identical set of experts. To probe how widespread this "router collapse" is, we adopt an extreme pruning protocol: using a single hand-written paragraph as the validation data, we retain only the top-12 experts at the bottom layers and examine NLL changes across a variety of corpora. The first four columns of Fig. 9 shows that pruning experts in the last two layers has a negligible effect ($<10\%$ NLL increase) for prompt-only input across four different topics.

The picture changes once the model begins generating. Bottom right in Fig. 10 shows that expert-usage patterns indistinguishable during prefilling begin to diverge as the reasoning chain unfolds; and the last two columns of Fig. 9 confirm that the pruned experts are highly consequential for the model output's (conditional) likelihoods, both on- and off-policy. The hidden state similarity across depth (Fig. 16a, appendix) further verifies the phase change.

We hypothesize that router collapse during prefilling is a property arising from gpt-oss's structured reasoning output: The input-invariant initial decoding tokens (examples shown in App. E.1) are largely controlled by the final-layer hidden states, which converge to similar representations across diverse prompts, causing the router to select the same experts. Consistent with this, we do not observe a comparable collapse in a non-reasoning instruction-tuned model (`ERNIE-4.5-21B-A3B-PT`; Fig. 30). Removing the chat template also reduces the collapse (lightest line, Fig. 11b and leftmost column, top figure in Fig. 29), though expert-usage similarity still increases with depth in those models, likely due to the hidden state correlation discussed in the previous section. Finally, Fig. 11 shows that this collapse is not inevitable: for models that do not exhibit it naturally, prefill-stage hidden state similarity alone is insufficient to produce router collapse. However, duplicating the input sequences (excluding the chat template) amplifies the shared direction in the hidden states enough to force expert-usage convergence even across semantically unrelated inputs (Fig. 11).

Taken together, these three examples illustrate why deeper layers can behave differently from shallower ones: while early-layer hidden states reflect the token distribution and thus the input's semantics, later-layer representations may instead encode the model's anticipated output, rendering the mapping from surface content to expert usage opaque.

C Expert properties arise from the embeddings

Given two hidden states $x_i, x_j \in \mathbb{R}^D$ and the router $P \in \mathbb{R}^{E\times D}$.

Naively, we have

$$\|Px_i - Px_j\|_2 \le \|P\|_2 \|x_i - x_j\|_2 \tag{22}$$

this bound is too loose, and it does not take into account data geometry. We can introduce an orthogonal base $Q \in \mathbb{R}^{D\times D}$, and we look at

$$\left\|(PQ^\top)Qx_i - (PQ^\top)Qx_j\right\|_2 \le \|PQ\|_2 \|Q(x_i - x_j)\|_2 \tag{23}$$

now if $x_i - x_j$ lies on span(Q), we have $\|Q(x_i - x_j)\|_2 = \|x_i - x_j\|_2$.

To construct such Q, we could consider having Q as the right singular matrix of $\boldsymbol{X}$. This is formalized in the next lemma.

Lemma 1 (Data-aware Lipschitz bound via principal subspace projection)**.** *Let $X \in \mathbb{R}^{N\times D}$ be a data matrix whose rows are $x_1^\top, \dots, x_N^\top$, and let*

$$X = U\Sigma V^\top$$

be its singular value decomposition. Denote by $m = \min\{N, D\}$ the rank of X. Fix $r \in \{1, 2, \dots, m\}$, and let

$$V_r \in \mathbb{R}^{D\times r}$$

be the matrix formed by the top r right singular vectors of X. Define the orthogonal projector onto the corresponding principal subspace by

$$\Pi_r := V_r V_r^\top \in \mathbb{R}^{D\times D}.$$

Let $P \in \mathbb{R}^{E\times D}$ be the router weight matrix. Then, for any $i, j \in \{1, \dots, N\}$,

$$\|Px_i - Px_j\|_2 \le \|P\Pi_r\|_2 \|\Pi_r(x_i - x_j)\|_2 + \|P(I - \Pi_r)(x_i - x_j)\|_2.$$

In particular,

$$\|Px_i - Px_j\|_2 \le \|P\Pi_r\|_2 \|x_i - x_j\|_2 + \|P(I - \Pi_r)\|_2 \|(I - \Pi_r)(x_i - x_j)\|_2,$$

and

$$\|(I - \Pi_r)(x_i - x_j)\|_2^2 = \sum_{k=r+1}^{D} \langle x_i - x_j, v_k\rangle^2.$$

Proof. Fix i, j, and define

$$\Delta_{ij} := x_i - x_j \in \mathbb{R}^D.$$

Since Π_r is the orthogonal projector onto $\mathrm{span}(V_r)$, we have the decomposition

$$\Delta_{ij} = \Pi_r \Delta_{ij} + (I - \Pi_r)\Delta_{ij}.$$

Applying P to both sides yields

$$P\Delta_{ij} = P\Pi_r \Delta_{ij} + P(I - \Pi_r)\Delta_{ij}.$$

By the triangle inequality,

$$\|P\Delta_{ij}\|_2 \le \|P\Pi_r \Delta_{ij}\|_2 + \|P(I - \Pi_r)\Delta_{ij}\|_2.$$

Using the operator norm bound on the first term,

$$\|P\Pi_r \Delta_{ij}\|_2 \le \|P\Pi_r\|_2 \, \|\Delta_{ij}\|_2,$$

we obtain

$$\|P\Delta_{ij}\|_2 \le \|P\Pi_r\|_2 \, \|\Delta_{ij}\|_2 + \|P(I - \Pi_r)\Delta_{ij}\|_2.$$

Recalling that $\Delta_{ij} = x_i - x_j$, this proves

$$\|Px_i - Px_j\|_2 \le \|P\Pi_r\|_2 \, \|x_i - x_j\|_2 + \|P(I - \Pi_r)(x_i - x_j)\|_2.$$

For the second bound, apply again the operator norm inequality:

$$\|P(I - \Pi_r)(x_i - x_j)\|_2 \le \|P(I - \Pi_r)\|_2 \, \|(I - \Pi_r)(x_i - x_j)\|_2,$$

where we have used the fact that $I - \Pi_r = (I - \Pi_r)^2$.

Now let $v_1, \ldots, v_D$ denote the columns of V. Since $\Pi_r = \sum_{k=1}^{r} v_k v_k^\top$, we have

$$I - \Pi_r = \sum_{k=r+1}^{D} v_k v_k^\top.$$

Therefore,

$$(I - \Pi_r)(x_i - x_j) = \sum_{k=r+1}^{D} \langle x_i - x_j, v_k \rangle v_k.$$

Because $v_{r+1}, \ldots, v_D$ are orthonormal, it follows from Pythagoras' theorem that

$$\|(I - \Pi_r)(x_i - x_j)\|_2^2 = \sum_{k=r+1}^{D} \langle x_i - x_j, v_k \rangle^2.$$

This concludes the proof. □

The result of Lemma 1 suggests that when data variance is mostly concentrated on few principal directions, the lipschitz constant $\|P\Pi_r\|_2$ is much sharper than the naive constant $\|P\|_2$. Intuitively, this is the case when the hidden states exhibit significant correlation.

C.1 Shared directions become routing-insignificant under load balancing.

Consider a router matrix $P \in \mathbb{R}^{E \times D}$ with logits

$$g(x) = Px \in \mathbb{R}^E, \qquad p(x) = \mathrm{softmax}(Px) \in \mathbb{R}^E.$$

Since $\mathrm{softmax}(z + \alpha \mathbf{1}) = \mathrm{softmax}(z)$ for every $z \in \mathbb{R}^E$, $\alpha \in \mathbb{R}$, only the component of Px orthogonal to $\mathbf{1}$ matters for routing. Let

$$\Pi_E := I_E - \frac{1}{E}\mathbf{1}\mathbf{1}^\top$$

denote the orthogonal projector onto $\mathbf{1}^\perp$. Then the *effective* routing signal is $\Pi_E P x$.

We study the simple correlated data model

$$x_i = \mu + \xi_i, \qquad i = 1, \dots, N,$$

where $\mu \in \mathbb{R}^D$ is a shared direction and $\frac{1}{N}\sum_{i=1}^N \xi_i \approx 0$ (uncorrelated centered ξ's). For tractability, we consider a simplified load-balancing loss

$$\mathcal{L}_{\text{bal}}(P) := \left\| \frac{1}{N}\sum_{i=1}^N \text{softmax}(Px_i) - \frac{1}{E}\mathbf{1} \right\|^2.$$

The following lemma shows that, in the small-logit regime, minimizing load balancing forces the shared direction μ to become routing-insignificant.

Lemma 2. *Assume that* $\max_{1\le i\le N} \|P(\mu+\xi_i)\| \le \varepsilon$ *for some* $\varepsilon > 0$*, and that* $\frac{1}{N}\sum_{i=1}^N \xi_i \approx 0$*. Then*

$$\frac{1}{N}\sum_{i=1}^N \text{softmax}(Px_i) \approx \frac{1}{E}\mathbf{1} + \frac{1}{E}\Pi_E P\mu + O(\varepsilon^2),$$

and therefore

$$\mathcal{L}_{\text{bal}}(P) \approx \frac{1}{E^2}\|\Pi_E P\mu\|^2 + O(\varepsilon^3).$$

In particular, any near-minimizer of $\mathcal{L}_{\text{bal}}$ *must satisfy*

$$\Pi_E P\mu \approx 0 \qquad \Longleftrightarrow \qquad P\mu \approx \overline{P\mu}\,\mathbf{1},$$

where $\overline{P\mu} := \frac{1}{E}\mathbf{1}^\top P\mu$*.*

Proof. Using the Taylor expansion of softmax at 0, for any $z \in \mathbb{R}^E$ with $\|z\| \le \varepsilon$,

$$\text{softmax}(z) = \frac{1}{E}\mathbf{1} + \frac{1}{E}\Pi_E z + O(\|z\|^2),$$

where the $O(\|z\|^2)$ term is uniform for $\|z\| \le \varepsilon$. Applying this with $z_i = P(\mu+\xi_i)$, we get

$$\text{softmax}(Px_i) = \frac{1}{E}\mathbf{1} + \frac{1}{E}\Pi_E P(\mu+\xi_i) + O(\varepsilon^2).$$

Averaging over i and using $\frac{1}{N}\sum_i \xi_i = 0$,

$$\frac{1}{N}\sum_{i=1}^N \text{softmax}(Px_i) = \frac{1}{E}\mathbf{1} + \frac{1}{E}\Pi_E P\mu + O(\varepsilon^2).$$

Subtracting $\frac{1}{E}\mathbf{1}$ and squaring yields

$$\mathcal{L}_{\text{bal}}(P) = \left\| \frac{1}{E}\Pi_E P\mu + O(\varepsilon^2) \right\|^2 = \frac{1}{E^2}\|\Pi_E P\mu\|^2 + O(\varepsilon^3),$$

since $\|\Pi_E P\mu\| = O(\varepsilon)$ in the small-logit regime. This proves the claim. □

Geometric interpretation. The condition $P\mu = \overline{P\mu}\,\mathbf{1}$ means that the shared direction μ contributes the *same* logit shift to every expert. Hence μ induces no relative preference between experts and is invisible to routing. Equivalently,

$$\Pi_E P\mu = 0,$$

so μ lies in the kernel of the discriminative part of the router $\Pi_E P$. Writing the rows of P as $p_1^\top, \dots, p_E^\top$, this is also equivalent to

$$\langle p_e - p_{e'}, \mu \rangle = 0 \qquad \forall e, e' \in \{1, \dots, E\},$$

showing that all expert-separating directions are orthogonal to μ. Thus the routing decision is invariant along the shared direction.

This gives a simple theoretical explanation for why highly correlated inputs tend to make the router suppress shared directions. In the model $x_i = \mu + \xi_i$, when μ is large, every token carries the same

strong common component. Load balancing discourages such common structure from affecting routing, because it would create the same expert bias across the batch. The optimal response is therefore to make μ *routing-insignificant*, namely $P\mu \in \mathrm{span}(\mathbf{1})$. Since adding a multiple of $\mathbf{1}$ does not change softmax, the router effectively ignores the shared direction and routes using only the residual variations ξ_i. In more geometric terms, balanced routing factors through the quotient that removes the common correlated direction.

This analysis considered a router P trained to achieve load-balancing only. Realistically, the training of P is conducted simultaneously with other weight matrices in pretraining, and the load-balancing loss is an auxiliary loss. Therefore, the conclusions from this analysis should only be partially observed in practice. Specifically, instead of the router P making high correlation directions fully insignificant, it should be expected that P *downscales* the magnitude of those directions instead.

Lemma 3 (Sigmoid-Normalize Router)**.** *Let the router probabilities be given by the sigmoid-then-normalize map*

$$\sigma\text{-norm}(z)_e \;=\; \frac{\sigma(z_e)}{\sum_{e'=1}^{E}\sigma(z_{e'})}, \qquad \sigma(t)=\frac{1}{1+e^{-t}}.$$

Assume that $\max_{1\le i\le N}\|P(\mu+\xi_i)\|\le\varepsilon$ *for some* $\varepsilon>0$*, and that* $\frac{1}{N}\sum_{i=1}^{N}\xi_i\approx 0$*. Then*

$$\frac{1}{N}\sum_{i=1}^{N}\sigma\text{-norm}(Px_i) \;\approx\; \frac{1}{E}\mathbf{1}+\frac{1}{2E}\,\Pi_E P\mu+O(\varepsilon^2),$$

and therefore

$$\mathcal{L}_{\mathrm{bal}}(P) \;\approx\; \frac{1}{4E^2}\left\|\Pi_E P\mu\right\|^2+O(\varepsilon^3).$$

In particular, any near-minimizer of $\mathcal{L}_{\mathrm{bal}}$ *must satisfy*

$$\Pi_E P\mu\approx 0 \quad\Longleftrightarrow\quad P\mu\approx\overline{P\mu}\,\mathbf{1}, \qquad \overline{P\mu}:=\tfrac{1}{E}\mathbf{1}^\top P\mu.$$

Proof. **Step 1 (Taylor expansion of σ at 0).** For a scalar t with $|t|\le\varepsilon$, antisymmetry of $\sigma(t)-\frac{1}{2}$ about the origin kills the quadratic term:

$$\sigma(t)=\frac{1}{2}+\frac{1}{4}t-\frac{1}{48}t^3+O(t^5)=\frac{1}{2}+\frac{1}{4}t+O(\varepsilon^2).$$

Step 2 (numerator and denominator). Writing $z=P(\mu+\xi_i)\in\mathbb{R}^E$ with $\|z\|\le\varepsilon$, componentwise

$$\sigma(z_e)=\frac{1}{2}+\frac{1}{4}z_e+O(\varepsilon^2),$$

and summing over experts,

$$S(z):=\sum_{e=1}^{E}\sigma(z_e)=\frac{E}{2}+\frac{1}{4}\mathbf{1}^\top z+O(\varepsilon^2).$$

Step 3 (expand the ratio). Using $\dfrac{1}{S(z)}=\dfrac{2}{E}\Big(1-\dfrac{1}{2E}\mathbf{1}^\top z\Big)+O(\varepsilon^2)$,

$$\begin{aligned}\sigma\text{-norm}(z)_e &= \sigma(z_e)\,\frac{1}{S(z)}\\ &= \Big(\frac{1}{2}+\frac{1}{4}z_e\Big)\,\frac{2}{E}\Big(1-\frac{1}{2E}\mathbf{1}^\top z\Big)+O(\varepsilon^2)\\ &= \frac{1}{E}+\frac{1}{2E}\Big(z_e-\frac{1}{E}\mathbf{1}^\top z\Big)+O(\varepsilon^2).\end{aligned}$$

Recognizing the projector $\Pi_E=I_E-\frac{1}{E}\mathbf{1}\mathbf{1}^\top$, this is, in vector form,

$$\boxed{\sigma\text{-norm}(z)=\frac{1}{E}\mathbf{1}+\frac{1}{2E}\,\Pi_E z+O(\|z\|^2).}$$

(Compare the softmax expansion $\frac{1}{E}\mathbf{1} + \frac{1}{E}\Pi_E z + O(\|z\|^2)$: the sigmoid-normalize router has the same structure but *half* the linear sensitivity, reflecting the flatter slope of σ at the origin.)

Step 4 (substitute and average). With $z_i = Px_i = P(\mu + \xi_i)$,

$$\sigma\text{-norm}(Px_i) = \frac{1}{E}\mathbf{1} + \frac{1}{2E}\,\Pi_E P(\mu + \xi_i) + O(\varepsilon^2).$$

Averaging over i and using $\frac{1}{N}\sum_i \xi_i \approx 0$,

$$\frac{1}{N}\sum_{i=1}^{N} \sigma\text{-norm}(Px_i) = \frac{1}{E}\mathbf{1} + \frac{1}{2E}\,\Pi_E P\mu + O(\varepsilon^2).$$

Step 5 (load-balancing loss). Subtracting $\frac{1}{E}\mathbf{1}$ and squaring,

$$\mathcal{L}_{\text{bal}}(P) = \Big\| \frac{1}{2E}\,\Pi_E P\mu + O(\varepsilon^2) \Big\|^2 = \frac{1}{4E^2}\,\big\|\Pi_E P\mu\big\|^2 + O(\varepsilon^3),$$

since $\|\Pi_E P\mu\| = O(\varepsilon)$ in the small-logit regime. This proves the claim. □

C.2 Empirical verification of Lemma 2

We pre-trained a 500M, 64-expert (8 activated), MoE for 2000 iterations, considered different batch size (2^{15} to 2^{22} tokens), and fixed load balancing loss coefficient as 1.0. We then study the per-direction data router alignment against data's singular value spectrum (Eq. (9)), similar to Fig. 20.

The results are shown in Fig. 12, where we confirm that auxiliary loss, combined with small batch size, push the router to ignore the data top direction, a behavior that solvable by increasing the batch size.

D Depth scaling and rank collapse

Tracking the residual update token-wise,

$$\boldsymbol{h}_i^{\ell+1} \;=\; \boldsymbol{h}_i^{\ell} + \alpha\,\boldsymbol{\Delta}_i^{\ell}, \tag{24}$$

where i indexes tokens and $\boldsymbol{\Delta}_i^{\ell}$ is updates from the attention and MoE block output at layer ℓ. The cross-token inner product evolves as

$$\langle \boldsymbol{h}_i^{\ell+1}, \boldsymbol{h}_j^{\ell+1}\rangle \;=\; \langle \boldsymbol{h}_i^{\ell}, \boldsymbol{h}_j^{\ell}\rangle + \alpha\big[\langle \boldsymbol{h}_i^{\ell}, \boldsymbol{\Delta}_j^{\ell}\rangle + \langle \boldsymbol{\Delta}_i^{\ell}, \boldsymbol{h}_j^{\ell}\rangle\big] + \alpha^2\langle \boldsymbol{\Delta}_i^{\ell}, \boldsymbol{\Delta}_j^{\ell}\rangle. \tag{25}$$

At *random initialization*, $\boldsymbol{\Delta}_j^{\ell}$ is a function of the layer-ℓ weight matrices with zero-mean and independent of $\boldsymbol{h}_i^{\ell}$ (whose randomness comes from earlier layers' weights). Hence $\mathbb{E}\left[\langle \boldsymbol{h}_i^{\ell}, \boldsymbol{\Delta}_j^{\ell}\rangle\right] = 0$, and the cross terms in (25) vanish in expectation. Taking expectation,

$$\mathbb{E}\left[\langle \boldsymbol{h}_i^{\ell+1}, \boldsymbol{h}_j^{\ell+1}\rangle\right] \;=\; \mathbb{E}\left[\langle \boldsymbol{h}_i^{\ell}, \boldsymbol{h}_j^{\ell}\rangle\right] + \alpha^2\,\mathbb{E}\left[\langle \boldsymbol{\Delta}_i^{\ell}, \boldsymbol{\Delta}_j^{\ell}\rangle\right], \tag{26}$$

$$\mathbb{E}\left[\big\|\boldsymbol{h}_i^{\ell+1}\big\|^2\right] \;=\; \mathbb{E}\left[\big\|\boldsymbol{h}_i^{\ell}\big\|^2\right] + \alpha^2\,\mathbb{E}\left[\big\|\boldsymbol{\Delta}_i^{\ell}\big\|^2\right], \tag{27}$$

where off-diagonal recursion accumulates contributions purely at second order in α. For $i \neq j$, $\mathbb{E}\left[\langle \boldsymbol{\Delta}_i^{\ell}, \boldsymbol{\Delta}_j^{\ell}\rangle\right]$ need not to be zero. There are two structural sources of cross-token correlation in the block output:

- *Attention value sharing.* Self-attention writes $\boldsymbol{\Delta}_i^{\ell}$ as a weighted sum over the same set of value vectors $\{\boldsymbol{V}\boldsymbol{h}_k^{\ell}\}_k$ as $\boldsymbol{\Delta}_j^{\ell}$. Both updates are convex combinations of a common pool, so $\langle \boldsymbol{\Delta}_i^{\ell}, \boldsymbol{\Delta}_j^{\ell}\rangle$ inherits a contribution from every value vector both tokens attend to.
- *Shared projection weights.* If $\boldsymbol{\Delta}_i = \boldsymbol{W}\phi(\boldsymbol{h}_i)$ and $\boldsymbol{\Delta}_j = \boldsymbol{W}\phi(\boldsymbol{h}_j)$ with $\boldsymbol{W}$ shared across tokens (true for the value, output, and MLP projections), then

$$\mathop{\mathbb{E}}_{\boldsymbol{W}}\left[\langle \boldsymbol{\Delta}_i, \boldsymbol{\Delta}_j\rangle\right] \;=\; \phi(\boldsymbol{h}_i)^{\top}\,\mathbb{E}\left[\boldsymbol{W}^{\top}\boldsymbol{W}\right]\phi(\boldsymbol{h}_j),$$

  which is non-zero whenever $\phi(\boldsymbol{h}_i)$ and $\phi(\boldsymbol{h}_j)$ have non-trivial inner product.

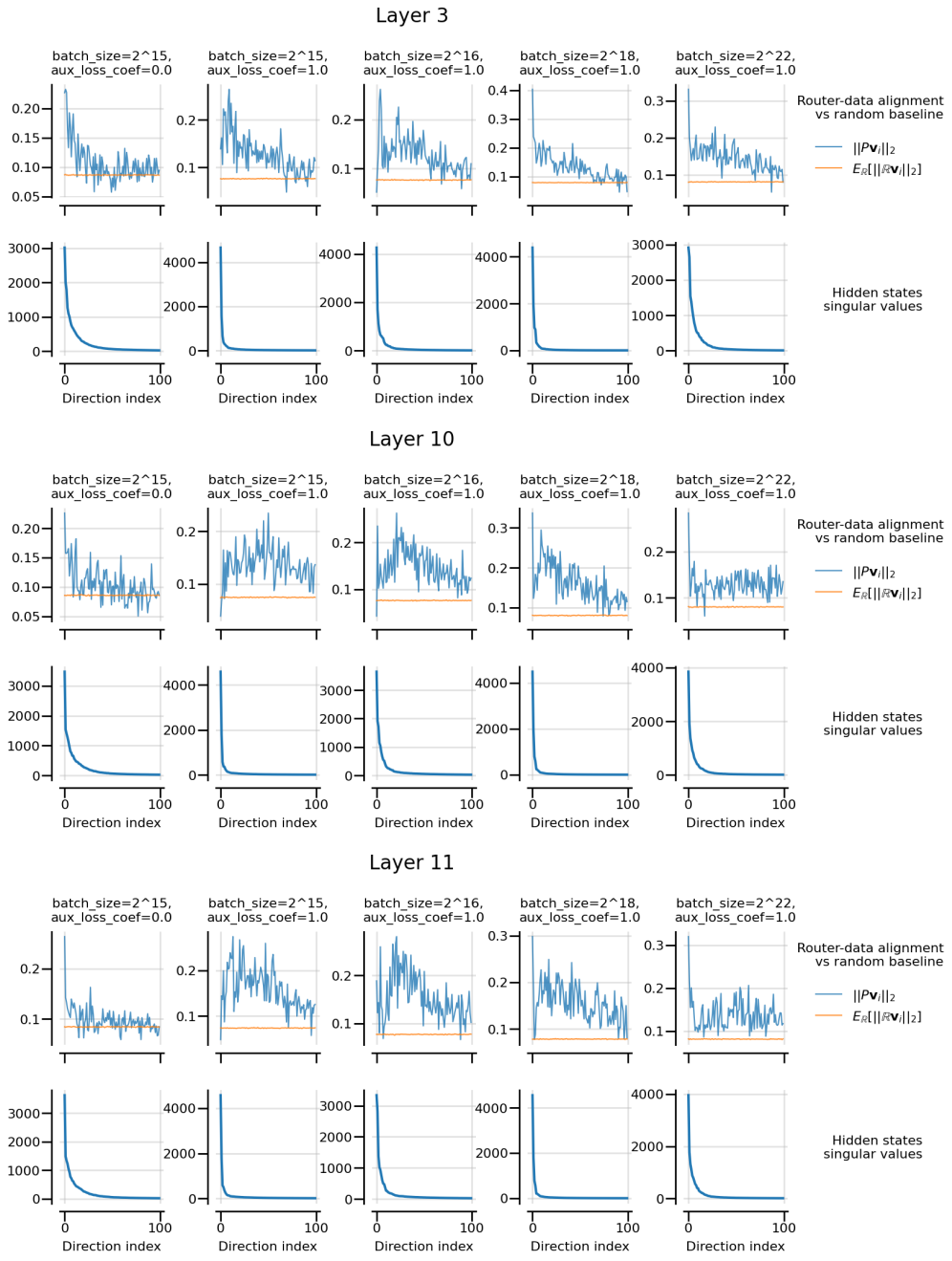


Figure 12: Empirical verification of router suppression data top directions under load balancing loss (Lemma 2). **Left-most column:** No load balancing loss, router aligns with data's top directions. **Second column to the right-most column:** Batch size from small to large, at small batch size, the alignment shows a reverse U shape, where the router is least aligned with the top singular direction, i.e. the shared common direction. As the batch size increases (e.g. 2^{22}), the router starts to, again, align with the data's leading directions.

Solving the recursion. By symmetry over tokens, write $D^\ell := \mathbb{E}\left[\left\|\boldsymbol{h}_i^\ell\right\|^2\right]$ and $C^\ell := \mathbb{E}\left[\langle \boldsymbol{h}_i^\ell, \boldsymbol{h}_j^\ell \rangle\right]$ for any $i \neq j$. The cross-token cosine similarity at depth L is

$$\rho^L := \frac{\mathbb{E}\left[\langle \boldsymbol{h}_i^L, \boldsymbol{h}_j^L \rangle\right]}{\mathbb{E}\left[\left\|\boldsymbol{h}_i^L\right\|^2\right]} = \frac{C^L}{D^L} \tag{28}$$

Define the per-layer *correlation injection rate* as $\kappa^\ell := \frac{\mathbb{E}\left[\langle \mathbf{\Delta}_i^\ell, \mathbf{\Delta}_j^\ell \rangle\right]}{\mathbb{E}\left[\left\|\mathbf{\Delta}_i^\ell\right\|^2\right]}, \kappa^\ell \in [0,1]$, which captures how much of the block's update lies along directions shared across tokens. Under the structural sources above, κ^ℓ is bounded away from zero at random initialization. Note that κ^ℓ is not constant with depth; it has a self-amplification effect, which can be modeled as

$$\kappa^\ell \;=\; \kappa_0 + \beta\, \rho^\ell, \tag{29}$$

where $\kappa_0 \geq 0$ is the *exogenous* correlation injection (independent of the current ρ^ℓ, controlled by attention pattern entropy and FFN nonlinearity bias) and $\beta \geq 0$ is the *self-amplification* coefficient.

Assume for simplicity that $\mathbb{E}\left[\left\|\mathbf{\Delta}_i^\ell\right\|^2\right] = \sigma^2$ (true under pre-LN) Then (26) and (27) solve to

$$C^{\ell+1} \;=\; C^\ell + \alpha^2(\kappa_0 + \beta\rho^\ell)\,\sigma^2, \tag{30}$$

$$D^{\ell+1} \;=\; D^\ell + \alpha^2\sigma^2. \tag{31}$$

Dividing (30) by $D^{\ell+1}$ and using $C^\ell = \rho^\ell D^\ell = \rho^\ell(D^{\ell+1} - \alpha^2\sigma^2)$,

$$\rho^{\ell+1} \;=\; \frac{C^\ell + \alpha^2(\kappa_0 + \beta\rho^\ell)\sigma^2}{D^\ell + \alpha^2\sigma^2} \;=\; \rho^\ell + \frac{\alpha^2\sigma^2}{D^{\ell+1}}\big[\kappa_0 + (\beta-1)\rho^\ell\big].$$

For large enough ℓ that $D^\ell \approx \ell\alpha^2\sigma^2$ (Eq. (31) starting from $D^0 = \sigma^2 = d$), this is well-approximated by the continuous-depth ODE

$$\frac{d\rho}{d\ell} \;=\; \frac{\alpha^2}{1+\ell\alpha^2}\Big[\kappa_0 - (1-\beta)\rho\Big], \tag{32}$$

whose solution with $\rho^0 = 0$ is

$$\boxed{\rho^\ell \;=\; \frac{\kappa_0}{1-\beta}\left[1 - (1+\ell\alpha^2)^{-(1-\beta)}\right]} \qquad (\beta \neq 1). \tag{33}$$

Vanilla pre-norm ($\alpha = 1$). The factor $1 + \ell\alpha^2 = \ell + 1$ grows linearly. In the subcritical regime ($\beta < 1$), $\rho^L \to \kappa_0/(1-\beta)$ as $L \to \infty$; in the critical regime (($\beta = 1$)) it grows as $\log L$; in the supercritical regime ($\beta > 1$) it saturates at $\rho = 1$ within $O(1)$ layers. The $\beta > 1$ behavior matches the doubly-exponential rank collapse documented empirically in attention-only stacks (**?**).

Depth-scaled pre-norm ($\alpha^2 = 1/L$). The factor $1 + \ell\alpha^2$ ranges from 1 to 2 across the network. Substituting into (33), the cosine at the top layer is

$$\rho^L \;=\; \frac{\kappa_0}{1-\beta}\big[1 - 2^{-(1-\beta)}\big], \tag{34}$$

independent of L, in all three regimes including supercritical. Depth scaling does not just slow the supercritical blowup—it caps it at a constant determined only by κ_0 and β.

E EXAMPLE MODEL OUTPUTS

E.1 GPT-OSS-20B

```
Model input:
Summarize the following paragraph: [history of wimbledon]

Model output:
<|channel|>analysis<|message|>We need to summarize the text. Very long
history.
```

```
Model input:
Think step by step, answer the question: [HLE question]

Model output:
<|channel|>analysis<|message|>We need to interpret the problem:
```

E.2 DUPLICATION

Under duplications, we considered an instruction tuned model, while we did not see outputs of different inputs collapse to the same output (despite expert usage similarity increases), we did observe the output to lose randomness under random decoding.

```
Duplication x 1:
Query 1 output:

If you're a coder: What language are you coding in, and what does your
current stack look like?

Some key shifts:
- More attention on internal tooling (infrastructure, DevOps, internal
APIs)
- Higher value on high-quality, maintainable code
- Smaller, faster iterations with
******************************
Query 2 output:
```
#@title Sampling
def p_sample_loop_score(model, shape, clip_denoised=True, denoised_fn=
None, transform=None):
    '''
    A generic function to follow the denoising refinemnet API.
    '''
    assert transform is not None
    b =

Duplication x 4:
Query 1 output:

OpenAI and xAI are finally deprioritizing side quests and focusing on
coding, while Anthropic has been focused on coding from the start.

We're likely heading toward 3-4 top-tier coding models (including Gemini)
, it's not a winner-take-all market. Each model will excel
****
Query 2 output:

1. The Forward Process: The SDEThe forward diffusion process $\{ \mathbf{
x}_t \}_{t=0}^T$ transforms a data sample $\mathbf{x}_0 \sim p_{data}(\
mathbf{x})$ into Gaussian noise $\mathbf{x}_T \sim \mathcal{N}(\mathbf
{0},

```
```

```
Duplication x 16:
Query 1 output:

OpenAI and xAI are finally deprioritizing side quests and focusing on
coding, while Anthropic has been focused on coding from the start.

We're likely heading toward 3-4 top-tier coding models (including Gemini)
, it's not a winner-take-all market. Each model will excel
******************************
Query 2 output:

1. The Forward Process: The SDEThe forward diffusion process $\{ \mathbf{
x}_t \}_{t=0}^T$ transforms a data sample $\mathbf{x}_0 \sim p_{data}(\
mathbf{x})$ into Gaussian noise $\mathbf{x}_T \sim \mathcal{N}(\mathbf
{0},
------------------------------------------------------------
Duplication x 32:
Query 1 output:

OpenAI and xAI are finally deprioritizing side quests and focusing on
coding, while Anthropic has been focused on coding from the start.

We're likely heading toward 3-4 top-tier coding models (including Gemini)
, it's not a winner-take-all market. Each model will excel
******************************
Query 2 output:

1. The Forward Process: The SDEThe forward diffusion process $\{ \mathbf{
x}_t \}_{t=0}^T$ transforms a data sample $\mathbf{x}_0 \sim p_{data}(\
mathbf{x})$ into Gaussian noise $\mathbf{x}_T \sim \mathcal{N}(\mathbf
{0},
------------------------------------------------------------
```

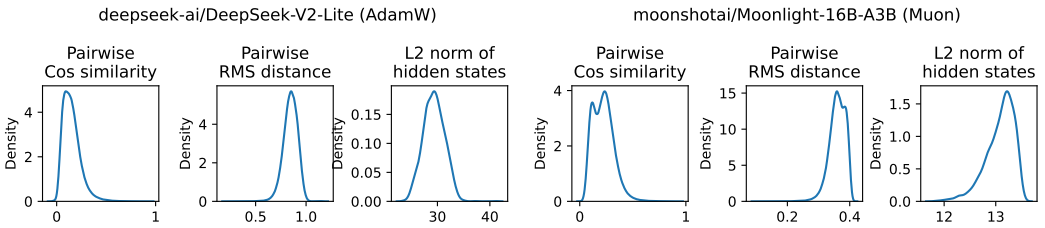


Figure 13: **Same model architecture, different optimizers induce different feature space properties.** At the same layer, given the same sequence, both models show a similar distribution of pairwise cosine similarity (over all tokens). However, embeddings in the model trained with AdamW show, on average, a higher RMS distance, indicating that different tokens exhibit high variation in norm (25 - 35). In contrast, Muon trained model shows more concentrated distribution of feature norm (12 - 14) and smaller pairwise RMS distances.

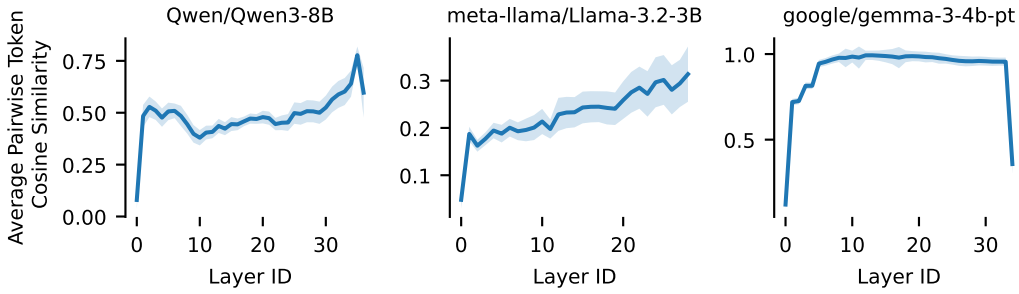


Figure 14: On dense model, except for Gemma3, which uses sandwich norm, token similarities also increase with depth.

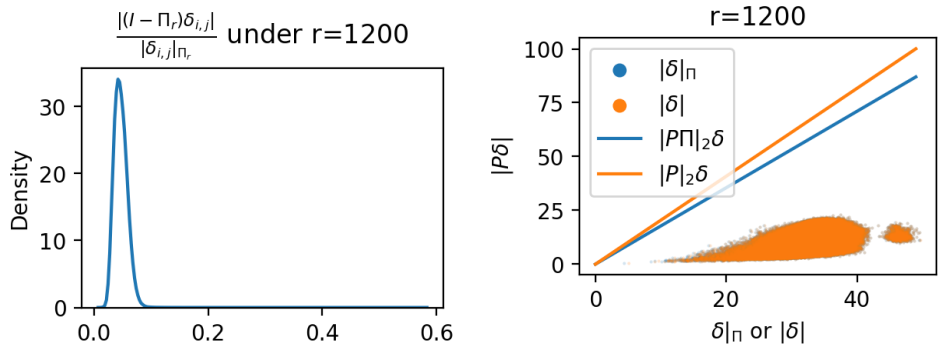
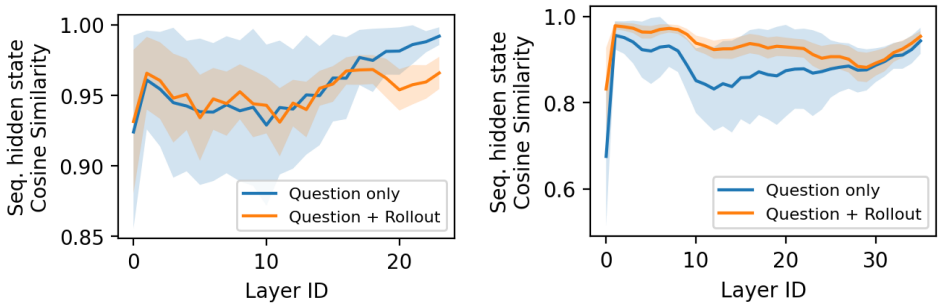
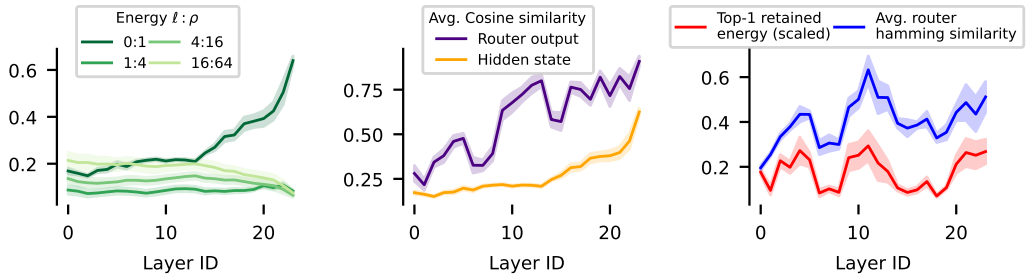


Figure 15: On Deepseek v2 lite, empirical comparison of the naive bound $\|P\|_2\|\delta_{i,j}\|$ versus the tighter projection-based bound $\|P\Pi_r\|_2\|\delta_{i,j}\|_{\Pi_r}$ from Proposition 1, where $\delta_{i,j} = \boldsymbol{h}_i - \boldsymbol{h}_j$. **Left:** the residual ratio $|(I - \Pi_r)\delta_{i,j}| \,/\, |\delta_{i,j}|_{\Pi_r}$ is sharply concentrated near zero under $r = 1200$, confirming that the residual term $\|P(I - \Pi_r)\delta_{i,j}\|_2$ is negligible and the bound in Eq. (4) is approximately tight with respect to the SVD truncation. **Right:** the actual router logit differences $|P\delta|$ (orange dots) fall far below both upper bounds. While the projection-based bound $\|P\Pi_r\|_2\,|\delta|_{\Pi_r}$ (blue line) is theoretically tighter than the naive bound $\|P\|_2\,|\delta|$ (orange line), P is already well-aligned with the principal subspace of the data, the gap between the two bounds is less significant in pre-trained models.

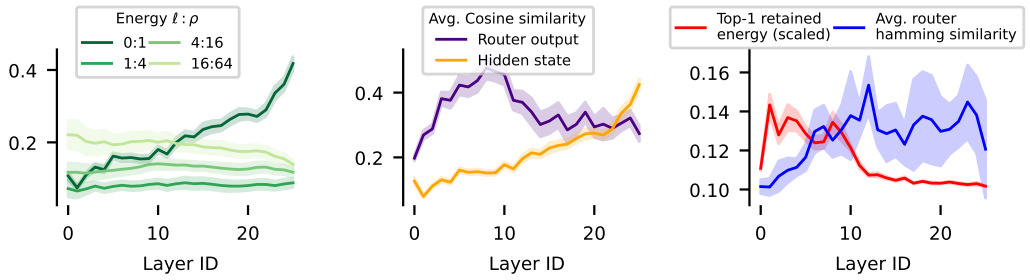


(a) gpt-oss-20b (b) Qwen3-4B-Thinking-2507

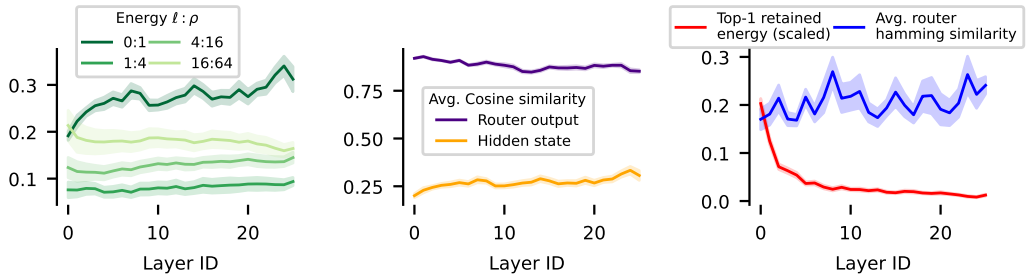
Figure 16: Hidden states similarity (measured by Eq. (19)) v.s. depth for the prompt / question part v.s. question + rollout on two different reasoning models tested on 10 questions from HLE. **Left** On gpt-oss, the similarity of the rollout part falls below the question-only line and does not show collapsing, aligning with the observations in Fig. 10. **Right** On a dense reasoning model from different lab, while we did see an increment of hidden state similarity across depth, we did not observe a clear separation in deeper layers.



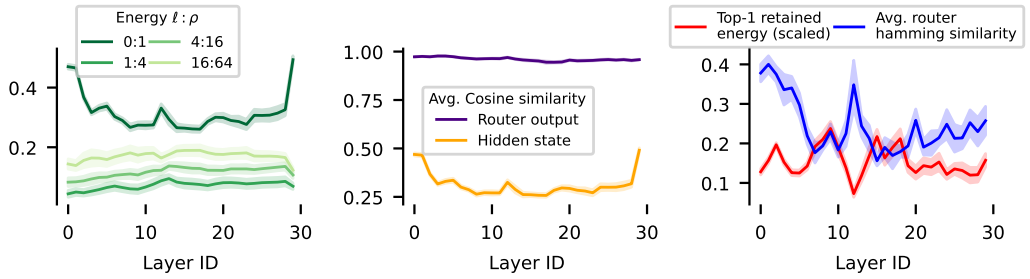
(a) GPT-OSS-20B (post-trained): **AdamW + explicit auxiliary loss**



(b) DeepSeek-V2-Lite (base): **AdamW + explicit auxiliary loss**



(c) Moonlight 16B (structure identical to DeepSeek-V2-Lite): **Muon + auxiliary-loss-free**



(d) Trinity-Mini-Base: **Muon + auxiliary-loss-free**

Figure 17: Similar to Fig. 2 in the main text, but for more models

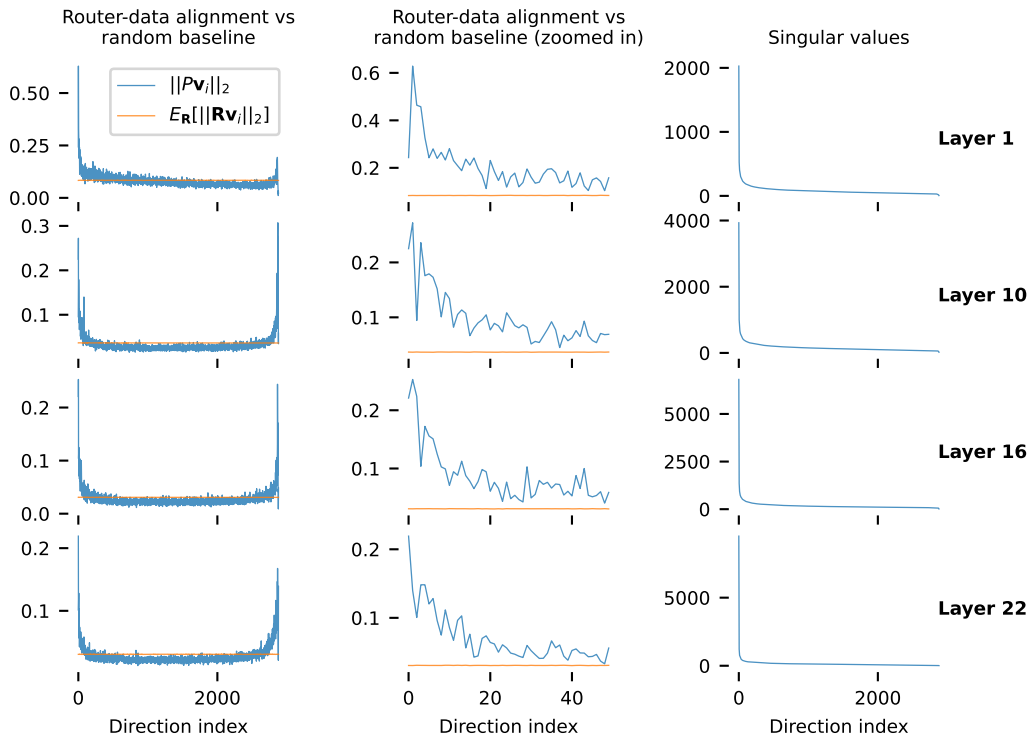


Figure 18: gpt-oss-20b (w/ auxiliary loss)

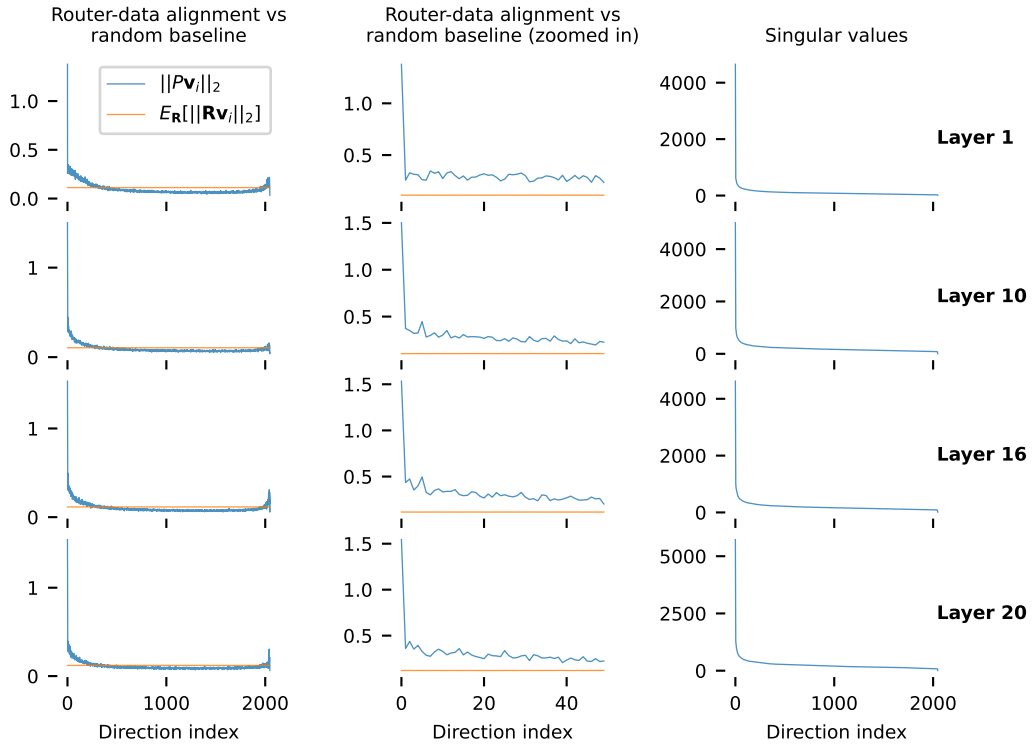


Figure 19: Trinity Mini Base (Muon, no auxiliary loss)

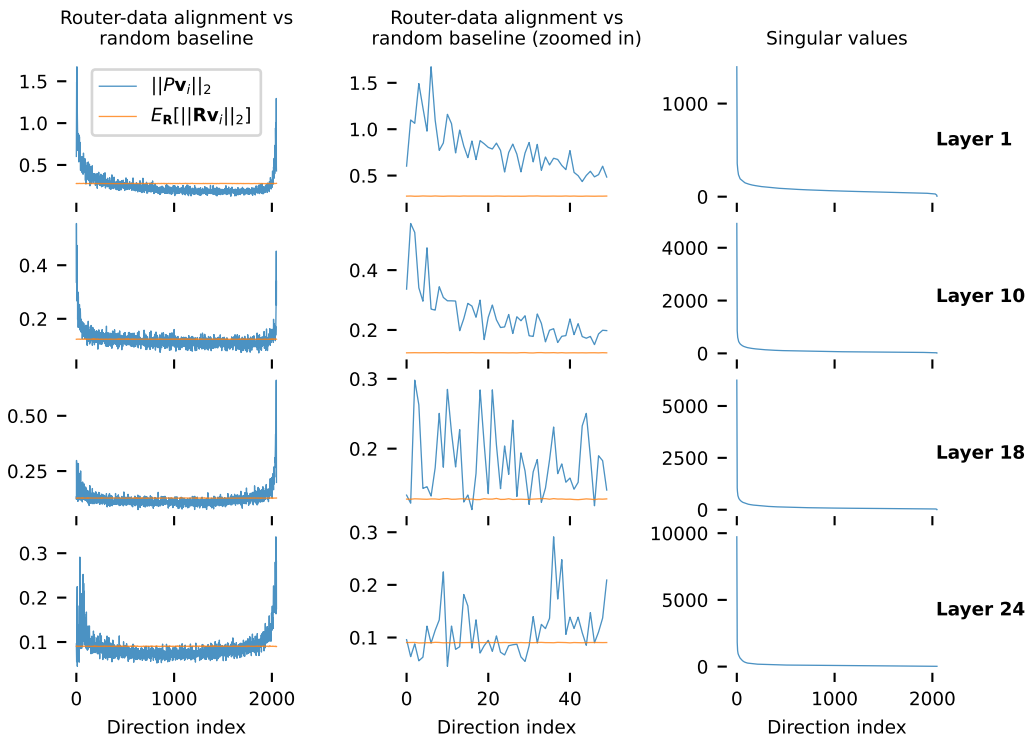


Figure 20: DeepSeek V2 Lite (AdamW, w/ auxiliary loss)

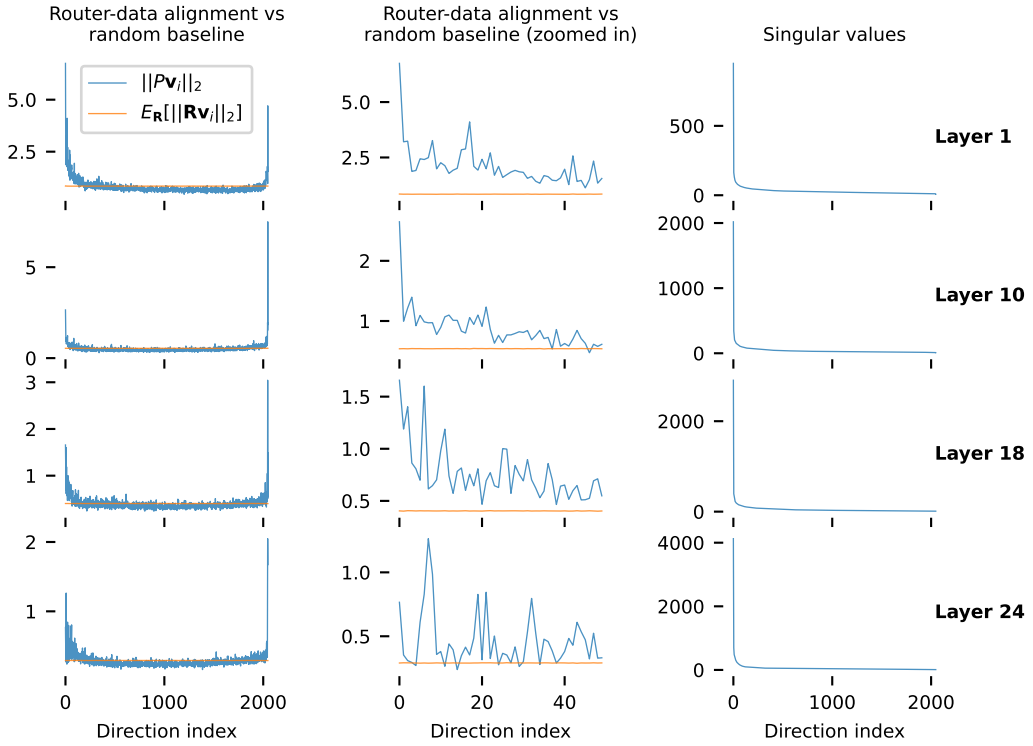


Figure 21: Moonlight 16B A3B, same architecture as DeepSeek V2 Lite, trianed with Muon and no auxiliary loss

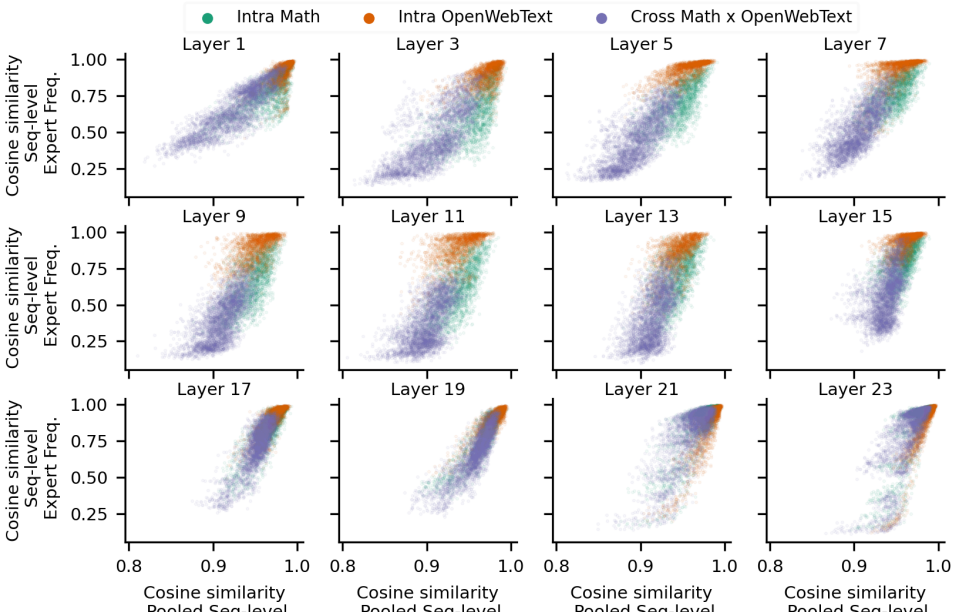


Figure 22: gpt-oss-20b

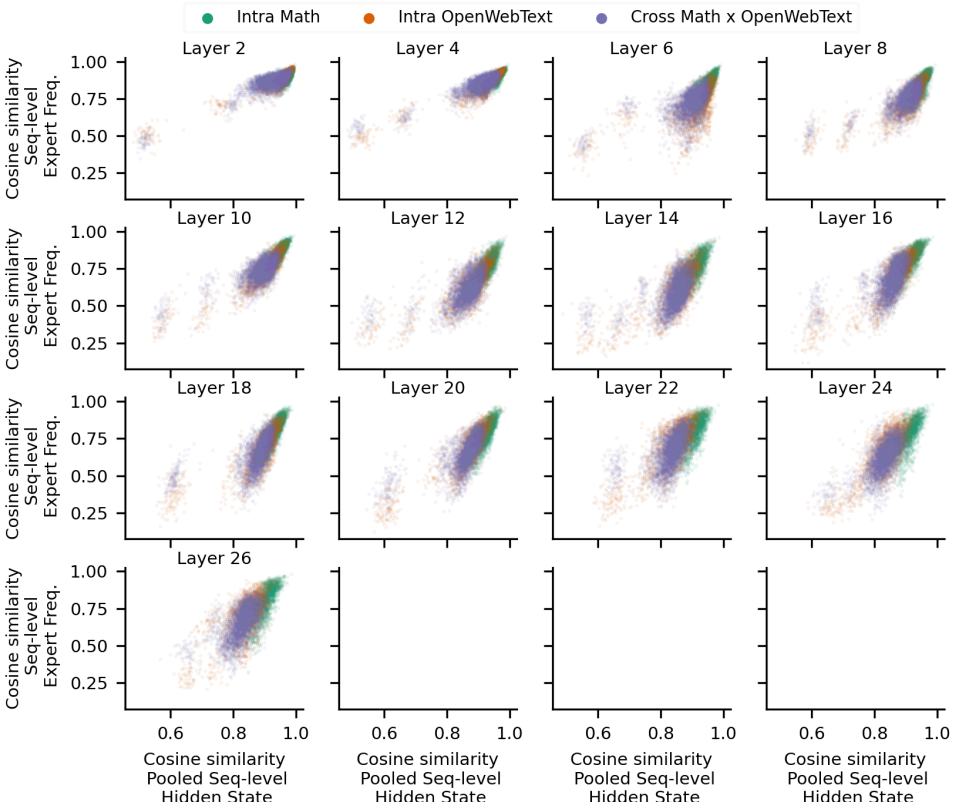


Figure 23: Ernie 4.5 21B-A3B-Base

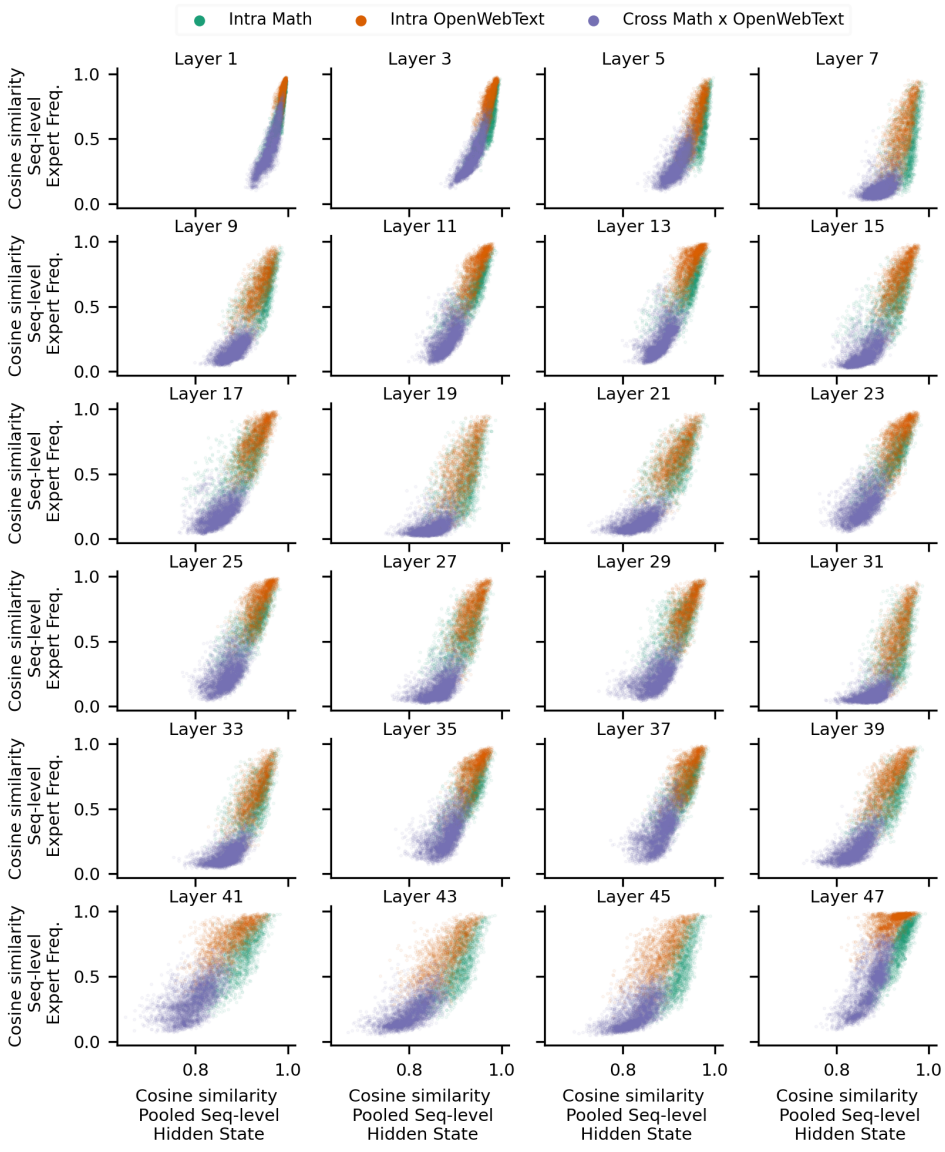


Figure 24: Qwen3-30B-A3B

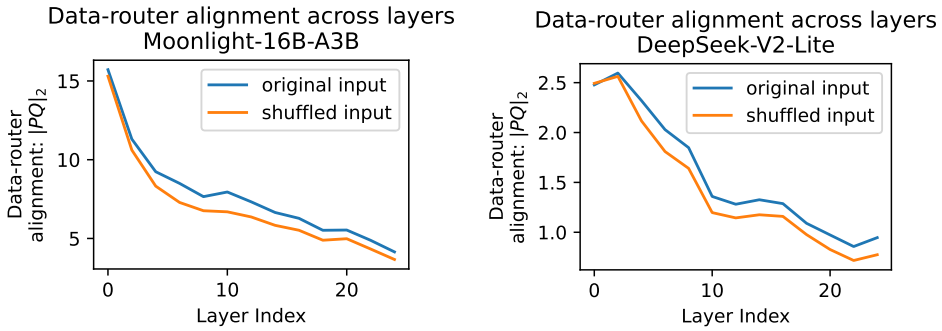


Figure 25: OOD inputs reduce router data alignment ($\|PQ\|_2$), where $Q = V_r V_r^\top$ denotes the orthogonal projector onto the subspace spanned by the data's top r right singular vectors.

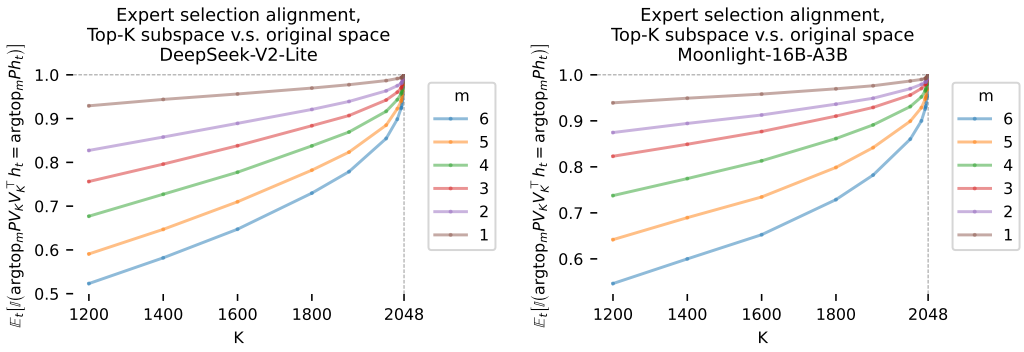


Figure 26: Keeping only top-K directions (V_K) of the data (i.e. the post attention hidden states, denoted by h_t), we study the ratio of expert usage agreement compared with the original inputs, averaged over 50,000 tokens from `nvidia/Nemotron-CC-Math-v1` at the 10th layer of each model. The expert receiving the largest logits (brown line, $m = 1$) shows high agreement over all tokens. Lower rank experts (e.g. the 6th expert, $m = 6$) demonstrate a higher degree of perturbations when the tail directions are removed; more than 40% of the tokens end up using a different expert compared with the original inputs at $K = 1200$ (out of 2048 directions in total).

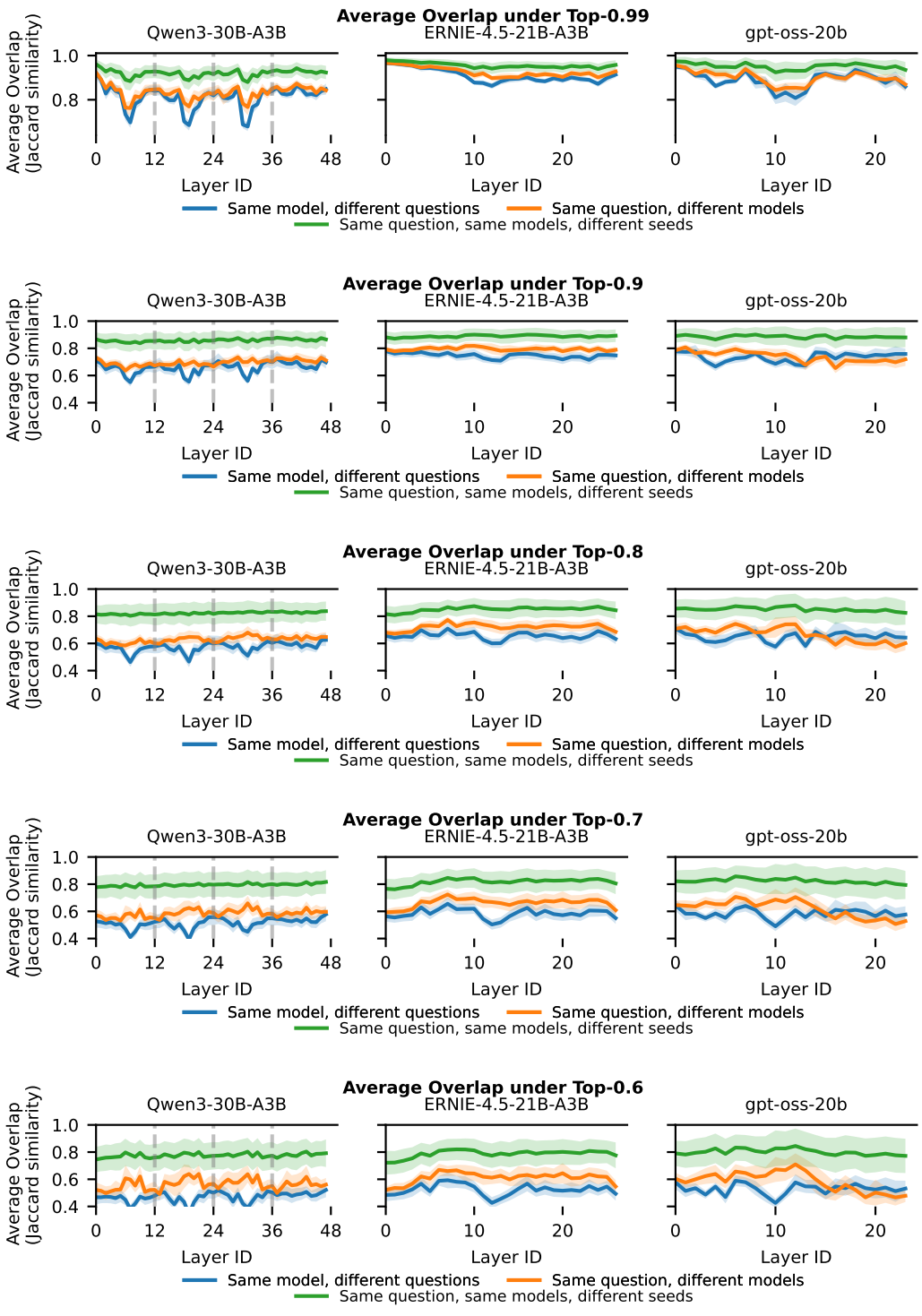


Figure 27: Similiar to Fig. 6 in the main text, but for more values of p.

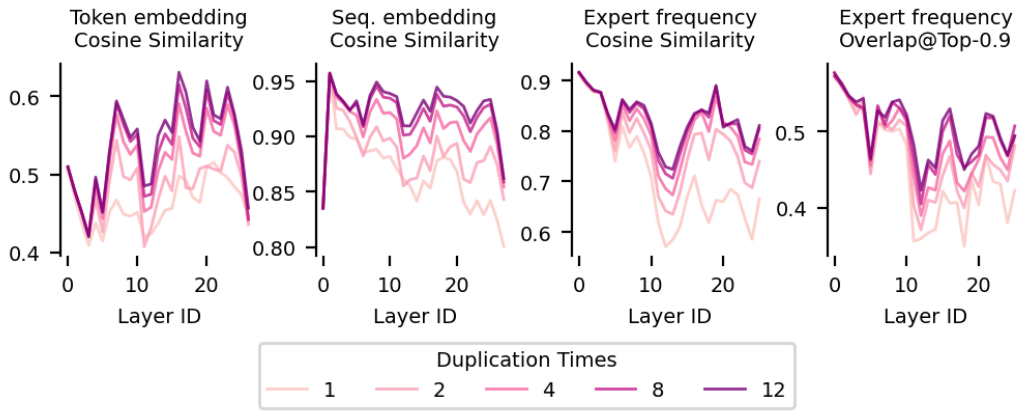


Figure 28: On Ernie-4.5, duplicating sequences increases token hidden states similarity (first column), i.e. stronger shared direction, resulting in higher sequence hidden state similarity (second column). The increment in hidden state space results in increased expert usage similarity (last two columns).

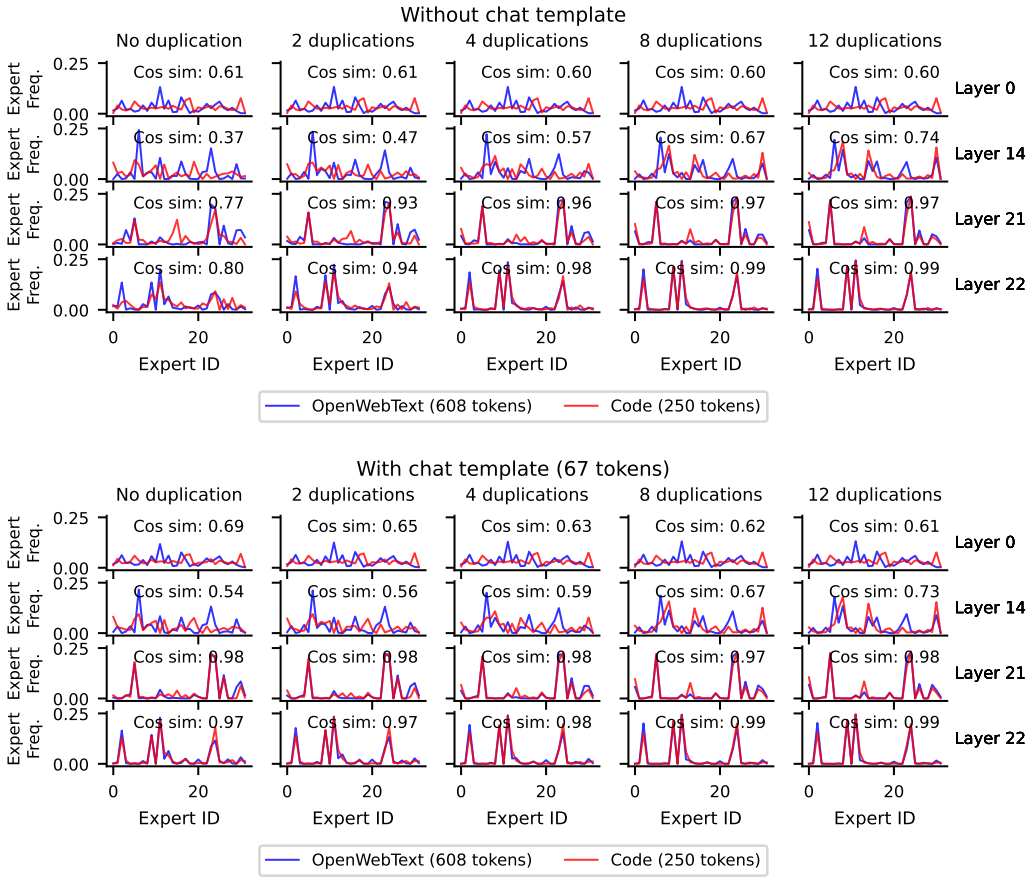


Figure 29: Comparison of expert usage under duplication for gpt-oss-20b with (bottom) and without (top) chat template on two pieces of input.

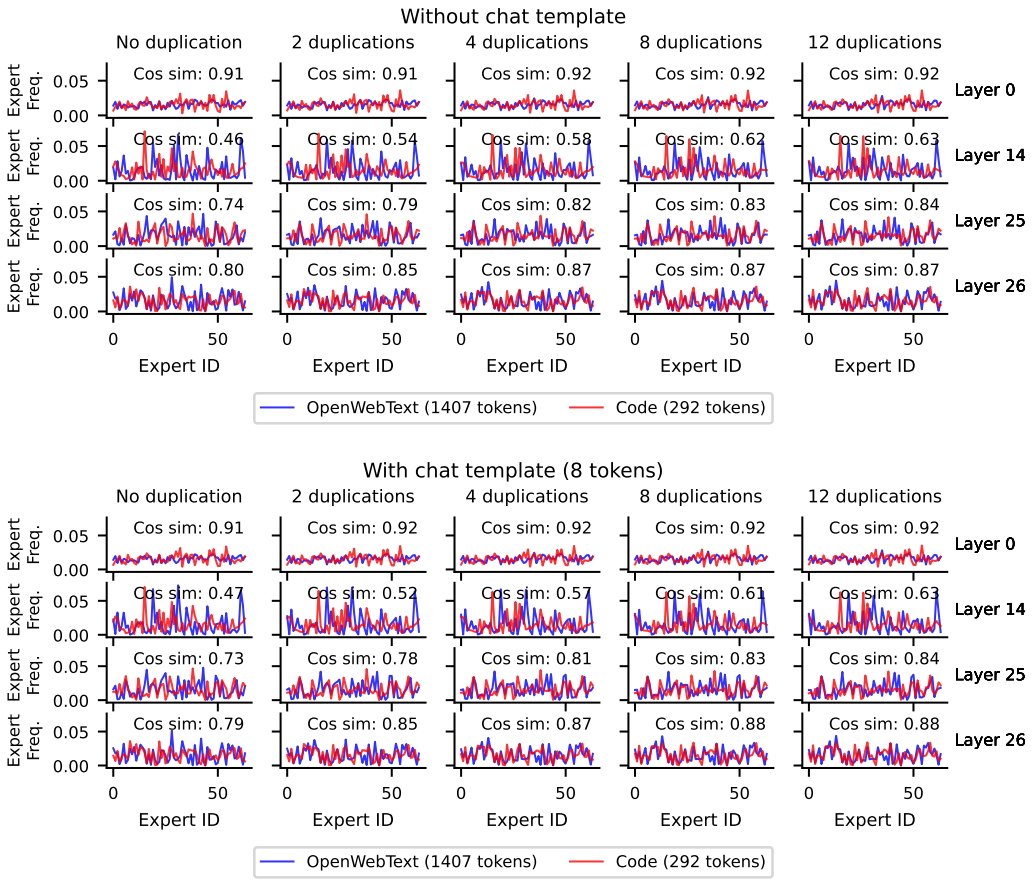


Figure 30: Comparison of expert usage under duplication for ERNIE-4.5-21B-A3B with (bottom) and without (top) chat template on two pieces of input.

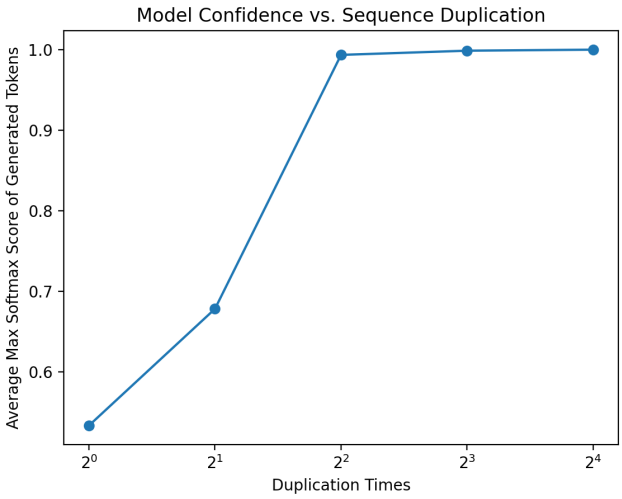


Figure 31: Duplication increases logit magnitude, increasing model confidence and causes random decoding to collapse.