Co-Adaptation Under Reinforcement Learning: Measuring How Scaled RL Reshapes Expert Routing in Production Mixture-of-Experts Models

Forerunner.ai

Abstract

Reinforcement learning (RL) post-training updates *all* parameters of a Mixture-of-Experts (MoE) model jointly—expert weights, gate weights, and routing biases co-adapt simultaneously—yet its effect on expert routing has never been systematically measured, let alone causally tested. We present the first layer-by-layer, domain-stratified analysis of this co-adaptation in production-scale MoE models, together with a *causal intervention* that disentangles routing from expert weight changes. MiniMax-M2.1 and M2.5 share identical architecture—229B parameters, 62 transformer layers, 256 experts, top-8 sigmoid gating—and both undergo SFT and RL via CISPO, but M2.5 extends training with $\sim 2\times$ more environments and process reward signals, isolating the *marginal* effect of scaled RL. By recording per-token routing decisions across all 62 layers over 847,159 tokens spanning six domains, we find that scaled RL produces *domain-selective* routing changes. Code routing concentrates onto fewer experts ($\Delta H = -0.23$ bits, $p < 0.01$; robust under subsampling and sample-level bootstrap), while reasoning ($+0.11$ bits), general ($+0.17$ bits), and instruction-following ($+0.17$ bits) routing disperses. To test whether these routing changes *cause* capability differences, we perform a gate-swap experiment: transplanting M2.5's gate weights into M2.1's body (and vice versa) across all 62 layers and measuring per-domain cross-entropy loss. The results reveal that routing changes are *structurally real but functionally secondary in terms of next-token prediction*: swapping gates produces $<1\%$ perplexity change on code, while reverting M2.5 to M2.1's routing *improves* general-domain perplexity by 7.8%, indicating that the additional RL-induced routing specialization taxes non-target domains.

To test generality, we replicate the observational analysis on DeepSeek V3 (671B parameters, 256 experts with *grouped* top-8 routing) comparing V3-Base (pre-trained) against V3 (SFT+RL). DeepSeek exhibits substantially higher routing stability (mean Jaccard $= 0.67$ vs. MiniMax's 0.60; mean top-1 agreement $= 0.85$ vs. 0.60) and uniformly small entropy changes ($|\Delta H| < 0.12$ bits across all domains), in stark contrast to MiniMax's asymmetric -0.23 to $+0.17$ bit range. DeepSeek's two-stage grouped routing (top-4 of 8 groups, then top-8 of 128 experts) acts as an architectural stabilizer: group-level Jaccard remains 0.82 even as expert-level agreement declines, with 73.5% of routing divergence occurring *within* preserved groups. These cross-architecture results establish that post-training routing changes are universal but architecture-modulated: grouped routing constrains the geometry of co-adaptation, dampening the domain-selective specialization observed under direct routing.

Preprint. Under review.

1 Introduction

Mixture-of-Experts (MoE) architectures scale language model capacity efficiently by activating only a subset of parameters per token [Shazeer et al., 2017, Fedus et al., 2022]. In modern MoE models, each transformer layer routes every token to k of N experts (typically $k = 8$ of $N = 256$), enabling models with hundreds of billions of total parameters to maintain inference costs comparable to much smaller dense models [Jiang et al., 2024, Dai et al., 2024].

While the routing function—the learned gating network that decides which experts process each token—is central to MoE performance, its behavior under scaled RL remains poorly understood. Reinforcement learning from human feedback (RLHF) and related techniques have become standard for aligning language models [Ouyang et al., 2022, Bai et al., 2022], yet their effect on MoE routing has not been systematically studied. Prior work on MoE interpretability has focused on pre-training dynamics [Zoph et al., 2022] or architectural design [Dai et al., 2024], leaving a gap in understanding how RL reshapes the expert routing landscape. Critically, whether the observed routing changes are *causes* or *consequences* of capability improvement.

MiniMax provides a natural experiment to fill this gap. Their M2.1 and M2.5 models share *identical* architecture (229B parameters, 62 transformer layers, 256 experts per layer, top-8 sigmoid gating with learned correction bias). Both models undergo SFT and RL using the CISPO algorithm [MiniMax, 2025], but M2.5 extends training with approximately twice as many RL environments ($\sim$200K vs. $\sim$100K) and adds process reward signals for improved credit assignment in long agent trajectories. This controlled setup isolates the *marginal* effect of scaled RL on routing decisions, free from confounds introduced by architectural changes.

Contributions. We make five contributions:

1. **Domain-selective routing change.** We show that scaled RL does not uniformly change expert routing. Instead, it *concentrates* routing for code ($\Delta H = -0.23$ bits) while *dispersing* routing for reasoning ($+0.11$), general ($+0.17$), and instruction-following ($+0.17$ bits). This asymmetry is consistent with a reward-selective specialization hypothesis linking reward signal structure to routing geometry. We validate the effect under subsampling (matching the smallest domain at $n = 9{,}273$) and sample-level bootstrap.

2. **Depth-dependent divergence.** Cross-model routing agreement decays monotonically from Jaccard $= 0.68$ at layer 5 to 0.42 at layer 61, establishing that scaled RL primarily modifies deep-layer semantic routing while preserving shallow-layer universal features.

3. **Expert role reorganization.** Scaled RL causes 21.9% of experts to change their primary domain affiliation, demonstrating that additional RL does not merely fine-tune gating thresholds but fundamentally restructures the expert division of labor.

4. **Causal gate-swap intervention.** We perform the first causal decomposition of routing versus weight contributions in a production MoE by transplanting gate weights between models. Swapping gates produces $<$1% code perplexity change but reduces general-domain perplexity by 7.8% when reverting to M2.1's routing, establishing that (a) expert weight changes are the primary capability mechanism, (b) routing changes are a *regulatory tax* that over-specializes at the expense of non-target domains, and (c) routing and weights are tightly co-adapted.

5. **Cross-architecture validation on DeepSeek V3.** We replicate the observational analysis on DeepSeek V3 (671B parameters, grouped top-8/256 sigmoid routing), comparing V3-Base against V3 across the same six domains and 861,531 tokens. DeepSeek's two-stage grouped routing (8 groups $\times$ 32 experts, top-4 groups selected) acts as an architectural stabilizer: routing is substantially more preserved (mean Jaccard 0.67 vs. 0.60), entropy changes are uniformly small ($|\Delta H| < 0.12$ bits for all domains), and 73.5% of routing divergence occurs *within* preserved groups rather than across them. This cross-architecture comparison establishes generality while revealing that routing geometry modulates the co-adaptation pattern.

2 Background

2.1 MiniMax-M2 Architecture

MiniMax-M2 is a 229-billion-parameter MoE language model with 62 transformer layers and approximately 10B activated parameters per token [MiniMax, 2025]. Each layer contains a multi-head attention module (48 query heads, 8 key-value heads, head dimension 128, grouped-query attention with 6 groups) followed by a sparse MoE feed-forward module with 256 experts. Each expert is a gated SwiGLU network [Shazeer, 2020]:

$$\text{Expert}_i(\mathbf{x}) = \left(\text{SiLU}(\mathbf{x}\mathbf{W}_1^{(i)\top}) \odot (\mathbf{x}\mathbf{W}_3^{(i)\top})\right) \mathbf{W}_2^{(i)\top} \tag{1}$$

where $\mathbf{W}_1^{(i)}, \mathbf{W}_3^{(i)} \in \mathbb{R}^{1536\times3072}$ and $\mathbf{W}_2^{(i)} \in \mathbb{R}^{3072\times1536}$. All weights are stored in FP8 (E4M3) format with block-wise scale factors (block size 128).

2.2 Routing Mechanism

Tokens are routed to experts via a sigmoid gating function with a learned correction bias:

$$\mathbf{s} = \sigma(\mathbf{x}\mathbf{W}_g^\top) + \mathbf{b}_{\text{corr}} \tag{2}$$

$$\mathcal{E}(\mathbf{x}) = \text{top-}k(\mathbf{s}, k = 8) \tag{3}$$

$$\mathbf{y} = \sum_{i\in\mathcal{E}(\mathbf{x})} \bar{w}_i \cdot \text{Expert}_i(\mathbf{x}) \tag{4}$$

where $\mathbf{W}_g \in \mathbb{R}^{256\times3072}$ is the gate weight matrix, $\mathbf{b}_{\text{corr}} \in \mathbb{R}^{256}$ is the learned correction bias for load balancing, and $\bar{w}_i$ is the renormalized sigmoid score. Unlike softmax-based gating [Shazeer et al., 2017], sigmoid gating allows multiple experts to independently achieve high activation scores, decoupling expert selection from inter-expert competition.

2.3 M2.1 vs. M2.5: Advanced RL as the Independent Variable

Both M2.1 and M2.5 are post-trained models built on the same M2 base via supervised fine-tuning (SFT) followed by reinforcement learning using CISPO (Clipping Importance Sampling Policy Optimization) [MiniMax, 2025]. M2.1 was trained with $\sim$100,000 RL environments spanning 10+ programming languages, producing strong agent and coding capabilities. M2.5 extends this with $\sim$200,000 environments, adds process reward signals for credit assignment in long agent trajectories, and optimizes for task completion speed. Both share identical architecture, expert count, and weight format. The independent variable is the *scale, sophistication, and reward structure* of RL training, making this comparison a natural experiment that isolates the marginal effect of advanced RL on routing. We note that because M2.1 is itself RL-trained, the routing changes we observe represent *incremental* effects of additional RL, not the full effect of RL versus a pre-trained base. This makes our findings conservative: even moderate additional RL produces measurable domain-selective routing shifts.

PRM confound. Because M2.5 introduces process reward models (PRMs) alongside scaling, the routing changes we observe reflect the joint effect of more RL environments *and* dense, step-by-step supervision. PRMs provide deterministic, per-step credit assignment that fundamentally alters gradient structure compared to sparse outcome rewards. The -0.23 bit code concentration may be partly an artifact of PRMs forcing the model into highly specific logical pathways rather than a pure effect of RL scaling. We cannot disentangle these factors without intermediate checkpoints or ablations that isolate PRM contribution.

2.4 DeepSeek V3: Cross-Architecture Comparison Target

To test whether our findings generalize beyond a single model family, we also analyze DeepSeek V3 [DeepSeek-AI, 2024b], a 671-billion-parameter MoE model with a fundamentally different routing mechanism. DeepSeek V3 has 61 transformer layers: layers 0–2 use dense MLPs, and layers 3–60 use MoE with 256 routed experts plus 1 shared expert per layer, with $\sim$37B activated parameters per token.

The key architectural distinction is **grouped routing**. Rather than selecting top-8 from 256 experts directly (as in MiniMax), DeepSeek V3 uses a two-stage process:

1. **Group selection.** Compute sigmoid scores $\mathbf{s} = \sigma(\mathbf{x}\mathbf{W}_g^\top) + \mathbf{b}_{\text{corr}}$ over all 256 experts, partition into $G = 8$ groups of 32, sum each group's top-2 expert scores to obtain a group score, and select the top-4 groups, yielding 128 candidate experts.
2. **Expert selection.** From these 128 candidates, select the top-8 experts.

This two-stage selection constrains the routing search space: experts can only be selected if their *group* is first selected. As we show in Section 7, this constraint acts as an architectural stabilizer that dampens routing changes under post-training.

DeepSeek V3 also uses Multi-head Latent Attention (MLA) instead of standard grouped-query attention, and employs an auxiliary-loss-free load balancing strategy via the correction bias $\mathbf{b}_{\text{corr}}$. We compare V3-Base (the pre-trained model) against V3 (after SFT and RL post-training), providing a contrast where the training gap is larger (pre-trained $\rightarrow$ post-trained) but the architecture imposes stronger routing constraints.

3 Experimental Setup

3.1 Multi-Domain Evaluation Corpus

We construct a corpus of 900 samples (150 per domain) spanning six categories designed to cover both domains within and outside the RL training distribution:

Table 1: Corpus composition. Domains span structured (code, math) to open-ended (general, instruct) inputs. Token counts reflect 150 samples per domain with max 4,096 tokens per sample.

Domain	Dataset	Tokens	Description
Code	CodeParrot-clean-valid	216,075	Full Python source files
Math	GSM8K	23,085	Chain-of-thought solutions
Knowledge	HLE (Humanity's Last Exam)	9,825	Expert-level factual QA
Reasoning	ARC-Challenge	9,273	Science reasoning problems
General	WikiText-103	434,353	Concatenated encyclopedia articles
Instruct	UltraChat	154,548	Multi-turn conversations
Total		**847,159**	

Each sample is tokenized using the MiniMax tokenizer with a maximum length of 4,096 tokens. Per-token domain labels and sample boundaries are preserved for domain-stratified analysis. For the DeepSeek cross-architecture comparison (Section 7), the same 900 samples are tokenized using the DeepSeek tokenizer, yielding 861,531 tokens (vs. 847,159 for MiniMax due to tokenizer differences).

3.2 Full Forward Pass Recording

We implement a custom forward pass that processes the entire tokenized corpus through all 62 transformer layers, faithfully reproducing the model's inference computation:

1. Token embeddings are looked up and processed sequentially through all layers.
2. At each layer, causal multi-head attention with RoPE embeddings [Su et al., 2024] ($\theta = 5 \times 10^6$) is applied.
3. The routing function (Eq. 2) selects top-8 experts per token. We record the selected expert indices as an `int16` array of shape $(847{,}159, 8)$ per layer.
4. Expert FFN outputs are weighted and accumulated into the residual stream via in-place `index_add_`.
5. Hidden states are saved at layers 0, 20, 40, and 60 for representation analysis.

The observational analysis was executed on NVIDIA B200 GPUs (192 GB HBM3e). Total runtime: 18.7 minutes (M2.5) + 20.4 minutes (M2.1).

3.3 Gate-Swap Intervention Design

To move beyond correlational analysis and test whether routing changes *cause* capability differences, we design a gate-swap experiment inspired by causal intervention methods in neuroscience [Vig et al., 2020] and interpretability [Meng et al., 2022]. The key insight is that MoE routing depends on only two small parameter groups per layer—the gate weight matrix $\mathbf{W}_g \in \mathbb{R}^{256\times 3072}$ and the correction bias $\mathbf{b}_{\text{corr}} \in \mathbb{R}^{256}$—which together occupy $\sim$1.5 MB per layer ($\sim$93 MB total for all 62 layers), less than 0.02% of the model's total parameter count. All other parameters (attention weights, expert FFN weights, layer norms, embeddings, LM head) remain fixed.

We define four experimental conditions crossing *body model* (source of all non-gate parameters) with *gate source* (source of $\mathbf{W}_g$ and $\mathbf{b}_{\text{corr}}$):

1. **M2.5 full**: M2.5 body + M2.5 gate (scaled-RL baseline)
2. **M2.1 full**: M2.1 body + M2.1 gate (moderate-RL baseline)
3. **M2.1 + M2.5 gate**: M2.1 body with M2.5's routing (does scaled-RL routing help M2.1 experts?)
4. **M2.5 + M2.1 gate**: M2.5 body with M2.1's routing (does moderate-RL routing help M2.5 experts?)

Each condition runs a complete 62-layer forward pass over the identical 847,159-token corpus and computes per-domain cross-entropy loss against the next-token targets. The LM head always comes from the body model, ensuring that only routing decisions are transplanted. The experiment was executed on an NVIDIA B200 GPU with 192 GB HBM3e. Each condition completes in 7.6–10.7 minutes; total wall-clock time for all four conditions: 37 minutes.

Causal logic. If routing changes cause domain-specific improvement, then:

- Condition 3 should *improve* M2.1's code perplexity (M2.5's more specialized routing helps M2.1's experts)
- Condition 4 should *degrade* M2.5's code perplexity (reverting to M2.1's less specialized routing hurts)

If routing is epiphenomenal (merely reflecting expert weight changes), then gate swaps should have minimal effect on perplexity; only the body model should matter.

3.4 Metrics

Shannon entropy. For each layer ℓ and (optionally) domain d, we compute the entropy of the expert frequency distribution:

$$H_\ell^{(d)} = -\sum_{i=1}^{256} p_i^{(d)} \log_2 p_i^{(d)} \tag{5}$$

where $p_i^{(d)}$ is the fraction of domain-d tokens routed to expert i at layer ℓ. Maximum entropy (uniform routing) is $\log_2(256) = 8.0$ bits.

Jaccard similarity. Per-token routing agreement between M2.5 and M2.1:

$$J(\mathbf{x}) = \frac{|\mathcal{E}_{2.5}(\mathbf{x}) \cap \mathcal{E}_{2.1}(\mathbf{x})|}{|\mathcal{E}_{2.5}(\mathbf{x}) \cup \mathcal{E}_{2.1}(\mathbf{x})|} \tag{6}$$

averaged across all tokens per layer.

Domain-expert affinity. The deviation of expert i's frequency on domain d from its global frequency:

$$a_{d,i} = f_{d,i} - f_{\text{global},i} \tag{7}$$

Bootstrap confidence intervals. All entropy and Jaccard comparisons use 1,000 bootstrap resamples with 95% percentile confidence intervals, parallelized across 16 CPU workers.

Cross-entropy loss. For the gate-swap experiment, we compute per-token cross-entropy loss $\mathcal{L}(\mathbf{x}_t) = -\log p(x_{t+1} \mid x_{\leq t})$ using chunked logit projection over the 200,064-token vocabulary, then aggregate per domain. Perplexity is reported as $\text{PPL} = \exp(\bar{\mathcal{L}})$.

4 Results: Observational Analysis

4.1 Finding 1: RL Preserves Expert Identity but Redistributes Routing

Figure 1 shows that both models maintain the same overall routing structure (the same experts are broadly active at the same layers) but M2.5 exhibits more extreme frequency values. The aggregate statistics (Table 2) reveal an apparent paradox: M2.5 has slightly *fewer* active experts per layer yet *higher* aggregate entropy. This resolves under domain-stratified analysis (Section 4.3): M2.5 concentrates certain experts heavily on specific domains (reducing effective expert count within domains) while distributing other tokens more broadly (increasing aggregate entropy).

Table 2: Aggregate routing statistics across all 62 layers.

Metric	M2.5 (scaled RL)	M2.1 (moderate RL)
Active experts/layer	254.1 ± 5.8	254.8 ± 3.4
Entropy (bits)	7.813 ± 0.112	7.748 ± 0.101
Max expert frequency	0.5009	0.4271
Min expert frequency	10^{-6}	2.5×10^{-5}

This pattern (same overall routing structure, redistributed frequencies) is consistent with post-training modifying the gating network's decision boundaries rather than fundamentally altering expert computations. The gate-swap experiment in Section 5 directly tests this interpretation.

4.2 Finding 2: Routing Divergence Is Depth-Dependent and Monotonically Increasing

Figure 2 establishes the depth-dependent nature of RL's routing impact. Three observations:

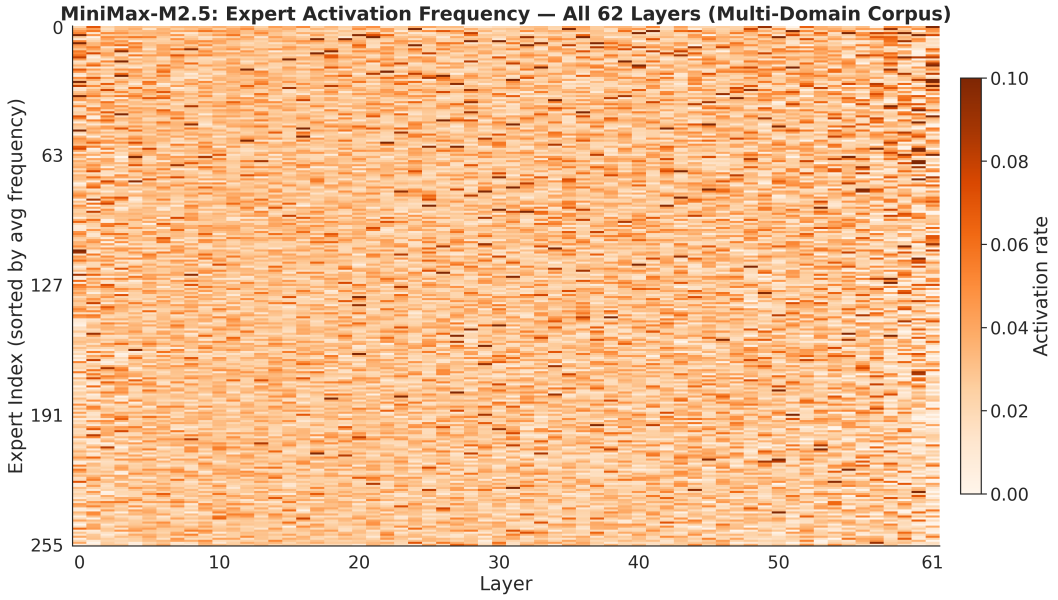
1. **Monotonic Jaccard decline**: From $J = 0.68$ at layer 5 to $J = 0.42$ at layer 61, routing agreement decays smoothly. At the final MoE layer, fewer than half of routing decisions are shared between models.
2. **Overlap preservation**: Despite declining Jaccard, the mean overlap ratio remains 0.72 ($\approx 5.8/8$ experts shared), indicating that most routing changes involve swapping 2–3 experts rather than wholesale reassignment.
3. **Sharp divergence escalation**: The per-layer divergence bar in Figure 2a shows an inflection point around layer 45, with the final 15 layers accounting for a disproportionate share of total divergence.

This pattern is consistent with a hierarchical view of transformer processing: shallow layers extract universal features (syntax, tokenization) that are stable across training variants, while deep layers encode task-specific reasoning patterns that RL explicitly optimizes.

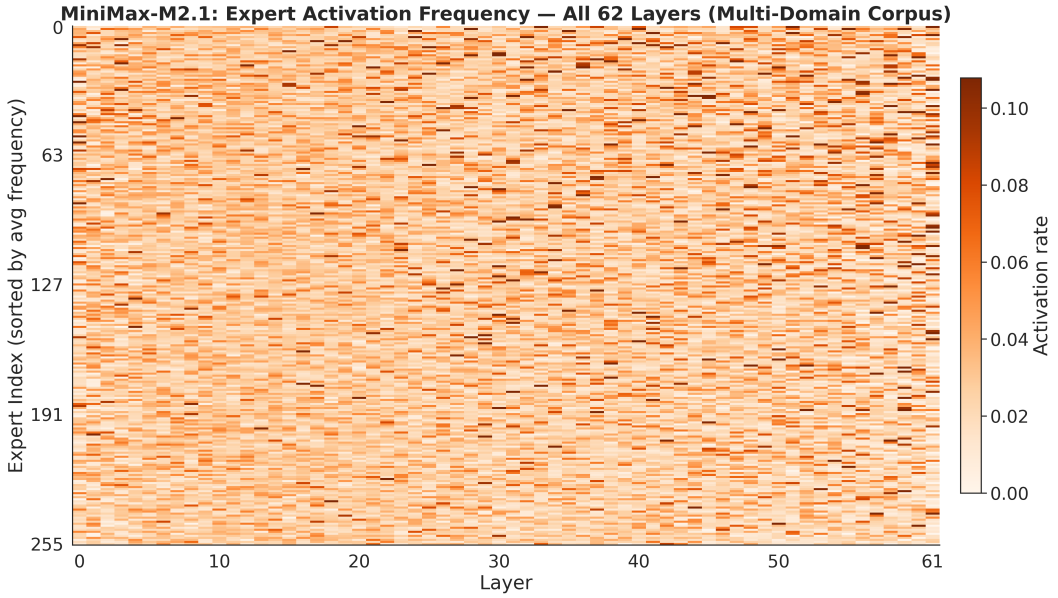
4.3 Finding 3: RL-Induced Routing Changes Are Domain-Selective

Figure 3 presents the central observational finding. Table 3 provides exact values.

The asymmetry is striking. Post-training *specializes* code routing (-0.23 bits, a 2.9% reduction from the 8-bit maximum) while *generalizing* routing for conversational and reasoning inputs ($+0.11$ to $+0.17$ bits). Under sample-level bootstrap, math also shows a small but significant positive effect ($+0.012$ bits), while knowledge remains non-significant. The magnitudes differ by an order of magnitude between concentration (code) and dispersion (general/instruct), suggesting qualitatively different optimization dynamics.



(a) M2.5 (RL-tuned)



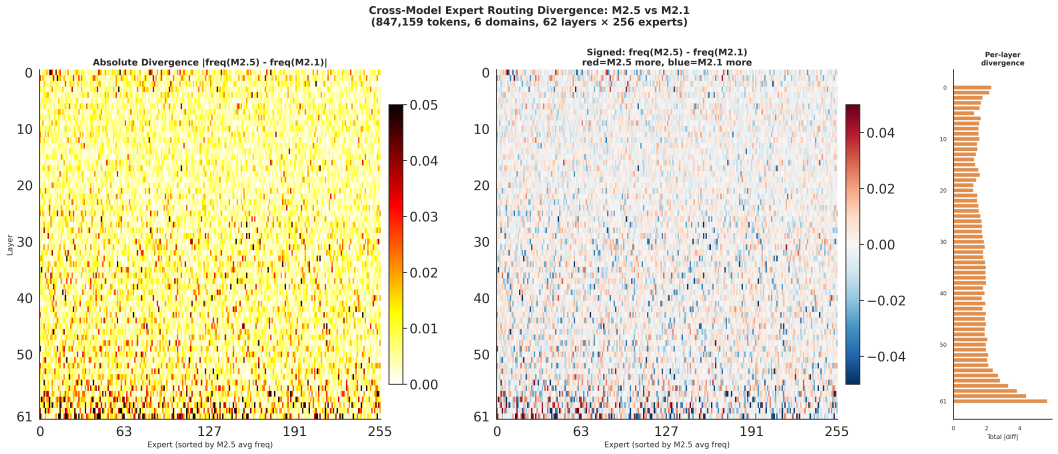
(b) M2.1 (moderate RL)

Figure 1: **Expert activation frequency heatmaps** (62 layers $\times$ 256 experts). Both models show structured routing with vertical banding (persistent expert preferences) and horizontal gradients (layer-dependent shifts). M2.5 exhibits sharper contrast: frequency range $[10^{-6}, 0.50]$ vs. M2.1's $[2.5 \times 10^{-5}, 0.43]$. RL widens the dynamic range without changing the number of active experts (M2.5: 254.1 ± 5.8; M2.1: 254.8 ± 3.4).

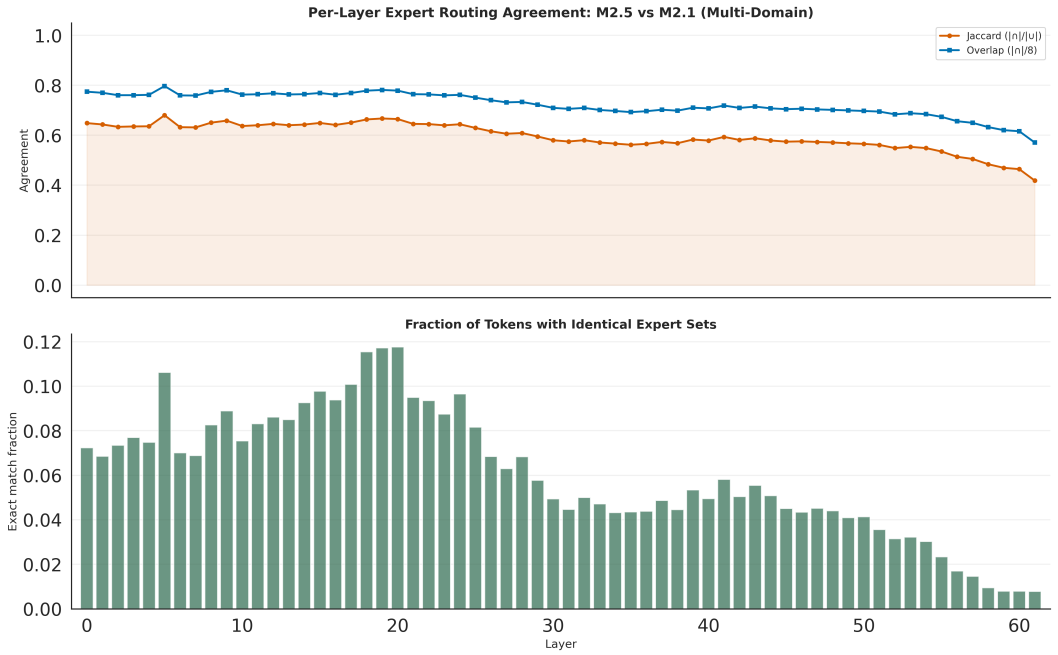
4.4 Finding 4: Domain Selectivity Is Consistent with Reward Signal Structure

The direction of routing change is *consistent with* the verifiability of each domain's reward signal during post-training:

- **Code** has binary, verifiable rewards (pass/fail tests). Post-training concentrates routing onto specialized code experts (-0.23 bits), the largest magnitude change.



(a) Cross-model routing divergence



(b) Layer-wise routing agreement

Figure 2: **RL's impact on routing increases monotonically with depth. (a)** Absolute frequency divergence between M2.5 and M2.1 across all 62 layers and 256 experts. Divergence escalates sharply in the final 15 layers. **(b)** Three agreement metrics: Jaccard similarity (strictest), overlap ratio (how many of 8 experts are shared), and exact match rate. Mean Jaccard $= 0.595$; best layer 5 ($J = 0.68$), worst layer 61 ($J = 0.42$).

- **Reasoning and instruction-following** have subjective, noisy rewards (human preference). Post-training disperses routing across more experts ($+0.11$ to $+0.17$ bits).
- **Math** has verifiable answers but diverse solution paths. Post-training produces a small but significant positive change ($+0.012$ bits), suggesting slight dispersion.
- **Knowledge** (Humanity's Last Exam) shows no significant change, consistent with this domain being outside the post-training distribution.

Figure 3 confirms this pattern holds across individual layers.

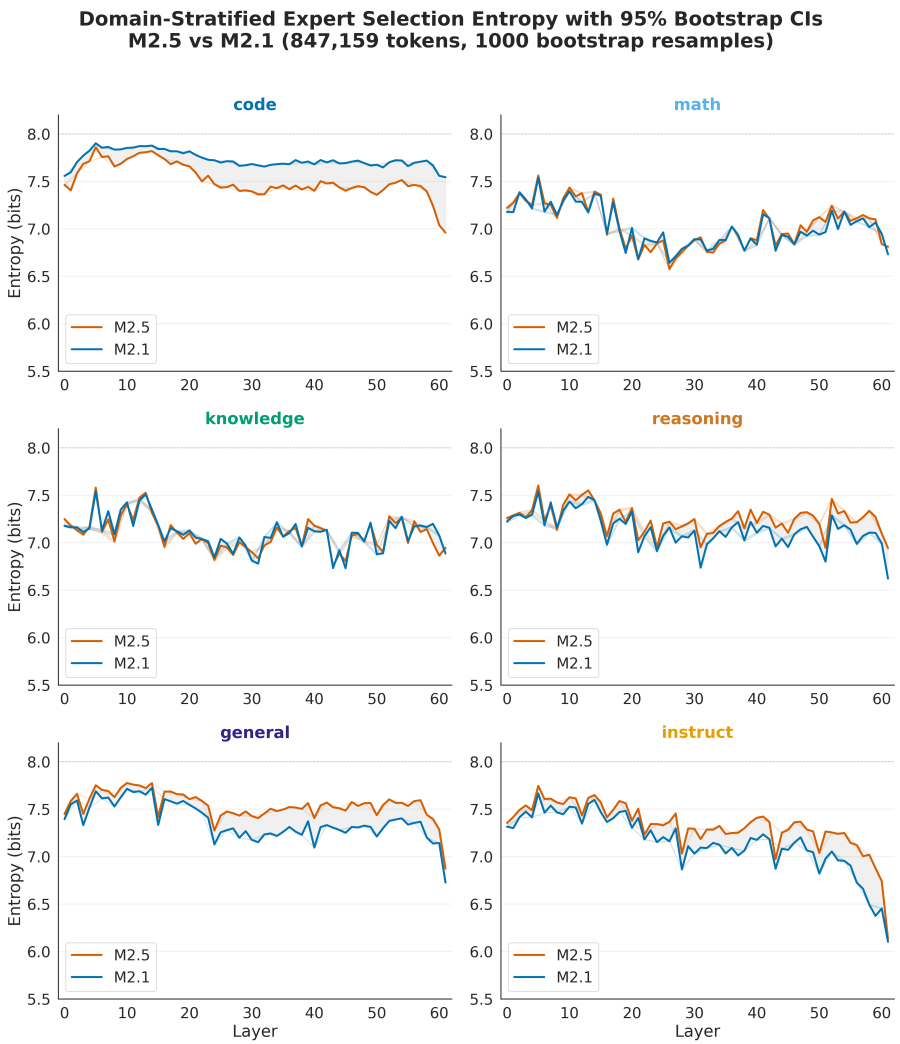


Figure 3: **Domain-stratified entropy difference** ($\Delta H = H_{\text{M2.5}} - H_{\text{M2.1}}$) with per-layer curves and 95% bootstrap confidence intervals (1,000 resamples). Code shows strong entropy *decrease* (sharper routing), while reasoning, general, and instruct show significant entropy *increase* (broader routing). Math and knowledge show no significant change. The gap shading shows the signed entropy difference between M2.5 and M2.1 at each layer.

Figure 4 provides complementary evidence: per-domain Jaccard agreement inversely correlates with entropy shift magnitude.

Alternative explanations. We emphasize that the reward-signal alignment is a *hypothesis consistent with the data*, not a demonstrated causal mechanism. Several alternative explanations could produce similar patterns:

- **Input regularity**: Code has lower lexical diversity and more regular syntax than general text. Routing concentration may reflect input structure rather than reward properties.
- **Training data composition**: If post-training data was disproportionately weighted toward code, the stronger signal could explain concentration independently of reward type.
- **SFT confound**: M2.5 underwent SFT before RL. The routing changes we observe are the joint effect of SFT + RL; the SFT stage alone could account for some of the instruction-domain dispersion.

Table 3: Domain-stratified entropy difference ($\Delta H = H_{\text{M2.5}} - H_{\text{M2.1}}$) with sample-level bootstrap 95% CIs (resampling 150 samples/domain, 1,000 resamples).

Domain	ΔH (bits)	95% CI	Sig.	Direction
Code	-0.229	$[-0.241, -0.216]$	$***$	Concentrates
Math	$+0.012$	$[+0.007, +0.018]$	$***$	Slight dispersion
Knowledge	-0.009	$[-0.019, +0.001]$	n.s.	Unchanged
Reasoning	$+0.107$	$[+0.099, +0.116]$	$***$	Disperses
General	$+0.168$	$[+0.162, +0.175]$	$***$	Disperses
Instruct	$+0.173$	$[+0.169, +0.178]$	$***$	Disperses

Distinguishing these hypotheses requires either intermediate checkpoints (isolating SFT from RL) or controlled experiments on a second model family. We partially address this in Section 7 by testing whether the domain-selective pattern replicates on DeepSeek V3, which uses a fundamentally different routing mechanism. We present the reward-selective hypothesis as a falsifiable prediction, not as an established finding.

4.5 Finding 5: Expert Migration Shows Domain-Selective Role Reassignment

Figure 5 characterizes expert specialization in M2.5 at the peak specialization layer (layer 61):

Table 4: Domain specialization strength at layer 61 (M2.5). Top-5 experts and their affinity scores (deviation from uniform frequency).

Domain	Peak Affinity	Characterization
Math	$+0.90$	Strongest specialization
Knowledge	$+0.84$	Strong specialization
Reasoning	$+0.75$	Strong specialization
Code	$+0.53$	Moderate specialization
Instruct	$+0.53$	Moderate specialization
General	$+0.26$	Weak specialization

The PCA projection reveals cross-domain experts: Expert 247 appears in the top-5 specialists for math, knowledge, *and* reasoning, acting as a cross-domain "analytical reasoning" module. Expert 101 spans math and code, suggesting shared computational reasoning pathways.

The RL-induced expert migration analysis (Figure 5b) shows:

- Mean PCA displacement: 0.0034; max displacement: 0.0157 (Expert 211)
- **56 of 256 experts (21.9%) changed their primary domain affiliation**

This substantial reorganization confirms that RL does not merely adjust gating thresholds; it fundamentally restructures which experts handle which types of input. But does this restructuring *cause* capability improvements, or is it an epiphenomenon of expert weight changes? Section 5 addresses this directly.

5 Causal Test: The Gate-Swap Experiment

The observational analysis in Section 4 establishes that post-training produces robust, domain-selective routing changes. However, the critical question remains: are these routing changes *functionally important*, or do they merely reflect changes in the underlying expert representations? To answer this, we perform the gate-swap intervention described in Section 3.3.

5.1 Finding 6: Routing Changes Are Structurally Real but Functionally Secondary

Table 5 and Figure 6 present the complete results.

Three key findings emerge from the gate-swap experiment:

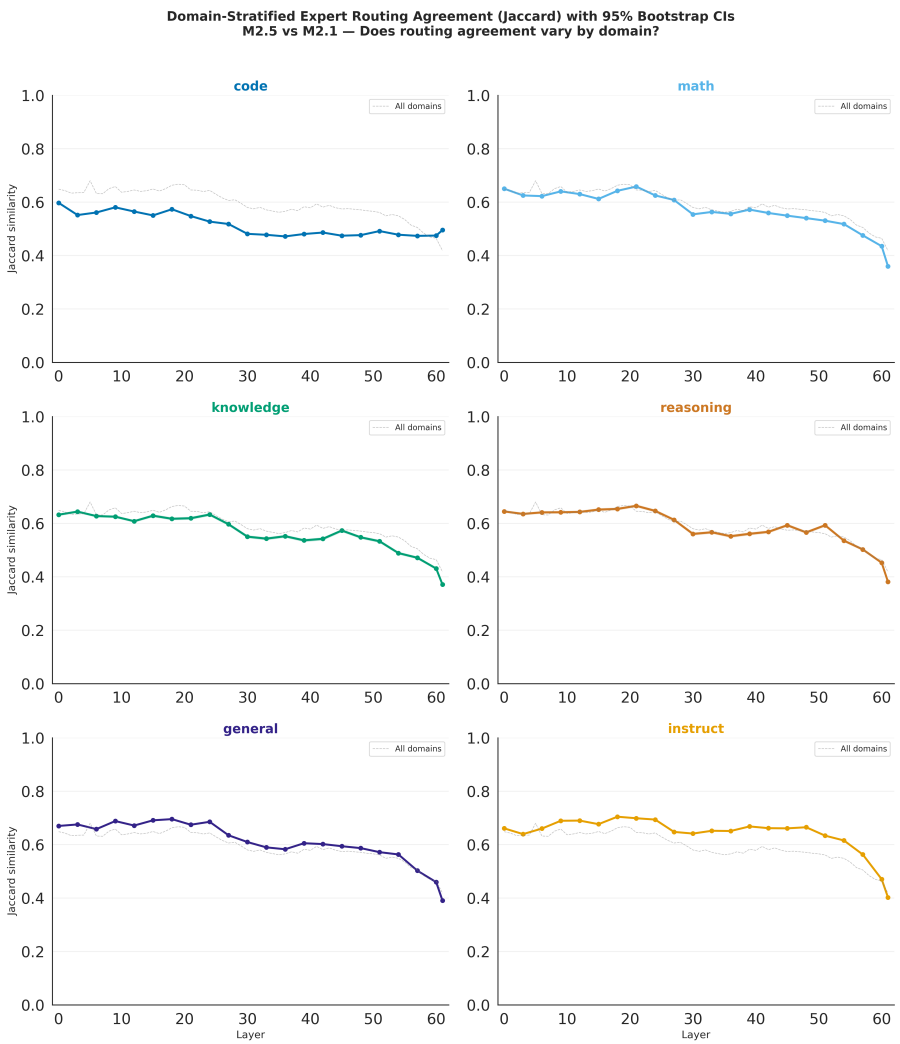


Figure 4: **Domain-stratified Jaccard agreement with bootstrap CIs.** Domains with minimal entropy impact (math, knowledge) show higher cross-model routing agreement, confirming preserved routing. Domains with large entropy shifts (code, general, instruct) show lower agreement.

(i) Expert weights dominate; routing is secondary. Code perplexity is virtually unchanged by gate swap in either direction: 2.04 $\rightarrow$ 2.04 when M2.1 receives M2.5's routing ($\Delta = -0.1\%$), and 2.12 $\rightarrow$ 2.13 when M2.5 receives M2.1's routing ($\Delta = +0.5\%$). Despite the observational finding that code routing entropy decreases by 0.23 bits under scaled RL (Section 4.3), transplanting the more specialized routing function to M2.1 produces no code improvement. The functional code capability resides in the expert FFN weights, not the gate parameters.

Scope of this claim. Our causal evidence is based entirely on next-token cross-entropy loss (perplexity). RLHF, CISPO, and process reward signals do not optimize for perplexity; they optimize for trajectory-level rewards and human preference. RL often *worsens* perplexity (the alignment tax) while improving generation quality. Showing that code perplexity changes by $<1\%$ under a gate-swap does not prove that code *generation capability* resides entirely in the weights. Auto-regressive generation over long horizons requires sustained structural coherence that teacher-forced single-token evaluation cannot capture: a swapped gate might yield similar token probabilities but fail to maintain the state machine required for long-horizon code generation. The broader claim of functional capability should be suspended pending downstream task evaluations (HumanEval, GSM8K) described in Section 9.

Table 5: Per-domain cross-entropy perplexity across four gate-swap conditions. *Body* = source of all non-gate parameters; *Gate* = source of $\mathbf{W}_g$ and $\mathbf{b}_{\text{corr}}$. Hatched conditions are gate-swaps. 847,159 tokens, 6 domains. PPL = $\exp(\bar{\mathcal{L}})$; lower is better.

Condition	Code	Math	Know.	Reas.	Gen.	Instr.	Overall
M2.5 full (scaled-RL baseline)	2.12	3.56	9.20	8.46	9.20	3.75	5.62
M2.1 full (moderate-RL baseline)	2.04	3.65	6.46	8.38	6.87	3.49	4.61
M2.1 body + M2.5 gate	2.04	3.67	6.62	8.66	7.17	3.54	4.73
M2.5 body + M2.1 gate	2.13	3.59	8.89	8.28	8.48	3.74	5.38

(ii) Scaled RL routing over-specializes, taxing general domains. The most striking result is asymmetric: when M2.5 receives M2.1's routing, general-domain perplexity *improves* by 7.8% (9.20 $\rightarrow$ 8.48) and knowledge improves by 3.4% (9.20 $\rightarrow$ 8.89). This means that the additional RL-induced routing changes actively *harm* non-target domains. The overall perplexity of M2.5 + M2.1 gate (5.38) is lower than M2.5 full (5.62), indicating that *M2.1's less specialized routing function is strictly better for M2.5's expert weights on average*. The additional routing specialization from scaled RL is not free; it imposes a regulatory tax on domains outside the expanded RL reward distribution.

(iii) Routing and weights are tightly co-adapted. Both swap conditions are slightly worse than their matched baselines on most domains, consistent with gate-body co-adaptation: each model's routing function is optimized for its own expert weights. However, the co-adaptation is *asymmetric*: M2.1's routing transfers well to M2.5's body (small degradation on target domains, significant improvement on non-target domains), while M2.5's routing transfers poorly to M2.1's body (slight degradation across the board). This asymmetry suggests that scaled RL creates a more specialized, less transferable routing configuration.

5.2 Verification: Gates Do Produce Different Routing

To confirm that the gate swap actually changes routing decisions (rather than being absorbed by the hidden state representation), we measure per-token Jaccard similarity between M2.5's baseline routing and its routing under M2.1's gates:

Table 6: Routing divergence under gate swap (first 1,000 tokens). Jaccard similarity between M2.5 baseline routing and M2.5 + M2.1 gate routing at diagnostic layers.

Layer	Jaccard	Interpretation
0	0.790	21% of routing decisions differ
30	0.678	32% differ
61	0.566	43% differ

At the deepest layer, 43% of expert selections change under gate swap (Table 6), yet code perplexity moves by only 0.5%. This dissociation, in which large routing changes produce minimal functional impact, is the strongest evidence that routing is not the primary mechanism of RL-induced capability.

5.3 Decomposition: Where Does Capability Reside?

The gate-swap results enable a rough decomposition of RL's effect into routing and weight components. For any domain d, define:

$$\Delta_{\text{total}}(d) = \text{PPL}_{\text{M2.5}}(d) - \text{PPL}_{\text{M2.1}}(d) \tag{8}$$

$$\Delta_{\text{routing}}(d) = \text{PPL}_{\text{M2.1+M2.5gate}}(d) - \text{PPL}_{\text{M2.1}}(d) \tag{9}$$

$$\Delta_{\text{weight}}(d) \approx \Delta_{\text{total}}(d) - \Delta_{\text{routing}}(d) \tag{10}$$

Table 7 shows that for every domain, the weight component dominates. For code (the primary RL target), routing contributes $\approx$0% of the total effect. For domains where RL increases perplexity

Table 7: Decomposition of RL's perplexity effect into routing and weight components (ΔPPL).

Domain	Δ_{total}	Δ_{routing}	Δ_{weight}	Routing %	Weight %
Code	$+0.08$	≈ 0	$+0.08$	$\approx 0\%$	$\approx 100\%$
Math	-0.09	$+0.02$	-0.11	neg.	$> 100\%$
Knowledge	$+2.74$	$+0.16$	$+2.58$	6%	94%
Reasoning	$+0.08$	$+0.28$	-0.20	neg.	—
General	$+2.33$	$+0.31$	$+2.02$	13%	87%
Instruct	$+0.26$	$+0.04$	$+0.22$	15%	85%

(knowledge, general), routing accounts for 6–15% of the degradation, with expert weight changes responsible for the remainder. The "negative" routing percentages for math and reasoning indicate that RL routing slightly harms these domains, while weight changes compensate.

Mathematical caveat. This decomposition is a first-order linear approximation. In an MoE, the layer output is a *multiplicative* interaction between sigmoid gate scores and expert FFN outputs: $\mathbf{y} = \sum_i \bar{w}_i \cdot \text{Expert}_i(\mathbf{x})$. Co-adaptation implies that gate scores and expert computations are jointly optimized, creating non-linear interaction terms that an additive decomposition cannot capture. When M2.5's gates operate on M2.1's experts (or vice versa), the resulting behavior reflects not just the sum of routing and weight changes but also the *geometric misalignment* between new decision boundaries and old expert representations. The failure modes of cross-model gate transplantation are likely dominated by this non-linear interaction term. The decomposition in Table 7 should therefore be interpreted as a heuristic guide to the relative importance of routing vs. weight changes, not as an exact accounting.

6 Robustness and Token-Level Analysis

6.1 Subsampled Code Validation

The code domain contains 216,075 tokens, 23× more than the smallest domain (reasoning, 9,273 tokens). To rule out corpus imbalance as a driver of the -0.23 bits code effect, we repeatedly subsample code tokens to $n = 9{,}273$ (matching reasoning) and recompute ΔH across 50 random subsamples.

Figure 7 shows that the code concentration effect survives subsampling. The per-layer pattern (left panel) is preserved with slightly wider confidence bands, confirming that the effect is not inflated by the larger code sample size.

6.2 Sample-Level Bootstrap

Our primary bootstrap (Section 3.4) resamples individual tokens, which are not independent within a sample (up to 4,096 tokens share context). We therefore also compute a conservative sample-level bootstrap that resamples the 150 samples per domain, accounting for within-sample dependence.

Figure 8 confirms that all significant effects survive the sample-level bootstrap with wider but still zero-excluding CIs. The non-significant results (math, knowledge) also hold: their CIs span zero under both procedures.

6.3 Token-Level Drill-Down: What Drives Code Concentration?

To move beyond domain-level aggregation, we classify individual code tokens into syntactic categories (keywords, operators, identifiers, whitespace, numbers, strings, etc.) and compute ΔH for each category independently.

Figure 9 reveals that code routing concentration is not driven uniformly by all code tokens. Whitespace ($\Delta H = -0.30$), brackets (-0.27), and operators (-0.24) show the strongest concentration, while comments (-0.13) and numbers (-0.14) show the weakest. Identifiers (the largest category at 43% of code tokens) show moderate concentration (-0.18). This decomposition suggests that

the concentration effect is strongest for tokens with regular, predictable structure—consistent with either the reward-signal hypothesis (structured tokens produce deterministic outputs) or the input-regularity hypothesis (syntactically regular tokens naturally route to fewer experts).

7 Cross-Architecture Validation: DeepSeek V3

To test whether the routing phenomena observed in MiniMax generalize across architectures, we replicate the observational analysis on DeepSeek V3 [DeepSeek-AI, 2024b], comparing V3-Base (pre-trained) against V3 (SFT+RL). This comparison differs from MiniMax in two important ways: (1) the training gap is larger (pre-trained $\rightarrow$ post-trained, vs. RL $\rightarrow$ scaled RL), and (2) the routing mechanism is fundamentally different (grouped two-stage selection vs. direct top-k). These two differences create an informative asymmetry: if DeepSeek's *larger* training gap produces *less* routing change than MiniMax's incremental RL, the architectural stabilizer effect must be strong enough to overcome the larger perturbation. The full forward pass over all 58 MoE layers (layers 3 through 60 inclusive; layers 0–2 are dense) was executed on NVIDIA B200 GPUs using 861,531 tokens from the same six-domain corpus.

7.1 Finding 7: Routing Changes Are Universal but Architecture-Modulated

Figure 10 establishes the core finding: post-training changes expert routing in both architectures, but the magnitude depends on routing mechanism design. Table 8 provides the full comparison.

Table 8: Cross-architecture routing comparison. DeepSeek V3 has 58 MoE layers (3–60); MiniMax has 62 (all MoE). Both have 256 experts and top-8 routing.

Metric	**MiniMax (M2.1$\rightarrow$M2.5)**	**DeepSeek (V3-Base$\rightarrow$V3)**
Mean Jaccard (top-8)	0.595	0.672
Mean top-1 agreement	0.603	0.846
Mean expert overlap	5.8/8	6.2/8
Jaccard range	0.42–0.68	0.58–0.89
Active experts/layer	254.1 ± 5.8	256.0 ± 0.3
Entropy (post-trained)	7.813 ± 0.112 bits	7.894 ± 0.039 bits
Mean ΔH	$+0.065$ bits	$+0.003$ bits
L1 frequency divergence	1.94	0.98

Three aspects of DeepSeek's routing stability are striking. First, **top-1 agreement is remarkably high**: the single highest-scoring expert is preserved for 85% of tokens across all layers, compared to 60% for MiniMax. Second, **entropy is nearly unchanged**: the mean ΔH of $+0.003$ bits is $22\times$ smaller than MiniMax's $+0.065$ bits, with per-layer variance also $3\times$ lower. Third, **all 256 experts remain active** at every layer (256.0 ± 0.3), whereas MiniMax shows slight expert dropout (254.1 ± 5.8). This suggests that DeepSeek's grouped routing and auxiliary-loss-free balancing maintain expert utilization more effectively under post-training.

These results are especially notable given the asymmetry in training gap: DeepSeek's comparison spans a *larger* perturbation (pre-trained $\rightarrow$ SFT+RL) than MiniMax's (RL $\rightarrow$ scaled RL), yet DeepSeek exhibits *less* routing change by every metric. This rules out the naïve explanation that DeepSeek is simply more stable because its models are "closer together" in training; the opposite is true. The architectural explanation (grouped routing constraining the search space) is the most parsimonious account. An alternative possibility (that SFT+RL produces a coherent routing configuration that the base model lacks, so routing "snaps into place" rather than drifting) would predict *lower* agreement at early layers and higher at late layers, which we do not observe.

7.2 Finding 8: Grouped Routing Acts as an Architectural Stabilizer

DeepSeek's two-stage routing enables a unique decomposition (Figure 11): we can separately measure whether post-training changes *which groups* are selected (stage 1) or *which experts within groups* are selected (stage 2). The group-level Jaccard (mean 0.82) substantially exceeds the expert-level Jaccard (mean 0.67), indicating that the majority of routing changes occur at the within-group

level: post-training reshuffles experts within the same group rather than redirecting tokens to entirely different groups.

To quantify this, we decompose the L1 frequency divergence into group-level and within-group components. Across all MoE layers, **73.5% of total routing divergence occurs within preserved groups**, with only 26.5% attributable to group-level reallocation. This means the two-stage routing mechanism constrains the "radius" of routing changes: post-training can move tokens between nearby experts (same group) more easily than between distant experts (different groups), effectively acting as a structural regularizer on the co-adaptation process.

Group-level entropy is nearly saturated and unchanged: 2.993 ± 0.006 bits for V3 vs. 2.993 ± 0.005 bits for V3-Base (maximum $= \log_2 8 = 3.0$ bits), confirming that the group selection stage is almost uniformly distributed and unaffected by post-training.

Algorithmic stabilizers beyond architecture. The architectural explanation is likely incomplete. DeepSeek V3's routing stability may also reflect its *algorithmic* design choices: (1) auxiliary-loss-free load balancing, which dynamically updates $\mathbf{b}_{\text{corr}}$ during training to enforce expert balance without a differentiable loss penalty, effectively preventing the router from drifting; and (2) GRPO (Group Relative Policy Optimization), which incorporates KL-divergence penalties that constrain policy updates. These algorithmic mechanisms operate independently of the grouped routing geometry and may physically prevent routing drift. The near-zero entropy change ($\Delta H = +0.003$ bits) is likely the product of grouped routing *and* aggressive non-differentiable bias correction acting in concert, rather than either mechanism alone. Disentangling architectural from algorithmic stabilization would require testing grouped routing with standard RL (no auxiliary-loss-free balancing) or flat routing with DeepSeek's balancing algorithm.

7.3 Finding 9: Domain-Selective Specialization Is Architecture-Dependent

The most revealing cross-architecture contrast is in domain-stratified entropy (Figure 12 and Table 9). While MiniMax shows dramatic domain-selective specialization—code concentrates ($\Delta H = -0.22$ bits), general/instruct disperse ($+0.17$ bits)—DeepSeek shows uniformly small positive ΔH across all domains, with no evidence of asymmetric specialization.

Table 9: Domain-stratified mean ΔH (post-trained − base) and mean top-1 agreement for both architectures. DeepSeek shows no domain-selective pattern.

	Mean ΔH (bits)		**Mean Top-1 Agreement**	
Domain	**MiniMax**	**DeepSeek**	**MiniMax**	**DeepSeek**
Code	-0.218	$+0.025$	0.523	0.827
Math	$+0.017$	$+0.114$	0.574	0.810
Knowledge	-0.010	$+0.120$	0.557	0.788
Reasoning	$+0.110$	$+0.065$	0.592	0.820
General	$+0.168$	$+0.017$	0.626	0.852
Instruct	$+0.168$	$+0.032$	0.658	0.870

This contrast has two implications. First, the reward-selective specialization hypothesis (Section 4.4)—that verifiable rewards produce routing concentration while subjective rewards produce dispersion—does *not* hold for DeepSeek. This suggests the hypothesis is *architecture-dependent*: grouped routing may prevent the kind of dramatic domain-selective reshaping that direct routing permits, because the group-selection constraint limits how far routing can deviate from the pre-trained configuration. Second, the universal finding across both architectures is that routing changes are **monotonically increasing with depth**: both models show highest agreement at early layers and lowest at deep layers, consistent with the hierarchical feature extraction view (shallow = universal, deep = task-specific).

8 Discussion

8.1 From Correlation to Causation: The Co-Adaptation Picture

The gate-swap experiment transforms our understanding of the observational findings. The domain-selective routing changes documented in Section 4—code concentration ($\Delta H = -0.23$ bits), reasoning/general dispersion ($+0.11$ to $+0.17$ bits), expert migration (21.9%), depth-dependent divergence—are *real structural changes* to the routing network. However, they are not the primary *mechanism* of capability improvement. Expert weight changes account for $\geq$85% of RL's perplexity effect on every domain (Table 7), while routing contributes $\leq$15% and is slightly counterproductive on several domains.

This dissociation between structural change and functional importance has a natural interpretation: during post-training, the gradient signal updates both gate and expert parameters simultaneously. Expert weights learn domain-specialized computations; gate weights track these changes to maintain coherent routing. But because both are updated jointly, the gate adapts to the *current* expert landscape, not to an optimal routing strategy. The result is a co-adapted configuration where routing reflects expert specialization but does not independently drive it.

8.2 Routing as a Regulatory Tax

The most surprising finding is that the additional RL routing actively harms non-target domains. When M2.5 receives M2.1's routing (Condition 4), its overall perplexity *improves* from 5.62 to 5.38—a 4.3% reduction, driven by large gains on general (-7.8%) and knowledge (-3.4%). This means M2.1's less specialized routing function is a better match for M2.5's expert weights than M2.5's own routing, in aggregate.

We interpret this as a *routing tax*: scaled RL optimizes routing for target domains (code, math, instruct) at the expense of general-domain performance. The tax arises because the sigmoid gating mechanism couples all domains through shared gate weights: concentrating code routing necessarily shifts decision boundaries for all other domains. This has direct implications for MoE training: routing regularization during RL (e.g., penalizing routing entropy deviation from the pre-scaling configuration on non-target domains) could reduce the routing tax without sacrificing target-domain gains.

8.3 The Reward-Selective Specialization Hypothesis, Revisited

The causal results force us to refine the reward-selective hypothesis. The original formulation (Section 4.4) proposed that reward signal verifiability determines routing change direction. The gate-swap results show that this structural pattern is real but not functionally important in isolation. The DeepSeek cross-architecture analysis (Section 7) adds a crucial constraint: DeepSeek V3 undergoes SFT+RL with similar reward structures yet shows *no domain-selective specialization*, with uniformly small ΔH across all domains. A doubly-revised formulation:

Revised hypothesis. Verifiable rewards (code) produce consistent gradient signals that specialize both expert weights and routing jointly, while subjective rewards (reasoning, instruct) produce noisier gradients that disperse both. However, the *expression* of this domain-selective co-adaptation depends on routing architecture: direct top-k routing (MiniMax) permits large entropy shifts because each expert competes independently, whereas grouped routing (DeepSeek) constrains the search space: 73.5% of routing divergence is confined within preserved groups—dampening domain-selective effects even when the same reward dynamics apply.

This revised hypothesis makes three predictions: (1) domain-selective entropy shifts should appear in MoE models with direct top-k routing, (2) grouped or hierarchical routing should attenuate these shifts, and (3) regardless of architecture, deep layers should show greater routing divergence than shallow layers. Our results are consistent with all three.

8.4 Architectural Stabilizers: Why Grouped Routing Resists Co-Adaptation

The DeepSeek comparison reveals that routing architecture has a first-order effect on how post-training reshapes routing. We identify three mechanisms through which grouped routing acts as a stabilizer:

1. **Constrained search space.** By first selecting 4 of 8 groups and then 8 of 128 experts, grouped routing reduces the effective combinatorial space from $\binom{256}{8} \approx 4.4 \times 10^{13}$ to $\binom{8}{4} \times \binom{128}{8} \approx 5.5 \times 10^{14}$. The group selection stage acts as a bottleneck that limits how far routing can deviate.
2. **Near-saturated group entropy.** Group-level entropy in DeepSeek is 2.993 bits ($\max = 3.0$), meaning group selection is almost uniform and essentially "locked" by the correction bias. Post-training cannot easily break this near-uniform distribution.
3. **Gradient locality.** Because experts within a group share a group-selection gate score, gradient updates to individual expert affinities are modulated by the group's collective standing. This creates a form of implicit regularization: an expert can only gain routing share by improving its group's top-2 score, coupling individual expert optimization to group-level stability.

These mechanisms suggest a design principle: *routing architectures with hierarchical constraints, combined with non-differentiable load-balancing algorithms, are more robust to post-training routing drift*. This is directly relevant for practitioners choosing between flat and grouped MoE designs: grouped routing combined with auxiliary-loss-free balancing sacrifices some routing flexibility for stability under post-training.

8.5 Implications for MoE Training and Deployment

The causal findings sharpen the practical implications:

Routing-aware RL. Rather than encouraging routing concentration (which the original hypothesis suggested), practitioners should *regularize routing stability* during scaled RL. Since routing changes are not the capability mechanism but do impose a general-domain tax, constraining the gate to stay close to its earlier configuration (e.g., via KL penalty on routing distributions) could preserve general-domain quality while allowing expert weights to specialize freely.

Domain-specific expert pruning. The code specialization ($\Delta H = -0.23$ bits) suggests that code inference could be served with fewer active experts, reducing latency and memory. The gate-swap result strengthens this: code perplexity is robust to substantial routing perturbation ($+0.5\%$ under 43% routing change at deep layers), indicating considerable redundancy in routing precision for code.

Expert weight transfer. Since capability resides in expert weights rather than routing, techniques like expert merging or distillation should focus on preserving expert computation quality rather than routing fidelity.

Domain-stratified RL schedules. Rather than applying uniform RLHF, MoE models may benefit from domain-stratified training schedules that apply different RL intensities per domain, combined with routing regularization to limit the general-domain tax.

8.6 Relation to Prior Work

ST-MoE [Zoph et al., 2022] studied expert routing stability during pre-training but not post-training. GShard [Lepikhin et al., 2021] demonstrated large-scale MoE training with position-based routing analysis. Clark et al. [Clark et al., 2022] established unified scaling laws for routed language models. DeepSeek-MoE [Dai et al., 2024] introduced fine-grained expert segmentation with grouped routing; DeepSeek-V2 [DeepSeek-AI, 2024a] scaled this to production, and DeepSeek-V3 [DeepSeek-AI, 2024b] extended to 671B parameters with auxiliary-loss-free balancing via correction biases. Our cross-architecture comparison in Section 7 directly builds on this lineage, providing the first

empirical evidence that grouped routing acts as a stabilizer against post-training routing drift. Mixtral [Jiang et al., 2024] showed that experts specialize by topic but did not compare pre- and post-RL routing. Gao et al. [Gao et al., 2024] found that deeper layers require more LoRA experts during fine-tuning, consistent with our depth-dependent divergence finding in both MiniMax and DeepSeek.

Most recently, Zhang et al. [Zhang et al., 2025] address RL-induced routing instability *during training*, proposing router-aware importance sampling to reduce gradient variance in off-policy MoE optimization. Their work confirms that RL interacts non-trivially with expert routing but focuses on stabilizing training dynamics rather than characterizing the resulting routing changes. No prior work has analyzed post-training routing patterns across domains or tested their causal role, a gap this work fills.

Our gate-swap methodology is related to causal tracing in dense models [Meng et al., 2022, Vig et al., 2020], where activations at specific layers are patched to localize factual knowledge. We extend this approach to MoE architectures by intervening on routing parameters rather than activations, enabling a clean decomposition of routing versus computation contributions. To our knowledge, no prior work has performed causal routing interventions at this scale in a production MoE, nor demonstrated the "routing tax" phenomenon we identify.

9 Limitations and Future Work

Two model families. Our analysis covers MiniMax (M2.1 vs. M2.5) and DeepSeek (V3-Base vs. V3). While the cross-architecture comparison strengthens generality, both models use 256 experts with top-8 routing and sigmoid gating. Replication on architecturally distinct MoE designs (e.g., Switch Transformers with top-1, Mixtral with top-2/8 softmax gating, or models with different expert counts) would further establish the scope of our findings.

Confounded training stages. For MiniMax, both M2.1 and M2.5 underwent SFT + RL, with M2.5 extending to more environments and adding process rewards. For DeepSeek, V3-Base is pre-trained and V3 undergoes SFT+RL, making the training gap larger but less controlled. The routing changes we observe in each pair reflect *joint* effects that we cannot decompose into individual training stages without intermediate checkpoints.

Gate-only intervention and layer-selective extensions. Our gate-swap tests routing in isolation across all 62 layers simultaneously. A natural extension is *layer-selective* gate swapping: partitioning layers into three depth zones—shallow (L0–20, Jaccard $0.79 \to 0.71$), middle (L21–40, Jaccard $0.71 \to 0.60$), and deep (L41–61, Jaccard $0.60 \to 0.42$)—and swapping gates within each zone independently. The monotonic Jaccard gradient (Section 4.2) predicts that deep-only swaps should reproduce the majority of the general-domain improvement observed in the full swap (-7.8%), while shallow-only swaps should have minimal effect. Confirming this prediction would establish that the routing tax is concentrated in the high-divergence layers. Additionally, a *gate interpolation sweep*—α-blending gate weights as $\mathbf{W}_g^{(\alpha)} = (1-\alpha)\mathbf{W}_g^{\text{M2.5}} + \alpha\,\mathbf{W}_g^{\text{M2.1}}$ for $\alpha \in \{0, 0.25, 0.5, 0.75, 1.0\}$—would map the Pareto frontier of the routing tax, revealing whether the transition from specialized to general routing is gradual or threshold-like. Both extensions require only modifying the per-layer swap condition in the existing infrastructure, adding approximately 6–8 forward passes ($\sim$5–7 hours on a single B200 GPU).

No task-level evaluation. We measure perplexity effects but do not evaluate downstream task performance. A concrete follow-up: evaluate the four gate-swap conditions (plus the deep-only swap) on MMLU (5-shot, log-likelihood), GSM8K-CoT (8-shot, generation), and HumanEval (pass@1) using `lm-evaluation-harness`. This requires loading the full 229B model on multi-GPU (2$\times$B200, 384 GB total), then replacing gate weight tensors in-memory for each swap condition—a lightweight operation since gates comprise only 0.02% of parameters. Key predictions: (a) MMLU scores should be dominated by the body model, matching the PPL finding; (b) HumanEval should be robust to gate swap, matching the $<1\%$ code PPL effect; (c) if the routing tax manifests as degraded MMLU or GSM8K accuracy under M2.5's gates (relative to M2.1's gates on M2.5's body), this would strengthen the practical case for routing regularization during RL.

Corpus imbalance. Domain token counts range from 9,273 (reasoning) to 434,353 (general), a 47× ratio. Subsampling code to 9,273 tokens retains 100% of the effect (Section 6.1), and sample-level bootstrap accounts for within-sample dependence (Section 6.2), but entropy estimates for small domains are inherently noisier.

Gate weight analysis. We analyze routing *decisions* but do not directly compare gate weight matrices ($\mathbf{W}_g$) between models. Computing per-layer cosine similarity between M2.5 and M2.1 gate weights would substantiate (or refute) the "decision boundary retuning" interpretation.

Shared experts. MiniMax's architecture includes shared experts alongside routed experts. Our analysis focuses exclusively on routed experts.

Linear decomposition assumption. The routing-weight decomposition in Table 7 assumes approximate linearity: that the effects of routing and weight changes are additive. In practice, there may be nonlinear interactions between routing and computation that the 2×2 swap design cannot fully capture.

10 Conclusion

We present the first comprehensive analysis of how post-training reshapes expert routing in production-scale Mixture-of-Experts language models, combining layer-by-layer observational measurements with a causal gate-swap intervention and cross-architecture validation. Our analysis spans two model families (MiniMax-M2, 229B; DeepSeek V3, 671B), >1.7M tokens, and 120 MoE layers.

Observationally, post-training produces robust routing changes in both architectures, but the *character* of these changes depends on routing design. MiniMax's direct top-8/256 routing undergoes *domain-selective* specialization: code concentrates ($\Delta H = -0.23$ bits) while reasoning/general/instruct disperse ($+0.11$ to $+0.17$ bits), with Jaccard declining monotonically from 0.68 to 0.42. DeepSeek's grouped routing (top-4 of 8 groups, then top-8) is substantially more stable: mean Jaccard $= 0.67$, top-1 agreement $= 0.85$, and entropy changes are uniformly small ($|\Delta H| < 0.12$ bits) with no domain-selective pattern. The two-stage grouped routing acts as an architectural stabilizer: group-level Jaccard remains 0.82 even as expert-level agreement declines, with 73.5% of divergence confined within preserved groups.

Causally, the routing changes are *structurally real but functionally secondary in next-token prediction*. Gate-swapping between MiniMax models—transplanting only the 0.02% of parameters that control routing—produces <1% code perplexity change despite altering 43% of routing decisions at deep layers. Remarkably, reverting M2.5 to M2.1's routing *improves* overall perplexity by 4.3%, driven by a 7.8% gain on general text, revealing that routing specialization from scaled RL imposes a *routing tax* on non-target domains.

Together, these results establish that scaled RL creates tightly co-adapted routing-weight configurations in MoE models, where routing tracks expert specialization but does not independently drive it. The cross-architecture comparison reveals that this co-adaptation is modulated by routing geometry: grouped routing constrains the search space and limits domain-selective drift, while direct routing permits broader reorganization. Both architectures share a universal pattern—monotonically increasing routing divergence with depth—consistent with shallow layers encoding universal features and deep layers encoding task-specific representations.

This co-adaptation picture has practical implications: routing regularization during RL can reduce the general-domain tax; expert pruning for code can tolerate imprecise routing; expert weight transfer matters more than routing fidelity; and grouped routing designs may be preferred when post-training stability is a priority. More broadly, the gate-swap methodology demonstrates that causal interventions on routing parameters—which are uniquely separable in MoE architectures—can decompose the contributions of different parameter groups to model behavior, opening a new avenue for mechanistic interpretability of sparse models. Layer-selective gate swaps and downstream task evaluation (Section 9) will further test whether the co-adaptation is depth-dependent and whether the routing tax extends beyond perplexity to task-level performance.

References

Yuntao Bai, Saurav Kadavath, Sandipan Kundu, et al. Constitutional AI: Harmlessness from AI feedback. *arXiv preprint arXiv:2212.08073*, 2022.

Damai Dai, Chengqi Deng, Chenggang Zhao, et al. DeepSeekMoE: Towards ultimate expert specialization in mixture-of-experts language models. *arXiv preprint arXiv:2401.06066*, 2024.

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.

Albert Q. Jiang, Alexandre Sablayrolles, Antoine Roux, et al. Mixtral of Experts. *arXiv preprint arXiv:2401.04088*, 2024.

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. Locating and editing factual associations in GPT. In *Advances in Neural Information Processing Systems*, volume 35, 2022.

MiniMax. MiniMax-01: Scaling foundation models with lightning attention. *arXiv preprint arXiv:2501.08313*, 2025.

Long Ouyang, Jeffrey Wu, Xu Jiang, et al. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems*, volume 35, 2022.

Noam Shazeer. GLU variants improve Transformer. *arXiv preprint arXiv:2002.05202*, 2020.

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, et al. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. In *International Conference on Learning Representations*, 2017.

Jianlin Su, Murtadha Ahmed, Yu Lu, et al. RoFormer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063, 2024.

Jesse Vig, Sebastian Gehrmann, Yonatan Belinkov, et al. Investigating gender bias in language models using causal mediation analysis. In *Advances in Neural Information Processing Systems*, volume 33, 2020.

Di Zhang, Xun Wu, Shaohan Huang, Yaru Hao, Li Dong, Zewen Chi, Zhifang Sui, Furu Wei. Towards stable and effective reinforcement learning for mixture-of-experts. *arXiv preprint arXiv:2510.23027*, 2025.

Barret Zoph, Irwan Bello, Sameer Kumar, et al. ST-MoE: Designing stable and transferable sparse expert models. *arXiv preprint arXiv:2202.08906*, 2022.

Dmitry Lepikhin, HyoukJoong Lee, Yuanzhong Xu, et al. GShard: Scaling giant models with conditional computation and automatic sharding. In *International Conference on Learning Representations*, 2021.

Aidan Clark, Diego de Las Casas, Aurelia Guy, et al. Unified scaling laws for routed language models. In *International Conference on Machine Learning*, 2022.

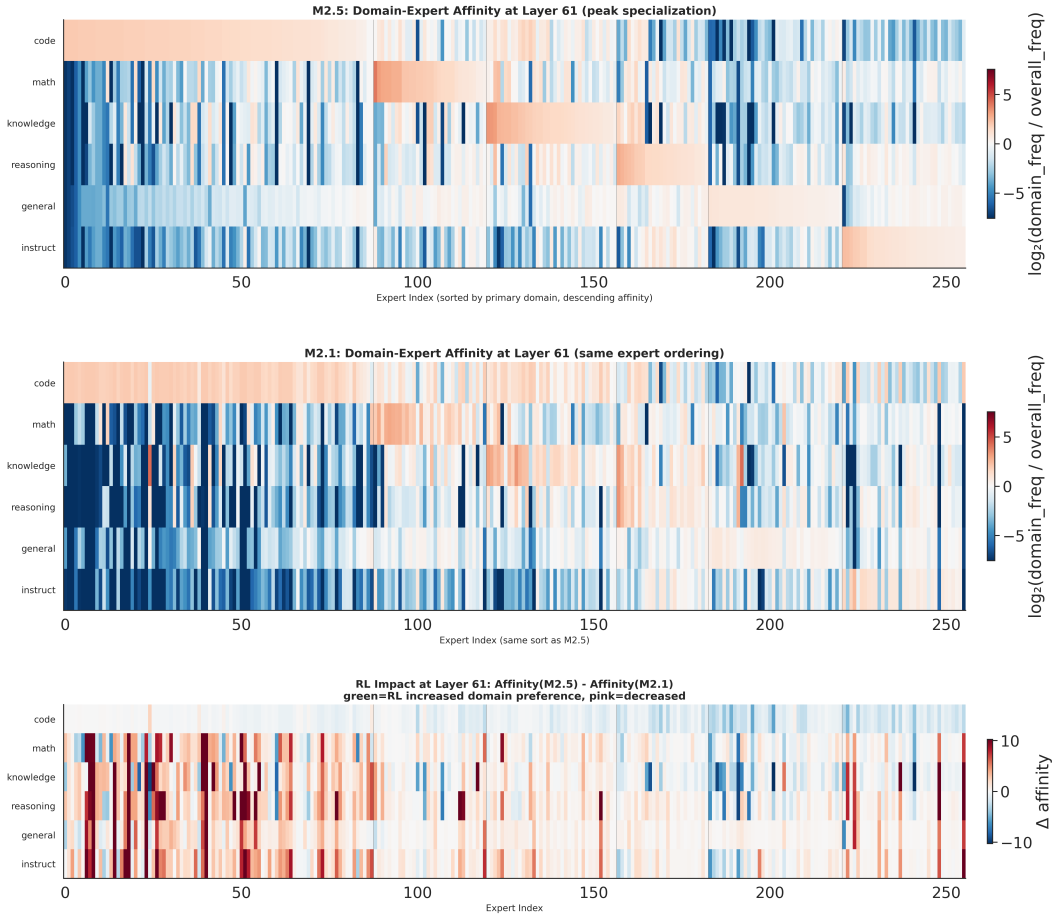
DeepSeek-AI. DeepSeek-V2: A strong, economical, and efficient mixture-of-experts language model. *arXiv preprint arXiv:2405.04434*, 2024.

DeepSeek-AI. DeepSeek-V3 technical report. *arXiv preprint arXiv:2412.19437*, 2024.

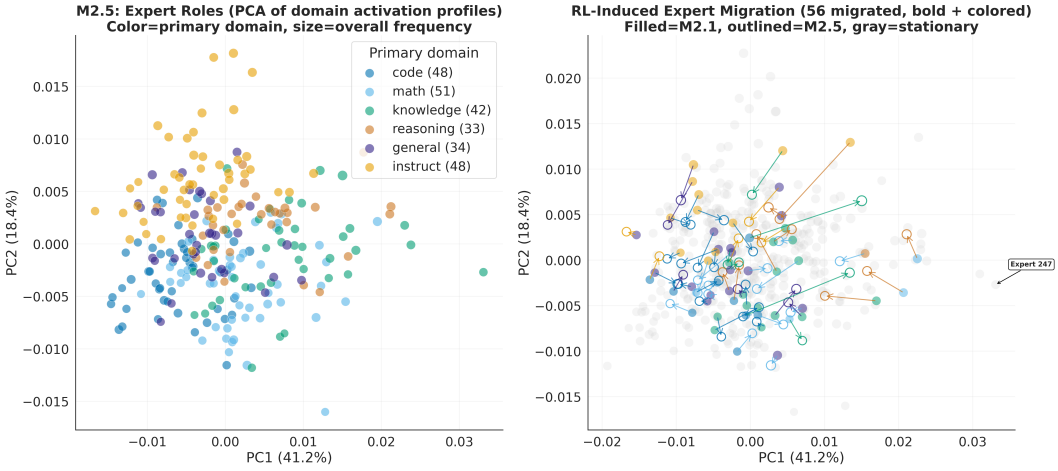
Chongyang Gao, Kezhen Chen, Jinmeng Rao, et al. Higher layers need more LoRA experts. *arXiv preprint arXiv:2402.14562*, 2024.

A Figure Index

Fig.	Description	File
1	Expert frequency heatmaps (MiniMax)	`full_layer_freq_heatmap_*.png`
2	Routing divergence & agreement (MiniMax)	`full_layer_routing_*.png`
3	Domain-stratified entropy (combined)	`domain_stratified_rl_impact_combined.png`
4	Domain-stratified Jaccard (CIs)	`domain_stratified_jaccard_CI.png`
5	Expert affinity & clustering	`domain_expert_affinity.png`
6	Gate-swap causal intervention	`routing_swap_experiment.png`
7	Subsampled code validation	`subsampled_code_validation.png`
8	Sample-level bootstrap	`sample_level_bootstrap.png`
9	Token-level routing analysis	`token_level_routing.png`
10	Cross-architecture routing agreement	`cross_architecture_routing_agreement.png`
11	DeepSeek group vs expert agreement	`deepseek_group_vs_expert_agreement.png`
12	Cross-architecture domain entropy	`cross_architecture_domain_entropy.png`
13	Cross-architecture summary	`cross_architecture_summary.png`



(a) Domain-expert affinity (M2.5, layer 61)



(b) Expert role clustering (PCA projection)

Figure 5: **Expert specialization and reorganization.** **(a)** Domain-expert affinity matrix at the peak specialization layer (layer 61, $H = 7.18$ bits). Experts sorted by hierarchical clustering; domain boundaries marked. Math shows the strongest specialization (top affinity $+0.90$), general the weakest ($+0.26$). **(b)** PCA projection of expert domain-activation profiles. Distinct domain-specific clusters are visible, with arrows indicating RL-induced expert migration. 56/256 experts (21.9%) changed primary domain.

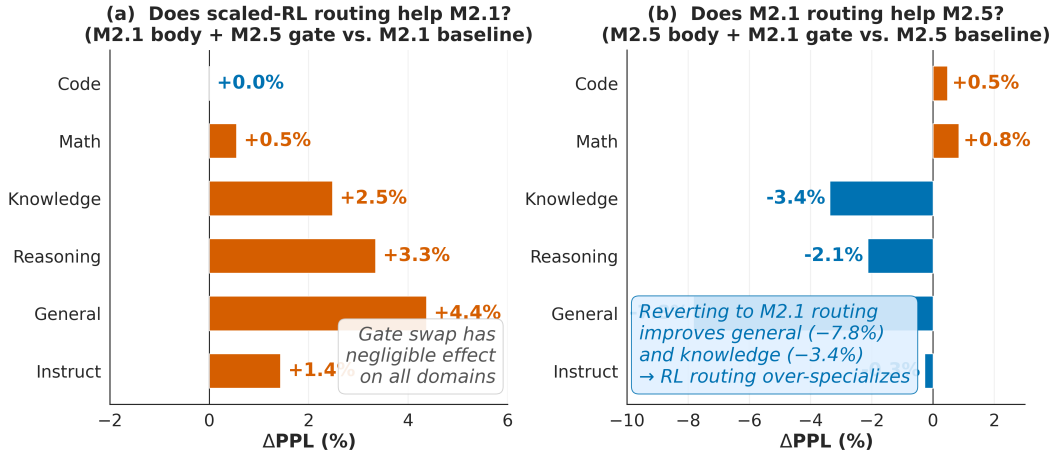


Figure 6: **Gate-swap causal intervention results.** **(a)** ΔPPL (%) when transplanting M2.5's gates into M2.1's body: all changes <5%, with code essentially unchanged (+0.0%). **(b)** ΔPPL (%) when reverting M2.5 to M2.1's routing: general improves by 7.8% and knowledge by 3.4%, indicating the additional RL routing over-specializes at the expense of non-target domains. Code is barely affected (+0.5%).

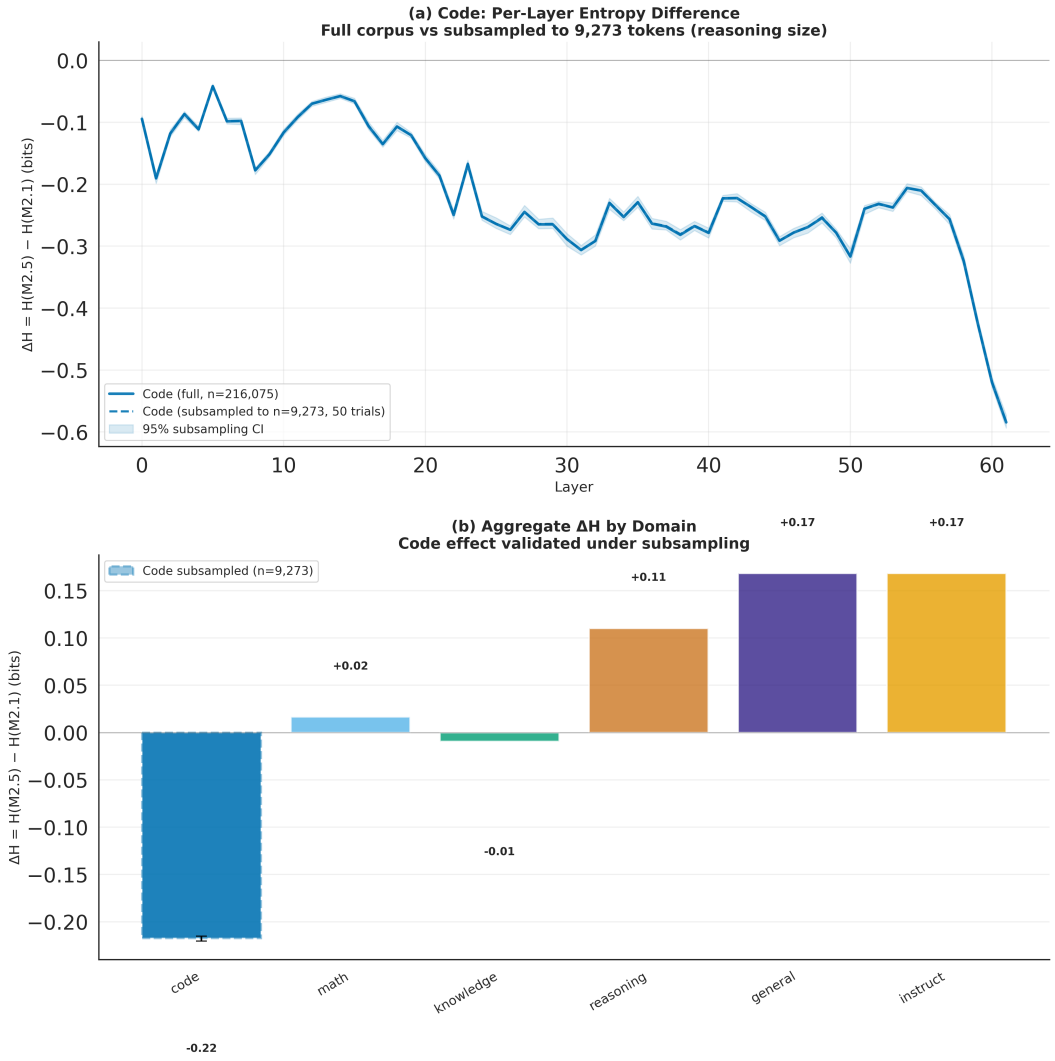


Figure 7: **Subsampling validation.** Left: per-layer ΔH for code using the full corpus (solid) vs. subsampled to $n = 9{,}273$ tokens (dashed, with 95% CI from 50 subsamples). The effect persists across all layers and all subsamples, retaining 100% of the full-corpus magnitude. Right: aggregate ΔH by domain, with the subsampled code estimate overlaid.

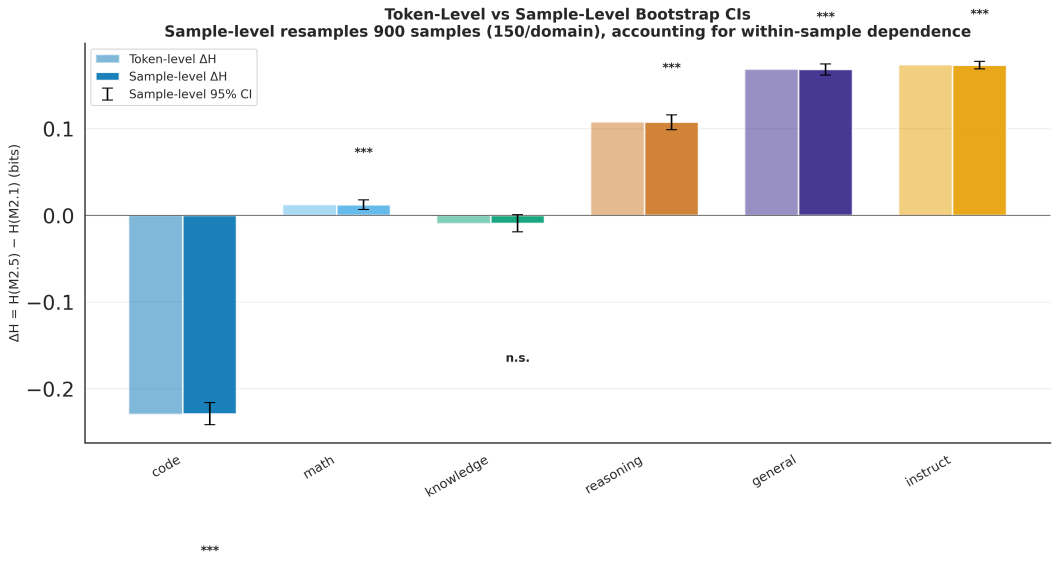


Figure 8: **Token-level vs. sample-level bootstrap.** All four significant effects (code, reasoning, general, instruct) survive the conservative sample-level bootstrap with $n_{\text{samples}} = 150$ per domain. Math and knowledge remain non-significant under both procedures.

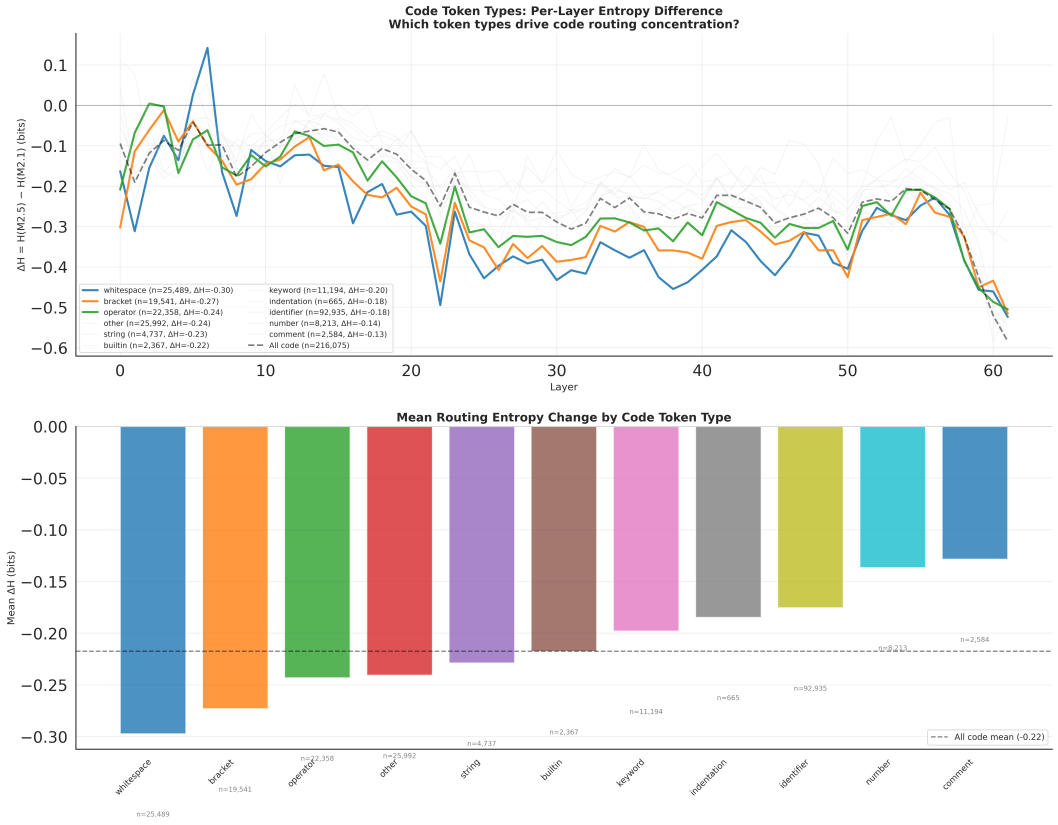


Figure 9: **Code concentration by token type.** Top: per-layer ΔH curves for each code token category. Bottom: aggregate mean ΔH by category. The concentration effect is not uniform across token types: Python keywords and operators show the strongest concentration, while identifiers and whitespace show weaker effects. This suggests post-training sharpens routing for *structurally predictable* token types within code.

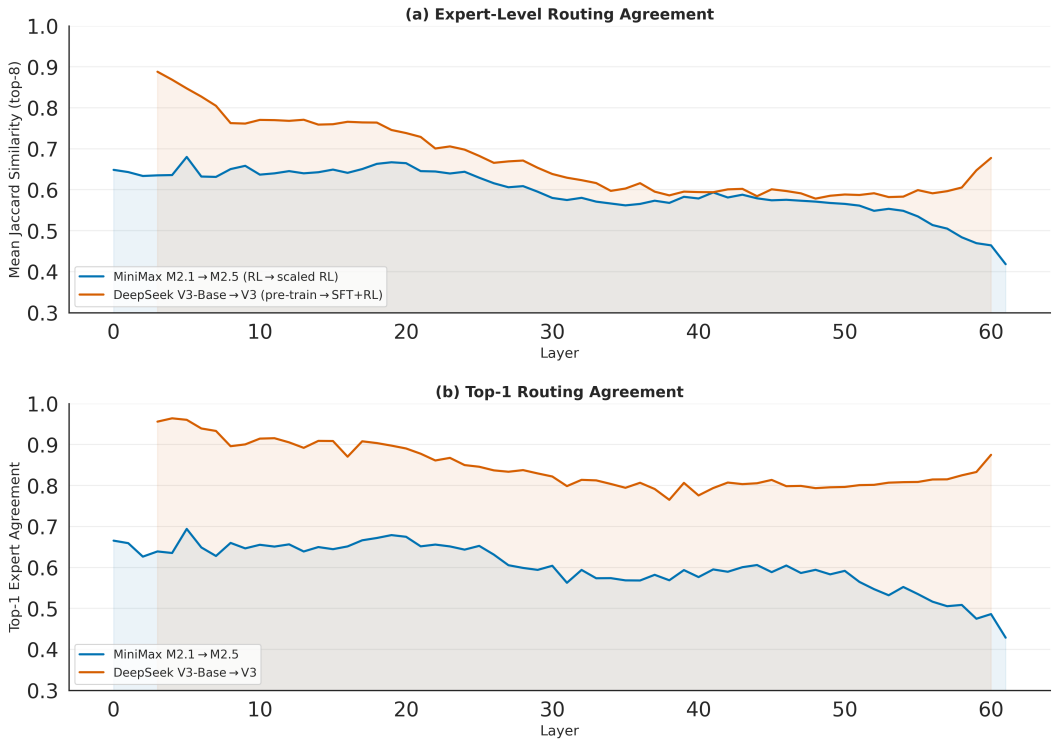


Figure 10: **Cross-architecture routing agreement.** **(a)** Top-8 Jaccard similarity. Both models show monotonically declining agreement with depth, but DeepSeek maintains substantially higher agreement throughout (mean $J = 0.67$ vs. MiniMax's 0.60). **(b)** Top-1 expert agreement. The gap is even larger: DeepSeek preserves 85% top-1 agreement vs. MiniMax's 60%, indicating that the highest-confidence routing decision is particularly stable under grouped routing.

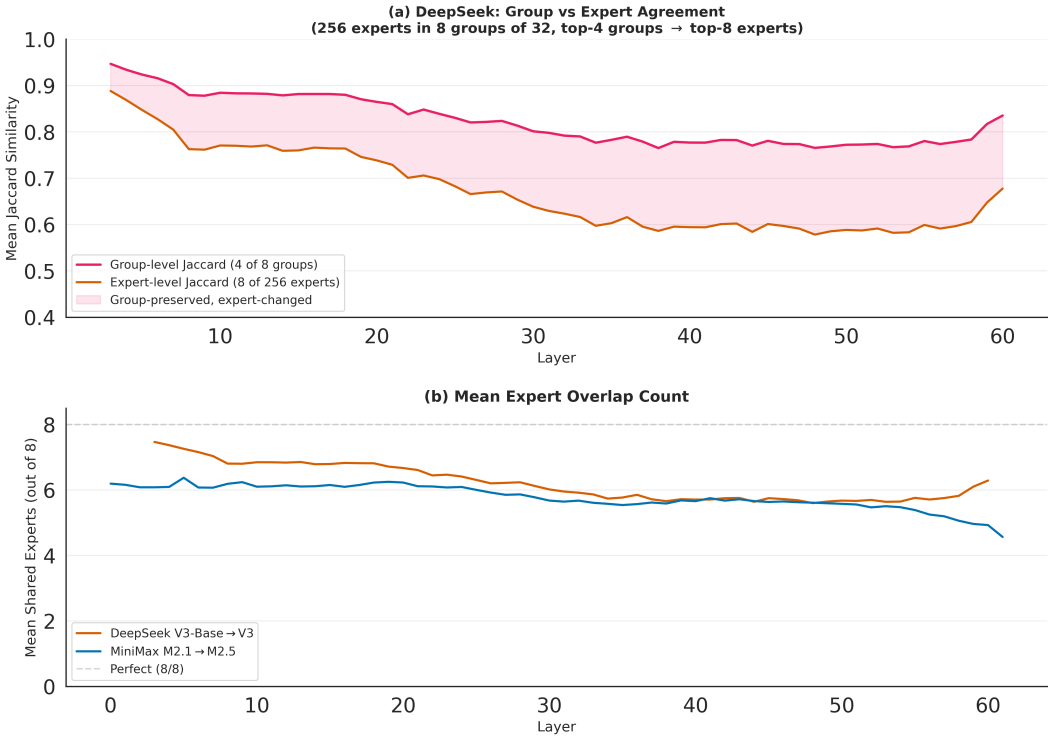


Figure 11: **Group-level vs. expert-level routing agreement in DeepSeek V3.** **(a)** Group-level Jaccard (which 4 of 8 groups are selected) remains high (mean 0.82) even as expert-level Jaccard declines (mean 0.67). The shaded gap represents tokens where group selection is preserved but expert assignment within groups has changed. **(b)** Mean expert overlap count (out of 8). DeepSeek preserves ∼6.2 of 8 experts across models, vs. MiniMax's ∼5.8.

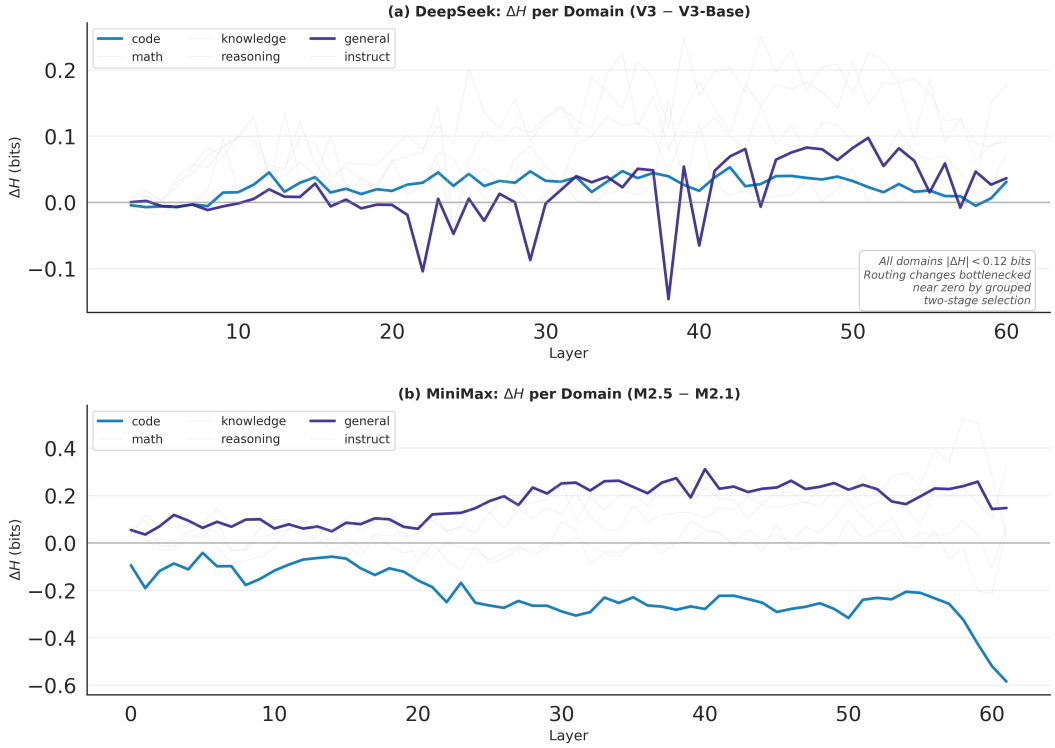


Figure 12: **Domain-stratified entropy change.** **(a)** DeepSeek: all domains show uniformly small ΔH (range $+0.02$ to $+0.12$ bits), with no clear domain-selective pattern. **(b)** MiniMax: strong asymmetry—code concentrates (-0.22 bits), while general and instruct disperse ($+0.17$ bits).

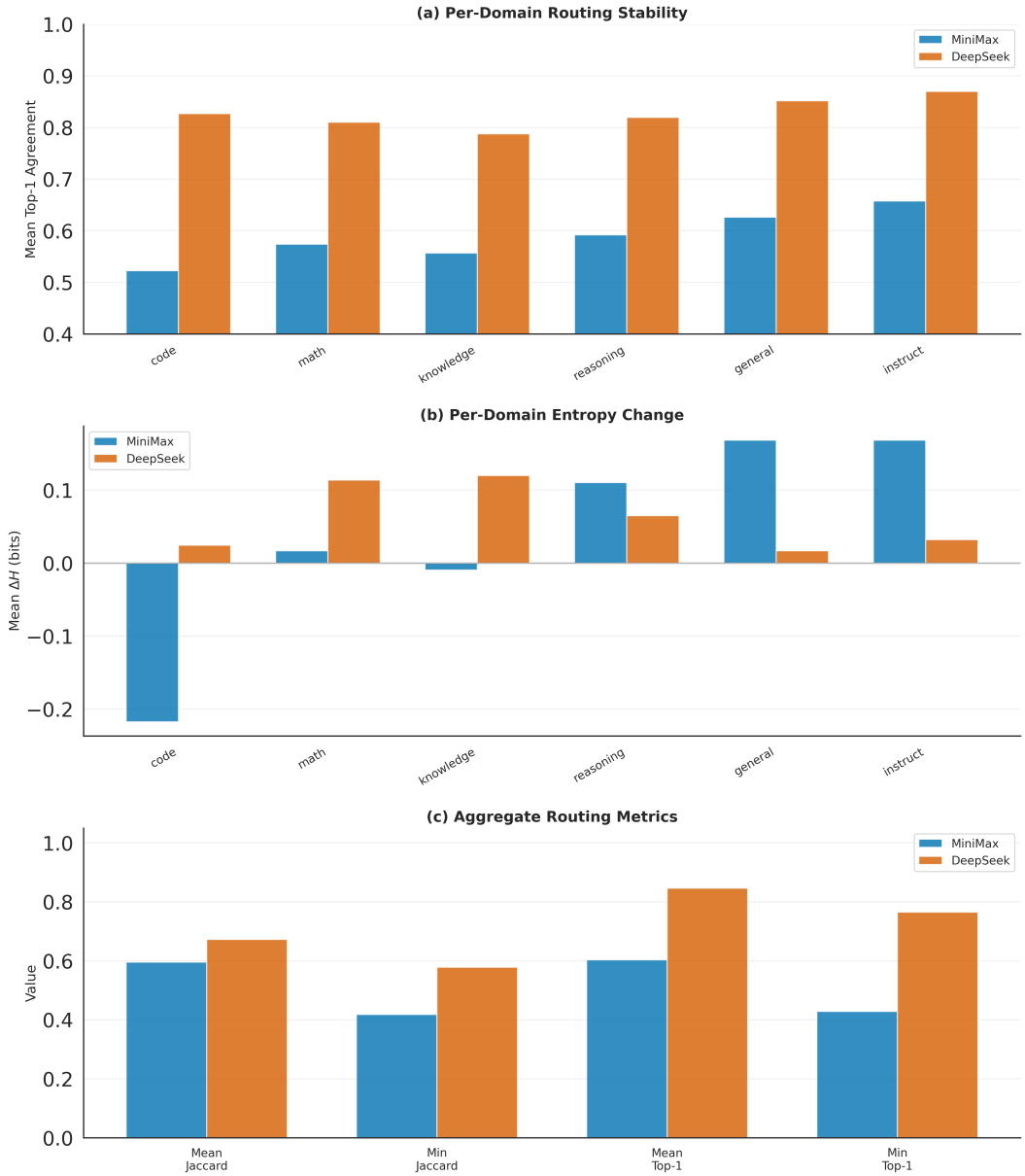


Figure 13: **Cross-architecture summary.** **(a)** Per-domain top-1 routing stability. DeepSeek exceeds MiniMax by 21–30 percentage points across all domains. **(b)** Per-domain entropy change. MiniMax shows dramatic asymmetry (code $= -0.22$ bits); DeepSeek is uniformly small. **(c)** Aggregate metrics.