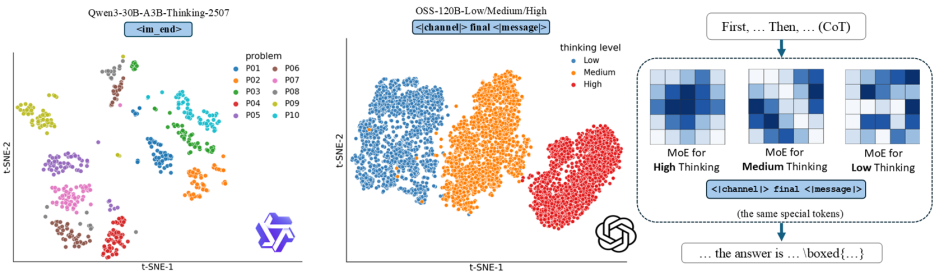
DOES THE SAME TOKEN MEAN THE SAME STATE? MOE ROUTING AS SIGNAL FOR REASONING CONTROL

Anonymous authors
Paper under double-blind review

ABSTRACT

Sparse Mixture-of-Experts (MoE) inference produces expert-routing decisions as a byproduct of generation. We ask whether these already-computed decisions can be reused to select among completed candidate solutions. Holding the emitted token id fixed at repeated anchors, we find that the same token is still produced by different experts across contexts, and that short routing traces at aligned answer-opening positions carry a consensus signal: near *boundary* anchors (the final-response transition) and *delimiter* anchors (which open the answer, e.g. `\boxed{` or code fences), routing neighborhoods align with final-answer basins, most strongly at the answer opening. We operationalize this as **RAD** (Routing Agreement Decoding), an answer-string-free multi-rollout selector: it locates a fixed anchor, represents each rollout by its anchor-window MoE routing states, and returns the densest Weighted-Jaccard K-NN route-basin center, without parsing, normalizing, executing, or voting over answer strings. Across 10 sparse-MoE configurations (gpt-oss, Qwen3-MoE) and 6 datasets spanning math, GPQA, and code, RAD is on par with Majority where string voting is well-posed, with small positive paired deltas (RAD 73.9 / RAD+DC 74.2 vs. Majority 73.6). Like majority voting, RAD is not a verifier: a dense *wrong* basin can still win. Its value is the interface: the same selector gives direct pass@1 on code, where exact-string voting is ill-defined, and the same routing-density principle, re-anchored to the agentic boundary, improves best-of-16 patch selection on SWE-bench Verified over random, where patches have no answer string to vote on.



(a) (b) (c)

Figure 1: **Same token, different generating experts.** At a fixed emitted token id, the generating experts still cluster by problem identity (Qwen EOS) and reasoning-effort mode (OSS boundary). *A,B*: t-SNE of pairwise WJ; *C*: schematic.

1 INTRODUCTION

A decoded token is a coarse summary of the computation behind it; even output-side signals such as entropy are low-dimensional projections of the internal state (Chen et al., 2025). Sparse Mixture-of-Experts (MoE) models expose one slice of that state: a router selects a few experts per token from the current hidden state (Shazeer et al., 2017; Fedus et al., 2022), so the same token id can be produced by different expert routes. Does the same token mean the same state, and if not, can the residual routing serve test-time control?

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.

The control problem is selection in multi-rollout reasoning: a system samples many traces and must return one (Wei et al., 2022; Wang et al., 2023). Voting over extracted answer strings works for short canonical answers but not for programs and patches, where exact-string votes collapse to singletons. Internal signals can arbitrate instead: NAD ranks candidates by activated-neuron agreement and NEX scores neuron states (Chen et al., 2025; 2026). But dense internals are costly to read: NAD hooks the feed-forward layers and keeps the top-500 neurons per token, whose storage "requires substantial disk and memory" (Chen et al., 2025), and the sparse-autoencoder route (Bricken et al., 2023; Cunningham et al., 2024) trains a separate encoder on collected activations (its own GPU budget) and runs it as an extra pass at inference. Neither is a free, real-time byproduct of decoding.

MoE routing is. The top-k expert ids and weights are computed to dispatch every token ($\sim$7% capture overhead; App. A.5) and are discarded once the token is dispatched. Prior work treats routing as a load-balancing or interpretability object (Xue et al., 2024; Ying et al., 2025), not as a record of what a rollout has committed to. We ask a narrower question: is this native sparse record, at most ten experts per layer, already a sufficient agreement signal, and *where* should it be read?

It is, and position matters (§3). Aligning rollouts at a repeated token id (Figure 1), the experts producing the *same* token still separate problem identity and reasoning-effort regimes, with under 5% nearest-neighbor leakage: token identity is not a sufficient statistic for router state. Within a problem, routing neighborhoods at the answer-opening delimiter align with *answer basins* (rollouts sharing a normalized answer) far above random windows, the marker alone is informative, the 16-token window is strongest, and basin purity tracks selector accuracy.

We operationalize this as **Routing Agreement Decoding (RAD)**: for one problem's completed rollout pool, RAD locates a fixed anchor by token id, pools each rollout's routing over a short anchor window, and returns the rollout with the densest Weighted-Jaccard K-NN neighborhood (Figure 3, §4). Selection adds no forward pass and never parses, normalizes, compares, executes, or votes over answer strings; the strict marker-only readout recovers most of the lift. Like Majority, RAD estimates shared commitment rather than truth: Majority counts normalized answers, RAD picks the locally densest route basin, and a dense wrong basin can win.

We evaluate RAD on 10 sparse-MoE configurations and 6 datasets (gpt-oss, Qwen3-MoE; math, GPQA, code). Where string voting is well-posed it is on par with textual Majority (73.9 vs. 73.6; non-significant paired deltas); on code, where exact-string voting is ill-defined, it returns direct pass@1 above the random floor and, on the full problem set, is level with execution-based functional majority voting (Launer et al., 2026) without executing anything; and on agentic SWE-bench Verified, re-anchored at the thinking→action boundary, it improves best-of-16 patch selection by $+2.2$ to $+2.9$ pp end-to-end over random.

Contributions. We (i) show that the same token id is generated by different expert routes and that aligned routing neighborhoods track final-answer basins, strongest at answer-opening readouts; (ii) propose **RAD**, a selector that reuses the routing MoE inference already computes and needs no answer-string grouping; and (iii) validate it on canonical answers, code, and agentic patch trajectories, and show that it follows the densest basin, right or wrong.

2 Related Work

Multi-rollout reasoning and test-time control. Chain-of-thought prompting and self-consistency improve reasoning by sampling multiple paths and aggregating final answers (Wei et al., 2022; Wang et al., 2023). Recent controllers reduce this cost using early stopping, rationale or prefix agreement, confidence scores, or thought-level pruning (Li et al., 2024; Wan et al., 2025; Hong et al., 2025; Jindal et al., 2026; Fu et al., 2026). These methods arbitrate through visible text, answer strings, or scalar scores. RAD instead selects from a completed rollout pool using MoE routing agreement and never compares final answer strings, so it extends to agentic best-of-N patch selection, where candidates expose no answer string to vote on.

Internal signals for selection. Mechanism-aware evaluation relates internal activations, neurons, or utilization patterns to model behavior (Cao et al., 2025; Zhang et al., 2025; Chen et al., 2025; 2026). NAD (Chen et al., 2025), the closest predecessor, builds token-conditioned activated-neuron sets over prefixes or chunks, and NEX (Chen et al., 2026) scores neuron states; RAD claims no new

agreement objective, only a different readout: the top-k expert assignments the MoE forward pass already emits, read at token-id-matched anchors, so the payload is $O(W \cdot L \cdot k_{\text{route}})$, the shallowest tenth of layers is within 1.7 pp of the full stack, and binary expert membership matches the weighted kernel (Apps. A.14, A.13). We retained no matched hidden-state features for these pools, so we claim operational sufficiency of routing, not uniqueness (§6), and treat routing density as an answer-basin signal rather than a correctness oracle.

MoE routing. Sparse MoE models route each token through a small subset of experts (Shazeer et al., 2017; Lepikhin et al., 2021; Fedus et al., 2022; Jiang et al., 2024; Dai et al., 2024). Prior work shows that routing is shaped by token identity as well as hidden-state geometry (Xue et al., 2024; Hayashi et al., 2026; Wang et al., 2026; Ying et al., 2025). RAD turns this confound into an alignment device: repeated anchor token ids hold the surface token fixed, while residual routing geometry is used to identify redundancy at trajectory anchors and dense answer basins around boundary and delimiter anchors.

3 Analysis of Anchor-Conditioned Routing

We use anchor-conditioned routing to ask whether the routing record MoE inference already produces (which experts drive each token) contains rollout-level structure beyond token identity. Exp I holds the emitted token id fixed and tests whether the experts that generate it still separate upstream context. Exp II tests whether routing neighborhoods align with final-answer basins near final-answer production. We do not interpret individual experts or claim that routing predicts correctness.

3.1 Answer-string-free protocol, route representations, and diagnostic

Setup and access policy. For each problem p, we sample $N{=}64$ rollouts and record both token ids and sparse MoE route vectors $R_i(t)$ (selected expert ids and post-softmax weights across layers). The index convention is token-synchronous but prefix-causal: $R_i(t)$ is recorded in the forward pass that predicts $y_{i,t}$ from $y_{i,<t}$, so it is aligned with token t, not conditioned on $y_{i,t}$ itself. Anchors are matched on the recorded token-id stream, not re-tokenized text. We use three anchor types: *trajectory anchors*, recurring intermediate markers such as `so`, `now`, or paragraph breaks; *boundary anchors*, final-response transition markers such as `<|channel|>final<|message|>` or `</think>`; and *delimiter anchors*, fixed answer-opening delimiters such as `\boxed{` or code fences. Readout windows (`type@W`, `type@marker`) are defined in §4, and the selector's access boundary in Appendix A.1.

Route representation and Weighted Jaccard. At token t and MoE layer ℓ, the router selects k_{route} experts with normalized weights. Across our model pool $k_{\text{route}} \in \{4, 8, 10\}$; for Qwen3-Next the constant shared expert is excluded from the WJ histogram (Appendix A.5). Concatenating the per-layer sparse vectors gives $R_i(t) \in \mathbb{R}^{L \cdot E}$, and averaging over an anchor window yields a per-rollout route vector. We compare two route vectors u, v by Weighted Jaccard:

$$\mathrm{WJ}(u, v) \;=\; \frac{\sum_e \min(u_e, v_e)}{\sum_e \max(u_e, v_e)}. \tag{1}$$

WJ preserves routed expert mass rather than binarizing fired experts; the binary comparison appears in §5.3.

Diagnostic metric. For analysis only, we measure chance-corrected K-NN answer-basin purity:

$$A_{K_{\text{nn}},r} \;=\; \frac{p_{K_{\text{nn}},r} - b}{1 - b}, \quad p_{K_{\text{nn}},r} = \frac{1}{N} \sum_i \frac{1}{K_{\text{nn}}} \sum_{j \in \mathrm{NN}_{K_{\text{nn}}}(i,r)} \mathbf{1}[a_i = a_j], \quad b = \frac{\sum_a n_a(n_a - 1)}{N(N-1)}. \tag{2}$$

Neighbors are defined in routing space (WJ at readout r), a_i is the normalized final-answer basin label, $p_{K_{\text{nn}},r}$ is the raw neighbor purity, and b is the chance rate at which two rollouts share a basin, so $A_{K_{\text{nn}},r}$ is chance-corrected purity: 0 at chance, 1 at perfect local alignment. RAD never reads this diagnostic at runtime; implementation details (degenerate cohorts, chance-baseline forms, ground-truth vs. majority robustness) are in Appendix A.6. After selection, we report accuracy / pass@1, token saving, and lift over the exact random 95% envelope.

3.2 EXP I: THE SAME TOKEN CAN BE GENERATED BY DIFFERENT EXPERTS

A first concern is that anchor-conditioned routing might simply recover token identity. Exp I therefore holds the emitted anchor token id fixed across rollouts and tests whether the prefix-causal routes that produce it (which do not condition on the anchor token itself, §3.1) still separate upstream factors that token identity cannot encode. Figure 1 shows this with pairwise WJ distances over per-rollout sparse expert-weight histograms (color labels are used only for visualization, not to fit the embedding). **Qwen EOS token:** at the same <|im_end|> token on BRUMO25, the generating route preserves problem identity across ten random problems, with less than 5% cross-problem neighborhood leakage. **OSS boundary anchor:** at the Harmony final-channel triplet <|channel|>final<|message|> on AIME25, it separates Low, Medium, and High reasoning-budget settings, with less than 1% cross-level contamination. The effect is not specific to protocol tokens: at the common discourse token `So` the same readout selects $+3.4$ pp above random at full budget (App. A.9).

Insight 1. *Token identity is not a sufficient statistic for routing state.* Even with the emitted token id held fixed, the generating experts retain information about task family, problem identity, and reasoning-budget mode, because the upstream context driving that generation step differs; anchor-conditioned routing is therefore not vacuous.

3.3 EXP II: ROUTING NEIGHBORHOODS ALIGN WITH ANSWER BASINS, STRONGEST AT DELIMITER WINDOWS

Exp I is a validity test, not yet a controller test: a route representation can separate tasks or budget modes without being useful for multi-rollout selection. Exp II therefore asks whether the non-token residual aligns with the answer-basin structure a controller can exploit, near final-answer production; only an affirmative answer licenses RAD (§4). We use $A_{K_{\mathrm{nn}},r}$ (Eq. 2) as a post-hoc diagnostic: high values mean nearby rollouts in routing space tend to share a normalized final-answer basin (robustness checks, including ground-truth-labelled variants, in Appendix A.6).

Anchor dependence. Figure 2 (left) compares five routing readouts: two length-matched random controls, a pre-delimiter 16-token window, the marker-only `delimiter@marker` readout, and the headline `delimiter@16` window. The `delimiter@16` window used by RAD wins by a wide margin, `delimiter@marker` already carries nontrivial state without conditioning on answer contents, and the two random controls are at chance. Among the four length-matched 16-token readouts, the gain is driven by *where* routing is read, not merely by how many states are averaged (anchor-round trajectory in Appendix A.6).

Accuracy correspondence. Figure 2 (right) compares `delimiter@16` basin purity with routing-density selector accuracy as the per-rollout reading budget grows, $16t$ tokens for $t{=}4$ to 64 (token positions $\{2^6, \ldots, 2^{10}\}$), with the rollout count fixed at $N{=}64$. On the math subset (aime24, aime25, brumo25, hmmt25), purity rises monotonically with t and selector accuracy rises in step (Spearman $\rho{=}{+}1.00$ and $+0.72$), with the selector's accuracy approaching the $t{=}64$ textual-majority baseline near the right edge of the budget range. So when routing neighborhoods better match answer basins, the densest routing basin better approximates the most populated answer basin (which can still be wrong; §6).

Insight 2. *K-NN basin purity quantifies how strongly routing neighborhoods align with final-answer basins.* The alignment is strongest at `delimiter@16` (the marker-only `delimiter@marker` is already informative; Appendix A.2) and tracks selector accuracy across budgets, so around boundary and delimiter anchors the densest routing basin often approximates the densest answer basin.

4 METHOD: ROUTING AGREEMENT DECODING

§3 motivates an anchor-aligned selector parameterized by anchor type and window width: routing must be conditioned on an anchor token id, since identical ids can be generated by different experts (§3.2), and marker-only readouts are already informative while the wider `delimiter@16` aligns best with answer basins (§3.3). RAD therefore reuses the recorded routing in a fixed `delimiter@16` window and selects the densest Weighted-Jaccard K-NN route basin (Figure 3).

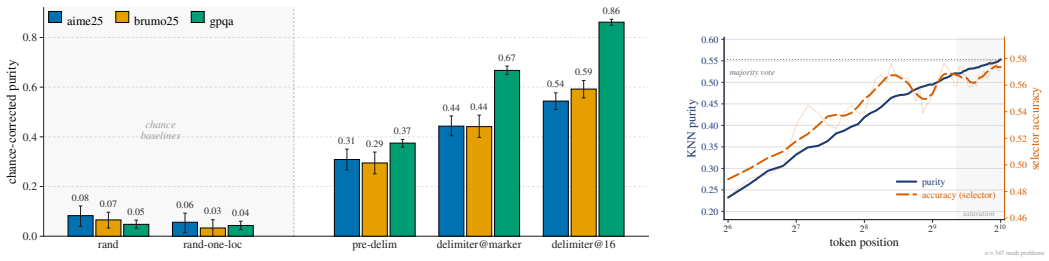


Figure 2: **Delimiter-window routing is the strongest basin-aligned readout, and its alignment tracks selector accuracy.** *Left:* chance-corrected K-NN purity (K_{nn}=1) for five routing readouts (8 MoE models, 1,180 (model, problem) cells, 95% bootstrap CIs); `delimiter@16` reaches 0.86/0.59/0.54 on gpqa/brumo25/aime25. *Right:* math-only `delimiter@16` purity and routing-density selector accuracy vs. $16t$, $t \in [4, 64]$ (347 cells, 8 models $\times$ 4 datasets); dotted = t=64 majority.

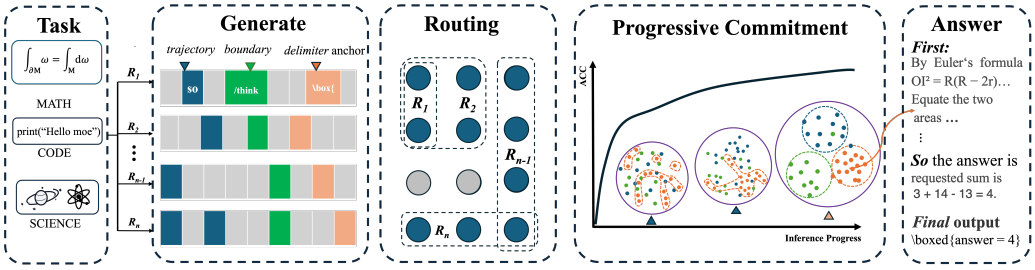


Figure 3: **RAD overview.** N=64 rollouts are aligned by token id at a fixed *trajectory*, *boundary*, or (headline) *delimiter* anchor; each anchor window (`delimiter@16`) is averaged into a routing vector z_i, pairwise Weighted Jaccard forms the consensus matrix, and RAD returns the rollout with the highest K-NN density q_i. Answer strings are read only after selection.

Anchor and window notation. An anchor is a token position used only to align routing readouts across rollouts; a readout is an anchor type (*trajectory*, *boundary*, or *delimiter*; §3.1) plus a window width. Trajectory anchors are used one at a time (by default `So`; `Now` and the paragraph break `.\n\n` are examined in App. A.9). We write `type@W` for the half-open W-token window $[a_i, a_i + W)$ and `type@marker` for the window $[a_i, a_i + m)$ spanning *all* m tokenizer rows of the anchor marker; rows are averaged per token (Eq. 4), so markers of different lengths stay comparable. Under the prefix-causal index (§3.1) a marker row conditions at most on earlier marker tokens, so `delimiter@marker` is a marker-only readout; the headline `delimiter@16` extends past the marker (16>m), so its later rows condition on the first answer-opening token positions, and it is used because it gives the strongest empirical selection signal.

Runtime interface and access boundary. For a problem p, let $\mathcal{D}_p = \{(y_i, R_i)\}_{i=1}^{N}$ be the completed rollout pool, with y_i the generated token sequence and $R_i(t)$ the prefix-causal route vector of §3.1 (*not* conditioned on $y_{i,t}$ itself). RAD is *answer-string-free* in all variants: it uses token ids only to locate anchors and routing states only from the fixed readout window, and it never parses, normalizes, compares, executes, or votes over final answer strings, nor uses correctness labels, execution results, textual majority labels, or answer-token ids as features (Appendix A.1). Its access level is configurable: `boundary@marker` and `delimiter@marker` are strict no-answer-content readouts, while the headline `delimiter@16` uses a stronger answer-opening routing window.

4.1 Routing Agreement Decoding (RAD)

RAD implements the distributional implication of Insight 2: it reads routing from a specified anchor-window readout and selects the densest K-NN route basin.

For each rollout i, let a_i be the delimiter anchor located by the dataset-specific answer-opening rule, the final answer-opening delimiter: for math and GPQA the boxed-answer marker, and for LiveCodeBench the code-output marker (occurrence rules and fallbacks in Appendix A.7). RAD forms the half-open anchor window

$$\mathcal{W}_i = \{a_i,\, a_i + 1,\, \ldots,\, a_i + W - 1\}, \qquad W = 16. \tag{3}$$

The rollout representation is the per-token average over this window (each per-layer block of z_i then sums to one; see §3.1):

$$z_i \;=\; \frac{1}{|\mathcal{W}_i|} \sum_{t \in \mathcal{W}_i} R_i(t). \tag{4}$$

Pairwise similarities among $\{z_i\}_{i=1}^{N}$ form a Weighted-Jaccard consensus matrix $S \in [0,1]^{N \times N}$ with $S_{ij} = \mathrm{WJ}(z_i, z_j)$ (Eq. 1). For each rollout, let $\mathrm{KNN}_{k_{\mathrm{rad}}}(i)$ be its k_{rad} most similar other rollouts under S; the user-facing default is $k_{\mathrm{rad}} = 10$ on the full $N = 64$ pool. RAD assigns each rollout a local agreement density and selects the densest:

$$q_i \;=\; \sum_{j \in \mathrm{KNN}_{k_{\mathrm{rad}}}(i)} S_{ij}, \qquad i^\star \;=\; \arg\max_i q_i. \tag{5}$$

The K-NN density objective is deliberately local: a global medoid can be pulled toward the space between several basins when multiple answer candidates are common, whereas q_i rewards being surrounded by many nearby rollouts and therefore selects a representative of one high-mass route basin, not an independently verified solution. **Majority counts answer strings; RAD aggregates routing agreement.**

Agentic instantiation. The same density selector carries to multi-step agentic rollouts (best-of-N patch selection on SWE-bench Verified, §5.2, App. A.10), with two changes forced by the setting rather than tuned. Agentic outputs open no `\boxed{`/code fence, so we use the *boundary* anchor (the thinking→action transition), unioning each rollout's per-step boundary rows into one routing fingerprint; and k_{rad} tracks the pool size (k_{rad}=4 for the N=16 agentic pools vs. 10 for N=64). We read the binary form of the kernel (set membership over fired (layer, expert) pairs), which §5.3 finds statistically indistinguishable from Weighted Jaccard; the K-NN density q_i and the answer-string-free access boundary are unchanged.

Confidence fusion (RAD+DC). RAD+DC uses DeepConf's per-rollout confidence scalar as a complementary signal, but not DeepConf's answer-string vote: rollouts are first filtered to the highest-confidence half (top-50% by cBW), and the RAD density is computed within the retained subset (Appendix A.15). Unlike the original DeepConf vote over normalized answer strings, which is degenerate for raw code outputs, RAD+DC stays well-defined on code: it uses only the scalar confidence plus routing density and outputs a single selected rollout.

5 Experiments

5.1 Setup, baselines, and exact random envelopes

We evaluate all selectors on the same fixed N=64 rollout pool per problem (each method's candidate-set eligibility and coverage are detailed in Appendix A.7).

Datasets and models. The full pool contains 10 sparse MoE reasoning model configurations and 6 datasets: AIME24/25, BRUMO25, HMMT25, GPQA (Rein et al., 2023), and LiveCodeBench v5 (Jain et al., 2025), for 4,850 model–problem cells and 310,400 generated runs. The models cover **gpt-oss** (20B / 120B) (OpenAI, 2025), **Qwen3-30B-A3B**, and **Qwen3-Next-80B-A3B** (Yang et al., 2025), spanning top-4, top-8, and top-10 routing regimes (Apps. A.5, A.5). We also evaluate a separate *agentic* regime, best-of-N patch selection on SWE-bench Verified (Jimenez et al., 2024; OpenAI, 2024), with three coding-agent backbones under their native harnesses (gpt-oss 20B/120B under *harmonyagent*, Qwen3.6-35B-A3B under *mini-swe-agent*; Appendix A.10), where each rollout is a trajectory ending in a code patch.

Inference setup. Rollouts use temperature 0.7, top-p 0.9, and maximum context length 32,768. We capture top-k expert ids and post-softmax routing weights from vLLM during generation (Appendix A.5).

Baselines. Baselines include random / Avg@64, textual self-consistency Majority, and the original DeepConf confidence-weighted vote (Fu et al., 2026). We distinguish *DeepConf-vote*, which votes over normalized answer classes and is therefore reported only where string voting is well-posed, from the scalar DeepConf confidence score used inside RAD+DC. Answers are parsed with `math_verify`/SymPy (Kydlíček et al., 2025) (GPQA: option letter); Majority ties break to the earliest class in the fixed rollout order, and a cell with no parseable answer counts as wrong for every method (App. A.7).

5.2 Results

Pool-answer selection. RAD improves over single-rollout accuracy across model groups (substantially in the well-posed math+GPQA regime, more modestly on code) and, where string voting is well-posed, is on par with textual Majority, modestly ahead in our pools (Table 1): pooled problem-weighted average Majority 73.6 vs. RAD 73.9 / RAD+DC 74.2; on the four math sets alone RAD 78.3 vs. Majority 76.6 (41 vs. 20 discordant cells of 1,200), on GPQA 71.2 vs. 71.8. The paired natural-pool check (n=3180) gives RAD−Majority $+0.28$ pp (bootstrap 95% CI $[-0.50, +1.07]$; McNemar p=0.52) and RAD+DC−Majority $+0.57$ pp ($[-0.22, +1.35]$; p=0.18). On code, raw outputs are near-unique strings, so exact-string Majority and DeepConf-vote collapse to near-singletons ("–"), while RAD and RAD+DC still produce direct pass@1 selections above the Avg@64 floor. The edge does not depend on reading answer-opening tokens: the strict `delimiter@marker` readout (no answer-region tokens) already reaches RAD 71.1 / RAD+DC 72.8 on the well-posed pool and 65.3 on code, `delimiter@16` adding only ∼1–3 pp (Table 3). RAD is thus competitive where voting works, defined where it does not.

Where the gains come from, and stronger baselines. An instance-level audit (App. A.3) shows what RAD adds: it is correct alone on 83 cells and Majority alone on 74; only 6 of the 83 involve answer merging or parsing, and in 72 strict cases RAD picks a textual-minority answer (vote rank 2/3/≥4 in 55/8/9 cases) that a same-rank random pick would get right 2–3× less often, so its wins are recovered minority basins, not normalization artifacts. Against *execution-based* functional majority voting on LiveCodeBench (FMV, Launer et al., 2026: 16 generated test inputs per problem, 93.1% coverage of the 1,670 cells; App. A.4), RAD with no execution is level on the full denominator (66.7 vs. 66.9); FMV leads by $+3.8$ pp on the cells it covers, and RAD applied within FMV's winning execution class (FMV+RAD) improves FMV by $+1.8$ pp (exact McNemar p=0.008), so the two signals are complementary. Against the decoding-time selector CoT-Decoding (Wang & Zhou, 2024) on gpt-oss-20B-Medium (AIME24/25, HMMT25), RAD@10 reaches 74.3 vs. CoT-max 70.0 and CoT-greedy 57.8 at equal candidate count (App. A.4).

Agentic best-of-N: SWE-bench Verified. Code generation already strains string voting; the agentic setting removes the answer string entirely, and this is where the interface gap is widest: each candidate is a multi-step trajectory whose output is a code patch scored pass/fail by a hidden test suite, so textual Majority and DeepConf-vote degenerate (near-unique patches collapse to singletons). We apply the agentic instantiation of §4.1 (boundary anchor, per-step union, binary K-NN at k=4), anchoring at each model's thinking→action transition (`</think>` for Qwen3.6-35B-A3B, `<|start|>` for gpt-oss). Over N=16 rollouts on 500 problems per model, RAD improves best-of-N selection by $+4.7/+4.8/+5.0$ pp on the decidable subset of problems whose rollouts split, and by $+2.2$ to $+2.9$ pp end-to-end over the random Avg@16 floor (Table 2; per-model setup and analysis in Appendix A.10). One recipe (a single boundary anchor, one k, no per-model tuning) transfers across all three backbones.

5.3 Analysis

Robustness and cost. Binary and weighted Jaccard are indistinguishable (pooled gap 0.22 pp, exact McNemar p=0.25, powered to exclude ≥ 1 pp), which licenses the binary fingerprint on SWE-bench; the shallowest tenth of layers matches the full stack within 1.7 pp; and early trajectory anchors

Table 1: **Pool-answer accuracy (%) across four model families**, pooled over reasoning-effort / thinking–instruct variants (sub-label: routed experts, top-k). *Avg@64*: single random rollout; *Cons@64*: textual Majority; *DeepConf*: its confidence-weighted vote; *RAD*: the `delimiter@16` selector (§4.1); *RAD+DC*: DeepConf-filtered RAD. **Bold**: best per column within a family; green (+Δ): gain over Avg@64; "–": exact-string votes are undefined on raw code. *Math avg.*: mean of the four math sets (30 problems each); *Avg.*: problem-weighted math+GPQA average.

Model	Method	Math Reasoning					GPQA	Avg.	Code
		AIME24	AIME25	BRUMO25	HMMT25	avg.		math+GPQA	
gpt-oss-20B 32 experts, top-4 Low/Med/High avg.	Avg@64	65.0	59.5	64.2	41.2	57.5	60.0	59.0	50.8
	Cons@64	78.9 (+13.9)	73.3 (+13.9)	**76.7** (+12.5)	52.2 (+11.0)	70.3	**67.0** (+7.0)	68.2 (+9.2)	–
	DeepConf	78.9 (+13.9)	73.3 (+13.9)	74.4 (+10.3)	54.4 (+13.2)	70.3	66.8 (+6.9)	68.1 (+9.1)	–
	RAD	**81.1** (+16.1)	**74.4** (+15.0)	**76.7** (+12.5)	60.0 (+18.8)	**73.1**	65.3 (+5.3)	68.2 (+9.2)	60.5 (+9.7)
	RAD+DC	80.0 (+15.0)	72.2 (+12.8)	**76.7** (+12.5)	**62.2** (+21.0)	72.8	66.0 (+6.0)	**68.6** (+9.5)	**61.3** (+10.5)
gpt-oss-120B 128 experts, top-4 Low/Med/High avg.	Avg@64	73.4	71.7	72.5	55.9	68.4	70.6	69.7	70.6
	Cons@64	83.3 (+9.9)	81.1 (+9.4)	**80.0** (+7.5)	71.1 (+15.2)	78.9	74.1 (+3.5)	75.9 (+6.1)	–
	DeepConf	**84.4** (+11.1)	82.2 (+10.5)	**80.0** (+7.5)	**73.3** (+17.4)	**80.0**	73.9 (+3.3)	**76.2** (+6.5)	–
	RAD	83.3 (+9.9)	**83.3** (+11.6)	77.8 (+5.2)	70.0 (+14.1)	78.6	**74.4** (+3.8)	76.0 (+6.3)	**77.0** (+6.5)
	RAD+DC	82.2 (+8.8)	**83.3** (+11.6)	**80.0** (+7.5)	71.1 (+15.2)	79.2	73.7 (+3.2)	75.8 (+6.0)	76.4 (+5.9)
Qwen3-30B-A3B 128 experts, top-8 Think/Instruct avg.	Avg@64	79.7	70.7	77.4	51.6	69.9	68.6	69.1	58.2
	Cons@64	88.3 (+8.6)	81.7 (+11.0)	86.7 (+9.3)	58.3 (+6.7)	78.8	70.2 (+1.6)	73.4 (+4.3)	–
	DeepConf	88.3 (+8.6)	80.0 (+9.3)	86.7 (+9.3)	58.3 (+6.7)	78.3	70.5 (+1.8)	73.4 (+4.3)	–
	RAD	**90.0** (+10.3)	**85.0** (+14.3)	88.3 (+10.9)	**63.3** (+11.7)	81.7	**70.7** (+2.1)	**74.8** (+5.8)	59.0 (+0.8)
	RAD+DC	**90.0** (+10.3)	**85.0** (+14.3)	**90.0** (+12.6)	**63.3** (+11.7)	**82.1**	70.2 (+1.6)	74.7 (+5.6)	**59.9** (+1.7)
Qwen3-Next-80B 512+shared experts, top-10 Think/Instruct avg.	Avg@64	83.9	75.2	80.6	57.0	74.2	66.5	69.4	66.5
	Cons@64	**91.7** (+7.8)	83.3 (+8.2)	81.7 (+1.1)	65.0 (+8.0)	80.4	77.0 (+10.5)	78.3 (+8.9)	–
	DeepConf	**91.7** (+7.8)	**85.0** (+9.8)	85.0 (+4.4)	65.0 (+8.0)	81.7	**77.3** (+10.8)	78.9 (+9.5)	–
	RAD	90.0 (+6.1)	83.3 (+8.2)	90.0 (+9.4)	66.7 (+9.7)	82.5	75.5 (+9.0)	78.1 (+8.8)	67.1 (+0.6)
	RAD+DC	**91.7** (+7.8)	83.3 (+8.2)	**91.7** (+11.1)	**71.7** (+14.7)	**84.6**	76.5 (+10.0)	**79.6** (+10.2)	**69.2** (+2.7)

Table 2: **Agentic best-of-N selection on SWE-bench Verified** (N=16 rollouts × 500 problems per model). Candidates are multi-step code patches scored pass/fail by the hidden test suite; exact-string votes over near-unique patches degenerate, so the baseline is random selection (*Avg@16*, whose expectation is the pool mean pass rate). RAD reads the `boundary@marker` anchor (the thinking→action transition) and selects the densest binary-Jaccard K-NN basin (K=4). *Mixed sel@1*: accuracy on the decidable subset whose 16 rollouts split (n_{mix} problems, the only ones where selection can change the outcome); *Avg@all*: end-to-end accuracy over all 500. Per-model setup in Appendix A.10.

Model	Anchor	**Mixed sel@1** (decidable subset)				**Avg@all** (500)		
		n_{mix}	Avg@16	RAD	Δ	Avg@16	RAD	Δ
Qwen3.6-35B-A3B	`</think>`	233	55.8	**60.5**	+4.7	49.2	**51.4**	+2.2
gpt-oss-120B	`<|start|>`	247	58.7	**63.6**	+4.8	55.4	**57.8**	+2.4
gpt-oss-20B	`<|start|>`	286	52.0	**57.0**	+5.0	41.3	**44.2**	+2.9

trade signal for coverage until $B \approx$ 2–4k tokens (Apps. A.13, A.14, A.9). Capture, selection, and deployment costs are reported in App. A.5.

What RAD selects: a soft consensus vote, not a verifier. Two controls (both panels of Figure 4) together show RAD is a soft consensus signal rather than a verifier: one for *when* it helps (left), one for *what* it selects (right). *When it helps.* RAD helps *only when the parallel rollouts disagree*: across the ten models Δ_{sel} falls as cohort accuracy rises (Figure 4, left; the cross-model slope is largely a headroom effect), and within a model a problem's advantage rises with its answer disagreement (partial correlation +0.28, $p = 2.9 \times 10^{-59}$, difficulty partialled out; App. A.11). Reasoning effort and thinking-vs-instruct are knobs on this axis: higher agreement shrinks the absolute gain as a ceiling effect, while Δ_{full} stays $\sim$ +12–15 pp at every effort with RAD at or above Majority. The agentic gain follows the same rule: on SWE-bench Verified only problems whose 16 rollouts split can move, and the +4.7–5.0 pp there dilutes to +2.2–2.9 pp end-to-end (Table 2).

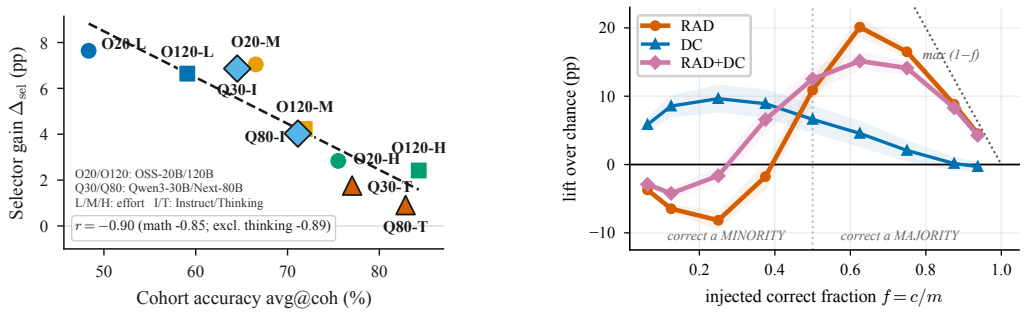


Figure 4: **RAD is a soft consensus vote, not a verifier.** (Left) Selector gain Δ_{sel} shrinks as cohorts converge (color = effort, shape = model). (Right) Composition control: RAD is below chance when correct rollouts are a minority and positive once they are a majority, while DeepConf is the complementary rare-correct signal. Bands: 95% cluster-bootstrap CIs.

What it selects. A composition control on a difficulty-balanced fixed set (m=16, n=703) injects exactly c correct rollouts and measures the pick's accuracy minus the random-pick baseline $f{=}c/m$ (Figure 4, right). RAD's lift over chance is a clean *S-curve in* f: below chance when correct answers are a minority, crossing zero near $f{\approx}0.4$, and large-positive once they are a majority, with an amplitude that grows with the pool size (App. A.15). RAD is therefore a soft, answer-string-free *consensus* vote: it tracks the most populated route basin, not correctness, and is *below random* under a wrong majority. DeepConf confidence is the complementary per-rollout *quality* signal, positive precisely in the rare-correct regime where RAD fails, and the two are distinct signals: RAD and DeepConf-argmax pick the same rollout only at chance ($\kappa_{\text{rand}} = -0.006$; within-problem Spearman ρ=0.05; App. A.3). Fused, RAD+DC statistically *ties* answer-aware Majority without reading an answer string (paired gap +0.1 pp $[-0.8, +1.1]$ delimiter, −0.4 pp $[-1.5, +0.6]$ boundary; full statistics and the per-domain breakdown, concentrated on mathematics, in App. A.15).

6 Limitations

RAD is an answer-string-free arbitration rule, not a verifier: a dense route basin can still be wrong, exactly as textual Majority can follow a dense wrong consensus. It requires MoE-router access, so it does not apply to dense models or closed APIs, and we retained no matched hidden-state or neuron features, so routing is shown to be sufficient, not uniquely informative, and the cost contrast with neuron-level readouts (§1) is architectural, not measured head-to-head. Anchors are needed only after generation: an EOS-aligned readout keeps ∼86% of the marker gain (70.4 vs. 71.3, random 64.6; App. A.7), but open-ended long-form QA (e.g. ASQA, Stelmakh et al., 2022) is unevaluated, and reading routing before the answer remains open: prefix readouts of the first 32–256 tokens stay near random (64.8–65.2 vs. 71.3; App. A.12). The headline `delimiter@16` window conditions on answer-opening tokens; the strict `delimiter@marker` readout recovers most of the lift (App. A.2). The edge over textual Majority in the canonical-answer regime is small and not significant (+0.28 pp, McNemar p=0.52), concentrated on math and on some configurations, and reverses at smaller pools (RAD trails Majority by 0.2–0.8 pp for $m{\leq}32$; App. A.15); the robust value is the interface. The agentic study is smaller in scale (best-of-16, one pool per model) and far from the any-of-16 oracle (App. A.10).

7 Conclusion

MoE inference already computes a routing record; RAD reuses it. The same token id is produced by different experts, the structure is strongest at boundary and delimiter readouts, and the selector is on par with textual Majority where voting is well-posed, level with execution-based voting on code without executing anything, and better than random on SWE-bench patch selection: an agreement signal, not a verifier. **Majority counts answer strings; RAD aggregates routing agreement.**

AI USE STATEMENT

In this work, we used generative AI tools for implementing methods: AI coding assistants helped write the code for routing capture, the RAD selector, and the analysis pipeline. We did not use generative AI tools for any other task with required disclosure, namely producing synthetic data, building theoretical models or conceptual frameworks, formulating mathematical claims or proofs, proposing or refining hypotheses, designing the methodology or experiments, translation, cleaning or reformatting data, qualitative analysis, or interpreting results. Additionally, we used generative AI tools for tasks with recommended disclosure: searching and summarizing related literature, brainstorming, converting the LaTeX sources of this paper to the ICLR 2027 template, condensing the main text to meet the page limit, drafting the ethics and reproducibility statements, drafting appendix sections and the corresponding main-text sentences that report additional analyses from the authors' own result tables, and restructuring the introduction. We have reviewed all AI-assisted work: AI-assisted code was verified by tests and by reproducing the reported numbers, and all AI-edited text was checked by the authors. We take responsibility for the final content of this work, including text, claims, or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This work analyzes open-weight sparse MoE language models (gpt-oss and Qwen3-MoE) on public benchmarks (AIME 2024/2025, BRUMO 2025, HMMT 2025, GPQA, LiveCodeBench v5, and SWE-bench Verified). It involves no human subjects, crowdsourcing, or personal data, and it releases no new dataset. RAD selects among a model's own completed rollouts; like textual majority voting it is not a verifier, and a dense but wrong route basin can be selected (§6), so it should not be used as a correctness guarantee in high-stakes settings.

REPRODUCIBILITY STATEMENT

The rollout protocol, route representation, and diagnostic metric are specified in §3.1, the selector in §4.1 (Eqs. 3–5), and the inference setup and baselines in §5.1. The appendix gives the answer-string-free access boundary (Appendix A.1), MoE routing capture in vLLM for each model (Appendix A.5), anchor occurrence rules, fallbacks, and the answer-extraction protocol of the textual baselines (Appendix A.7), the K-NN purity protocol (Appendix A.6), exact random baselines (Appendix A.8), the instance-level audit and the additional baselines (Appendices A.3, A.4), and the per-model agentic SWE-bench Verified configuration (Appendix A.10). All models and datasets are publicly available, and the code for routing capture and the RAD selector will be released upon publication.

REFERENCES

Trenton Bricken, Adly Templeton, Joshua Batson, Brian Chen, Adam Jermyn, Tom Conerly, Nick Turner, Cem Anil, Carson Denison, Amanda Askell, Robert Lasenby, Yifan Wu, Shauna Kravec, Nicholas Schiefer, Tim Maxwell, Nicholas Joseph, Zac Hatfield-Dodds, Alex Tamkin, Karina Nguyen, Brayden McLean, Josiah E. Burke, Tristan Hume, Shan Carter, Tom Henighan, and Christopher Olah. Towards monosemanticity: Decomposing language models with dictionary learning. Transformer Circuits Thread, 2023. URL https://transformer-circuits.pub/2023/monosemantic-features/index.html.

Yixin Cao, Jiahao Ying, Yaoning Wang, Xipeng Qiu, Xuanjing Huang, and Yugang Jiang. Model utility law: Evaluating LLMs beyond performance through mechanism interpretable metric. arXiv preprint arXiv:2504.07440, 2025. URL https://arxiv.org/abs/2504.07440.

Bei Chen, Fengji Zhang, Anh Nguyen, Daoguang Zan, Zeqi Lin, Jian-Guang Lou, and Weizhu Chen. CodeT: Code generation with generated tests. In *International Conference on Learning Representations (ICLR)*, 2023.

Kang Chen, Yaoning Wang, Kai Xiong, Zhuoka Feng, Wenhe Sun, Haotian Chen, and Yixin Cao. Do LLMs signal when they're right? evidence from neuron agreement, 2025. URL https://arxiv.org/abs/2510.26277.

Kang Chen, Zhuoka Feng, Sihan Zhao, Kai Xiong, Junjie Nian, Yaoning Wang, Changyi Xiao, and Yixin Cao. NEX: Neuron explore-exploit scoring for label-free chain-of-thought selection and model ranking. arXiv preprint arXiv:2602.05805, 2026. URL https://arxiv.org/abs/2602.05805.

Hoagy Cunningham, Aidan Ewart, Logan Riggs, Robert Huben, and Lee Sharkey. Sparse autoencoders find highly interpretable features in language models. In *International Conference on Learning Representations*, 2024. URL https://arxiv.org/abs/2309.08600.

Damai Dai, Chengqi Deng, Chenggang Zhao, R.x. Xu, Huazuo Gao, Deli Chen, Jiashi Li, Wangding Zeng, Xingkai Yu, Y. Wu, Zhenda Xie, Y.k. Li, Panpan Huang, Fuli Luo, Chong Ruan, Zhifang Sui, and Wenfeng Liang. DeepSeekMoE: Towards ultimate expert specialization in mixture-of-experts language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1280–1297, Bangkok, Thailand, 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.70. URL https://aclanthology.org/2024.acl-long.70/.

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. URL https://jmlr.org/papers/v23/21-0998.html.

Yichao Fu, Xuewei Wang, Hao Zhang, Yuandong Tian, and Jiawei Zhao. Deep think with confidence. In *International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=8LqHs0KIM7.

Hojae Han, Jaejin Kim, Jaeseok Yoo, Youngwon Lee, and Seung-won Hwang. ArchCode: Incorporating software requirements in code generation with large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 13520–13552, 2024.

Shun-ichiro Hayashi, Daichi Mukunoki, Tetsuya Hoshino, and Takahiro Katagiri. Layer-wise MoE routing locality under shared-prefix code generation: Token-identity decomposition and compile-equivalent fork redundancy. arXiv preprint arXiv:2604.17182, 2026. URL https://arxiv.org/abs/2604.17182.

Colin Hong, Xu Guo, Anand Chaanan Singh, Esha Choukse, and Dmitrii Ustiugov. Slim-SC: Thought pruning for efficient scaling with self-consistency. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 34500–34517, Suzhou, China, 2025. Association for Computational Linguistics. doi: 10.18653/v1/2025.emnlp-main.1750. URL https://aclanthology.org/2025.emnlp-main.1750/.

Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. LiveCodeBench: Holistic and contamination free evaluation of large language models for code. In *International Conference on Learning Representations (ICLR)*, 2025. arXiv:2403.07974.

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, Gianna Lengyel, Guillaume Bour, Guillaume Lample, Lélio Renard Lavaud, Lucile Saulnier, Marie-Anne Lachaux, Pierre Stock, Sandeep Subramanian, Sophia Yang, Szymon Antoniak, Teven Le Scao, Théophile Gervet, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. Mixtral of experts. arXiv preprint arXiv:2401.04088, 2024. URL https://arxiv.org/abs/2401.04088.

Carlos E. Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. SWE-bench: Can language models resolve real-world GitHub issues? In *International Conference on Learning Representations (ICLR)*, 2024. URL https://arxiv.org/abs/2310.06770.

Ishan Jindal, Sai Prashanth Akuthota, Jayant Taneja, and Sachin Dev Sharma. The path of least resistance: Guiding LLM reasoning trajectories with prefix consensus. In *International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=hrnSqERgPn.

Dominik Kundel. OpenAI Harmony Response Format, August 5 2025. URL https://developers.openai.com/cookbook/articles/openai-harmony. OpenAI Cookbook. Accessed 2026-05-07.

Hynek Kydlíček, Alina Lozovskaya, Nathan Habib, and Clémentine Fourrier. Math-verify. https://github.com/huggingface/Math-Verify, 2025.

Tim Launer, Jonas Hübotter, Marco Bagatella, Ido Hakimi, and Andreas Krause. Majority voting for code generation. In *Workshop on Test-Time Updates at the International Conference on Learning Representations (ICLR)*, 2026. arXiv:2604.15618.

Dmitry Lepikhin, HyoukJoong Lee, Yuanzhong Xu, Dehao Chen, Orhan Firat, Yanping Huang, Maxim Krikun, Noam Shazeer, and Zhifeng Chen. GShard: Scaling giant models with conditional computation and automatic sharding. In *International Conference on Learning Representations*, 2021. URL https://arxiv.org/abs/2006.16668.

Yiwei Li, Peiwen Yuan, Shaoxiong Feng, Boyuan Pan, Xinglin Wang, Bin Sun, Heda Wang, and Kan Li. Escape sky-high cost: Early-stopping self-consistency for multi-step reasoning. In *International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=ndR8Ytrzhh.

Borislav Mavrin. In harmony with gpt-oss. arXiv preprint arXiv:2604.00362, 2026. URL https://arxiv.org/abs/2604.00362.

OpenAI. Introducing SWE-bench verified. OpenAI blog, 2024. URL https://openai.com/index/introducing-swe-bench-verified/.

OpenAI. gpt-oss-120b & gpt-oss-20b Model Card, August 5 2025. URL https://cdn.openai.com/pdf/419b6906-9da6-406c-a19d-1bb078ac7637/oai_gpt-oss_model_card.pdf. Accessed 2026-05-07.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. Gpqa: A graduate-level google-proof q&a benchmark. *arXiv preprint arXiv:2311.12022*, 2023. URL https://arxiv.org/abs/2311.12022.

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=B1ckMDqlg.

Ivan Stelmakh, Yi Luan, Bhuwan Dhingra, and Ming-Wei Chang. ASQA: Factoid questions meet long-form answers. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2022.

Guangya Wan, Yuqi Wu, Jie Chen, and Sheng Li. Reasoning aware self-consistency: Leveraging reasoning paths for efficient LLM sampling. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 3613–3635, Albuquerque, New Mexico, 2025. Association for Computational Linguistics. doi: 10.18653/v1/2025.naacl-long.184. URL https://aclanthology.org/2025.naacl-long.184/.

Xi Wang, Soufiane Hayou, and Eric Nalisnick. The myth of expert specialization in MoEs: Why routing reflects geometry, not necessarily domain expertise. arXiv preprint arXiv:2604.09780, 2026. URL https://arxiv.org/abs/2604.09780.

Xuezhi Wang and Denny Zhou. Chain-of-thought reasoning without prompting. In *Advances in Neural Information Processing Systems*, volume 37, 2024. arXiv:2402.10200.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. In *International Conference on Learning Representations (ICLR)*, 2023. URL https://arxiv.org/abs/2203.11171.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc V. Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems*, 35, 2022. URL https://arxiv.org/abs/2201.11903.

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. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 55625–55655. PMLR, 2024. URL https://proceedings.mlr.press/v235/xue24c.html.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, et al. Qwen3 technical report, 2025. arXiv:2505.09388.

John Yang, Carlos E. Jimenez, Alexander Wettig, Kilian Lieret, Shunyu Yao, Karthik Narasimhan, and Ofir Press. SWE-agent: Agent-computer interfaces enable automated software engineering. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024. URL https://arxiv.org/abs/2405.15793.

Jiahao Ying, Mingbao Lin, Qianru Sun, and Yixin Cao. Beyond benchmarks: Understanding mixture-of-experts models through internal mechanisms. arXiv preprint arXiv:2509.23933, 2025. URL https://arxiv.org/abs/2509.23933.

Anqi Zhang, Yulin Chen, Jane Pan, Chen Zhao, Aurojit Panda, Jinyang Li, and He He. Reasoning models know when they're right: Probing hidden states for self-verification. arXiv preprint arXiv:2504.05419, 2025. URL https://arxiv.org/abs/2504.05419.

A Technical appendices and supplementary material

This appendix contains the full answer-string-free protocol details, exact-statistics derivations, anchor coverage tables, ablation studies, failure-case analyses, and per-dataset / per-model results that the main text references but does not include for space.

A.1 Answer-string-free protocol details and access boundary

This appendix makes the answer-string-free access policy precise, as referenced from the protocol (§3.1). The selector is allowed to read: token ids (used solely to locate anchors), routing tensors in the fixed anchor window, generated rollout length, and anchor occurrence counts. The selector is forbidden to read: normalized final-answer strings, textual majority labels, correctness labels, code-execution outputs, the surface-string contents of boxed-answer or code-output spans, or answer-token ids inside the window as features. RAD has configurable access levels: `boundary@marker` and `delimiter@marker` avoid answer-content-conditioned rows, while the headline `delimiter@16` readout uses a stronger answer-opening routing trace.

The routing index is prefix-causal. For a generated sequence y_i, the recorded row $R_i(t)$ is captured in the forward pass that predicts $y_{i,t}$ from $y_{i,<t}$; it is therefore aligned with `tok_token_ids`$[t]$, `tok_sampled_logprob`$[t]$, and `tok_topk_logprobs`$[t]$, but it does not condition on $y_{i,t}$ itself. A readout `type@W` is half-open, $\mathcal{W}_i = [a_i, a_i + W)$ with window width W (Eq. 3). For a delimiter anchor of token length m, `delimiter@marker` reads the m marker rows $R_i(a_i), \ldots, R_i(a_i{+}m{-}1)$: the first predicts the opening delimiter token and conditions on neither the delimiter nor any answer content, and each later marker row conditions only on *earlier marker tokens* (the fixed delimiter string), so `delimiter@marker` conditions on no answer-region content. And `delimiter@16` uses the same delimiter anchor with a wider fixed window, $\mathcal{W}_i = [a_i, a_i{+}16)$. When $m < 16$, the additional rows are co-located with the first few answer-opening token positions. The resulting distinction is an access–signal choice: `delimiter@marker` isolates the fixed marker rows, whereas `delimiter@16` keeps the same anchor but widens the routing window to answer-opening positions. Throughout the paper, *answer-string-free* denotes this interface-level property: selection is made from routing geometry rather than from parsed, normalized, compared, executed, or voted-over final answer strings. Route density can correlate with answer basins because the model's routing state near the committed answer carries basin information; this is the mechanism under study.

A.2 STRICT MARKER-ONLY READOUTS: ROUTING AGREEMENT WITHOUT ANSWER-REGION TOKENS

A fair objection to the headline `delimiter@16` setting is that its window conditions on a few answer-opening tokens (an explicit signal-strength choice, not a dependency; §6 above). Table 3 isolates how much of RAD survives a *strictly* answer-content-free readout: the marker-only `delimiter@marker`, which reads routing only at the answer-opening delimiter token(s) and conditions on no answer-region content. Across all ten configurations and six datasets it selects effectively. Pooled over the well-posed math+GPQA pool it reaches RAD 71.1 / RAD$_{+\mathrm{DC}}$ 72.8, well above the Avg@64 random floor (66.3) and close to textual Majority (73.6), while the wider `delimiter@16` adds only the remaining ∼1–3 pp (the same ordering the K-NN purity diagnostic shows in Fig. 2). The routing-agreement signal is thus already present *before* any answer token is read; the window is a signal-strength knob, not the source of the effect.

This is sharpest on code. A 16-token window opened at a code fence cannot span a full program (it covers roughly the first line) so even `delimiter@16` reads no meaningful portion of the answer on LiveCodeBench, and the marker-only readout makes the point explicit: reading only the fence token(s), RAD still lifts pass@1 to 65.3 (+4.0 over Avg@64) where exact-string Majority degenerates. The agentic setting is strict by construction: SWE-bench Verified (App. A.10) deploys `boundary@marker` (the thinking-to-action boundary, read once per reasoning step and unioned over the rollout's steps, with no answer-opening window) so its patch selection reads no answer content either.

In short, `delimiter@marker` and `boundary@marker` are the access-minimal configurations and the right default whenever no-answer-content routing is required; we report `delimiter@16` as the headline only because it is the strongest selector, at a pooled cost of ∼1–3 pp over the marker-only readout. Weighting is not load-bearing here either: binary Jaccard is equivalent (App. A.13).

A.3 WHERE RAD AND TEXTUAL MAJORITY DISAGREE: AN INSTANCE-LEVEL AUDIT

We audit all 3,180 model–problem instances of the well-posed pool of Table 1 (four math sets, 1,200; GPQA, 1,980; ten configurations), counting an instance on which RAD makes no valid selection as an error (Table 4). Pooled over the 1,200 math cells, Avg@64 66.56, Cons@64 76.58, DeepConf 77.08, RAD 78.33 (RAD correct on 940 cells vs. 919 for Cons: +1.75 pp); on GPQA RAD 71.16 vs. Cons 71.77 (−0.61 pp), which reduces the pooled difference to +0.28 pp.

Not a normalization artifact. Among the 83 RAD-only-correct instances, 3 involve unresolved answer merging and 3 parsing or equivalence issues; 72 are strict cases in which Majority selects an incorrect plurality (38 of the 42 GPQA cases stay strict under normalization). These are not near-ties: only 5 of the 72 involve a tie; the median vote count of RAD's correct answer is 17 against 28.5 for the wrong plurality (median gap 8.5; 47 cases with gap ≥ 5, 31 with gap ≥ 10). RAD's pick is the 2nd, 3rd, or $\geq$4th-ranked textual answer in 55, 8, and 9 cases; a random pick at the same rank is correct 17.0%, 9.3%, and 10.7% of the time versus 36.2%, 28.6%, and 28.9% for RAD's picks.

The signal is local density, and it is symmetric. In RAD-only-correct instances the selected answer's local WJ density exceeds the plurality's by +0.020 (95% CI [+0.007, +0.032]); in Majority-only-correct instances RAD's wrong pick is denser by +0.036 ([+0.023, +0.049]). Majority aggregates global frequency, RAD local density in routing space: it recovers compact correct minority basins and equally follows compact wrong ones. Against DeepConf-vote under the same normalization, RAD is correct alone 63 times and DeepConf-vote alone 80 times.

RAD and DeepConf select differently. DeepConf-argmax (highest-cBW rollout) is the answer-free comparator; DeepConf-vote aggregates cBW over normalized answer classes and picks the same class as Majority on 99.00% of instances. Because answer-class labels are problem-specific we report a problem-stratified chance-corrected agreement κ_{rand}, with chance $p_e = \sum_c (n_c/N)^2$ at the answer level and $1/N$ at the rollout level (Table 5; n=3,114 math+GPQA instances). RAD and DeepConf-argmax choose the same rollout only at chance, and the mean within-problem Spearman correlation between routing density and cBW is ρ=0.051 (95% CI [0.041, 0.060]; median 0.049;

Table 3: **Strict marker-only readout (`delimiter@marker`) across all ten configurations and six datasets.** Density-rank selector (K=10, Weighted-Jaccard) at the answer-opening *delimiter* anchor (`\boxed` on math/GPQA, a code fence on LCBv5), with the routing window reduced to the *marker token(s) themselves* (`delimiter@marker`), so it conditions on *no* answer-region tokens (App. A.1), unlike the headline `delimiter@16` of Table 1. Each cell is density-rank pick accuracy (%); the green/red (±Δ) is the gain over that cell's Avg@64 random-rollout floor. All RAD rows are the marker-only readout. Pooled over the well-posed math+GPQA pool (n=3180, problem-weighted), `delimiter@marker` reaches RAD 71.1 / RAD$_{+\mathrm{DC}}$ 72.8 versus Majority 73.6 and the Avg@64 floor 66.3, i.e. it recovers most of the lift while reading no answer content, and the headline `delimiter@16` adds only the remaining ∼1–3 pp (reaching 73.9 / 74.2). On code (n=1670), where a 16-token window cannot span a full solution and exact-string voting degenerates, `delimiter@marker` still lifts RAD to 65.3 (+4.0 over Avg@64). Weighting is not load-bearing (binary Jaccard is equivalent, App. A.13).

Model	Method	Math Reasoning				GPQA	Avg.	Code
		AIME24	AIME25	BRUMO25	HMMT25		math+GPQA	
gpt-oss-20B Low	**RAD**	50.0 (+6.4)	36.7 (+2.3)	46.7 (+3.4)	23.3 (+4.3)	57.1 (+0.8)	50.3 (+2.0)	35.9 (+2.4)
	RAD$_{+\mathrm{DC}}$	63.3 (+19.7)	43.3 (+9.0)	50.0 (+6.8)	30.0 (+10.9)	59.1 (+2.8)	54.4 (+6.1)	34.7 (+1.2)
gpt-oss-20B Med	**RAD**	80.0 (+5.2)	73.3 (+4.6)	86.7 (+14.1)	56.7 (+6.9)	65.2 (+0.8)	68.6 (+3.4)	70.1 (+2.7)
	RAD$_{+\mathrm{DC}}$	86.7 (+11.8)	86.7 (+18.0)	86.7 (+14.1)	56.7 (+6.9)	68.2 (+3.8)	72.3 (+7.2)	70.7 (+3.3)
gpt-oss-20B High	**RAD**	86.7 (+10.1)	86.7 (+11.4)	90.0 (+13.3)	83.3 (+28.5)	67.2 (+7.9)	74.5 (+10.9)	71.3 (+19.8)
	RAD$_{+\mathrm{DC}}$	86.7 (+10.1)	86.7 (+11.4)	90.0 (+13.3)	76.7 (+21.8)	66.2 (+6.9)	73.3 (+9.6)	67.7 (+16.2)
gpt-oss-120B Low	**RAD**	66.7 (+11.0)	56.7 (+4.4)	60.0 (+3.5)	43.3 (+12.5)	66.7 (+1.4)	62.9 (+3.8)	66.5 (+2.1)
	RAD$_{+\mathrm{DC}}$	53.3 (-2.3)	66.7 (+14.4)	53.3 (-3.2)	36.7 (+5.8)	68.2 (+2.9)	62.3 (+3.2)	67.1 (+2.7)
gpt-oss-120B Med	**RAD**	73.3 (-4.4)	80.0 (+1.9)	73.3 (-2.6)	70.0 (+8.6)	72.2 (+1.2)	73.0 (+1.1)	78.4 (+2.4)
	RAD$_{+\mathrm{DC}}$	80.0 (+2.3)	70.0 (-8.1)	83.3 (+7.4)	76.7 (+15.3)	72.7 (+1.7)	74.5 (+2.7)	76.6 (+0.6)
gpt-oss-120B High	**RAD**	93.3 (+6.5)	90.0 (+5.1)	93.3 (+8.2)	93.3 (+17.8)	80.8 (+5.4)	85.2 (+6.9)	79.0 (+7.7)
	RAD$_{+\mathrm{DC}}$	93.3 (+6.5)	90.0 (+5.1)	93.3 (+8.2)	90.0 (+14.5)	82.8 (+7.4)	86.2 (+7.9)	85.6 (+14.3)
Qwen3-30B-A3B Think	**RAD**	93.3 (+5.9)	93.3 (+12.6)	96.7 (+11.6)	76.7 (+14.7)	70.7 (-0.4)	78.0 (+4.0)	74.3 (+4.9)
	RAD$_{+\mathrm{DC}}$	93.3 (+5.9)	93.3 (+12.6)	96.7 (+11.6)	80.0 (+18.0)	71.2 (+0.1)	78.6 (+4.6)	71.9 (+2.6)
Qwen3-30B-A3B Inst	**RAD**	80.0 (+8.0)	66.7 (+6.1)	76.7 (+7.0)	40.0 (-1.2)	67.7 (+1.5)	67.0 (+2.8)	42.5 (-4.5)
	RAD$_{+\mathrm{DC}}$	86.7 (+14.6)	70.0 (+9.4)	80.0 (+10.3)	43.3 (+2.1)	68.2 (+2.0)	68.9 (+4.7)	44.3 (-2.7)
Qwen3-Next-80B Think	**RAD**	93.3 (+5.1)	90.0 (+7.7)	93.3 (+9.0)	76.7 (+12.9)	76.8 (+16.0)	81.1 (+13.2)	70.7 (+0.5)
	RAD$_{+\mathrm{DC}}$	93.3 (+5.1)	90.0 (+7.7)	96.7 (+12.3)	86.7 (+22.9)	77.3 (+16.5)	82.7 (+14.8)	70.7 (+0.5)
Qwen3-Next-80B Inst	**RAD**	80.0 (+0.5)	73.3 (+5.3)	76.7 (-0.2)	53.3 (+3.1)	70.7 (-1.5)	70.8 (-0.1)	64.7 (+1.8)
	RAD$_{+\mathrm{DC}}$	83.3 (+3.8)	76.7 (+8.6)	83.3 (+6.5)	60.0 (+9.8)	74.7 (+2.5)	75.2 (+4.3)	64.7 (+1.8)

Table 4: **RAD vs. textual Majority, instance-level cross-tabulation** (well-posed pool, n=3,180, `delimiter@16`, K=10). RAD is correct alone 83 times and Majority alone 74 times: net +9 (+0.28 pp).

Pool	Both correct	RAD only	Majority only	Both wrong
Math (1,200)	899	41	20	240
GPQA (1,980)	1,367	42	54	517
All (3,180)	2,266	83	74	757

pooled Pearson 0.043 over 188,608 rollouts; 62% of instances have $|\rho| < 0.2$): precisely estimated but negligible, consistent with Table 31.

A.4 Additional baselines: functional majority voting and CoT-Decoding

Functional majority voting on LiveCodeBench. Functional majority voting (FMV; Launer et al., 2026; cf. CodeT Chen et al., 2023 and ArchCode Han et al., 2024) votes over execution behaviour instead of program text. We run it on the identical fixed LCBv5 pools (167 problems × ten configurations × the same 64 candidates; 1,670 cells). Per problem, K_{test}=16 test inputs are generated once

Table 5: **Selection agreement between RAD and DeepConf** (math+GPQA, n=3,114). Rollout level: same rollout; answer level: same normalized answer class.

Pair	Level	Observed	Chance	κ_{rand}
RAD vs. DeepConf-argmax	rollout	1.25%	1.81%	−0.006
RAD vs. DeepConf-argmax	answer	80.86%	75.75%	0.211
RAD vs. DeepConf-vote	answer	93.32%	75.75%	0.725
DeepConf-vote vs. Majority	answer	99.00%	75.75%	0.959

from the public statement (the generator sees no candidate code, hidden tests, outputs, or labels); candidates that fail to execute on all inputs are discarded; FMV abstains only if none remain (116 cells, scored incorrect on the full denominator). FMV+RAD has identical coverage: it applies RAD only among candidates sharing FMV's winning execution vector and never falls back to standalone RAD.

Table 6: **Execution-based voting vs. RAD on LiveCodeBench v5** (pass@1 %, 1,670 cells; FMV-covered subset 1,554). RAD uses no execution.

Selector	Coverage	Full (1,670)	FMV-covered (1,554)
Avg@64 (random)	100%	61.34	—
RAD	100%	66.71	68.08
FMV (K_{test}=16)	93.05%	66.89	71.88
FMV+RAD	93.05%	**68.56**	**73.68**

On covered cells FMV exceeds RAD by +3.80 pp (130 vs. 71 discordant wins; exact McNemar p=3.8 × 10^{-5}); within FMV's execution-equivalence set RAD improves FMV by +1.80 pp (67 vs. 39; p=0.0084), +1.68 pp on the full denominator, and the full-denominator gain stays positive for $K_{\text{test}} \in \{8, 16, 32\}$ (+3.23/ + 1.68/ + 0.84 pp). FMV needs generated tests and sandboxed execution, which open-ended and agentic settings may not afford; RAD uses routing already produced during inference. The two are orthogonal: FMV identifies a consensus execution behaviour, RAD ranks the candidates that share it.

CoT-Decoding. CoT-Decoding (Wang & Zhou, 2024) is a verifier-free decoding-time selector: after a fixed analysis-channel prefix it branches on the top-10 candidates of the first reasoning token, continues each branch greedily, and returns the standard greedy branch (CoT-greedy) or the branch with the largest mean top-1/top-2 probability gap in the answer region (CoT-max). We run it on gpt-oss-20B-Medium on AIME24, AIME25, and HMMT25 (90 problems). To match the candidate count, RAD@10 draws 10 of the 64 full rollouts uniformly 10,000 times per problem, computes per-problem accuracy, and averages over problems (problem bootstrap 95% CI).

Table 7: **CoT-Decoding vs. RAD at ten candidates** (accuracy %, gpt-oss-20B-Medium, 30 problems per set). RAD@10 is a uniform 10-of-64 subsample, not a diversity-optimized construction, which favours CoT-Decoding.

Method	AIME24	AIME25	HMMT25	Overall
CoT-greedy@10	73.33	60.00	40.00	57.78
CoT-max@10	76.67	70.00	63.33	70.00
RAD@10 (mean)	82.35	78.44	62.11	**74.30**
95% CI	[69.76, 92.99]	[66.14, 89.86]	[47.30, 75.88]	[66.64, 81.76]

A.5 MoE ROUTING CAPTURE

Capture point in the forward pass. Routing tensors are obtained from an internally MoE-capture-adapted vLLM v0.10.0 serving runtime with the `enable_return_routed_experts` and `enable_return_routed_experts_weights` flags. A `RoutedExpertsCapturer` singleton intercepts the output of each MoE layer's `router.select_experts(...)` call:

this returns the top-K expert ids and post-softmax weights *after* the gate's softmax-then-top-K-renormalize step, so the recorded weights satisfy $\sum_k w_{i,\ell,t,k} \approx 1$ per token, layer, and rollout. Captured tensors are buffered on the GPU and copied to CPU staging buffers at request boundaries; we do not modify model weights, KV caches, or any compute path.

Storage schema. Per shard, captured routing for all P problems and all $N{=}64$ rollouts is written under `token_data_*/` as four files: `moe_meta.json` (schema version, L, K, total tokens, dtype dictionary), `moe_row_ptr.int64` (CSR row pointer mapping each run id to a token-index range), `moe_topk_expert_ids.int16.b2` (flat $[T_{\text{shard}} \cdot L \cdot K]$, int16, blosc2 ZSTD + SHUFFLE + BYTEDELTA at compression level 5, 4 MB chunks), and `moe_topk_expert_weights.float16.b2` (same shape, float16, same compression). A parallel pair of files stores prompt-side routing. Compression yields $\sim$1.8$\times$ savings.

Token alignment. We strictly read token identities from the binary `tok_token_ids.int32.b2` column rather than re-tokenizing the generated text. Re-tokenization through `tokenizer.encode(generated_text)` loses up to 7 trailing special-control tokens on OSS Harmony decoding (because the served text passes through `skip_special_tokens`), which causes a fixed off-by-Δ shift between text-derived token indices and the MoE routing tensor's index. All anchor-locating logic in this paper operates on `tok_token_ids`, so the post-marker window in RAD is exactly aligned with the routing tensor index.

Models and architectural diversity. The 10 model rollout pools span three structurally distinct sparse MoE designs (Table 8). **gpt-oss** (20B and 120B evaluated under Low/Medium/High prompt-level reasoning effort; Appendix A.5) uses a Transformer MoE with top-4 routing and alternating full / locally-banded sparse attention plus grouped multi-query attention; the 20B variant has a small expert pool ($E{=}32$, $\sim$12.5% activation rate per token) while the 120B variant has $E{=}128$ at $\sim$3.1%. **Qwen3-30B-A3B** (Instruct/Thinking) is a standard Transformer MoE with $E{=}128$ and a relatively wide top-8 ($\sim$6.25%) plus standard GQA attention. **Qwen3-Next-80B-A3B** (Instruct/Thinking) is a hybrid Gated DeltaNet / Gated Attention backbone with an ultra-sparse MoE: $E{=}512$ routed experts at top-10 ($\sim$1.95%) plus one shared expert that contributes a constant per-token mass on top of the routed selection.

GPT-OSS reasoning-effort prompt protocol. The gpt-oss Low/Medium/High conditions are not separate model checkpoints and do not modify model weights, MoE router parameters, expert pools, KV-cache behavior, quantization, temperature, or top-p. They are prompt-level conditions expressed in OpenAI's Harmony chat format. OpenAI's gpt-oss model card states that the models were trained to support three reasoning levels, *low*, *medium*, and *high*, configured in the system prompt by inserting keywords such as `Reasoning: low`; increasing the reasoning level increases the model's average reasoning trace length (OpenAI, 2025). The Harmony documentation likewise places the reasoning-effort field in the system message, with `medium` as the default when no level is specified (Kundel, 2025).

Concretely, the OSS shards in this paper differ only in the following Harmony system-message line:

```
Reasoning: low
```

versus

```
Reasoning: medium
```

versus

```
Reasoning: high
```

All downstream capture code, routing hooks, sampling hyperparameters, and answer extraction logic are otherwise shared across the three conditions.

A minimal prompt-level example is:

```
<|start|>system<|message|>You are ChatGPT, a large language model
↪  trained by OpenAI.
Knowledge cutoff: 2024-06
Current date: 2025-06-28

Reasoning: low

# Valid channels: analysis, commentary, final. Channel must be
↪  included for every message.<|end|>
<|start|>developer<|message|># Instructions
Solve the problem. Put the final answer in \boxed{}.<|end|>
<|start|>user<|message|>What is 17*23? Put the final answer in
↪  \boxed{}.<|end|>
<|start|>assistant
```

The Medium and High versions replace only the `Reasoning: low` line with `Reasoning: medium` or `Reasoning: high`. Harmony uses special control tokens such as `<|start|>`, `<|message|>`, and `<|end|>` to delimit messages, and `<|return|>` to terminate a completed assistant response. Assistant messages are further tagged by channels, including `<|channel|>analysis` for the model's internal reasoning stream and `<|channel|>final` for the user-facing answer (Kundel, 2025). In our OSS same-token analysis, the boundary anchor `<|channel|>final<|message|>` is therefore a Harmony protocol boundary, not an answer-string feature. This is why Exp. I can hold the measured token ids fixed at the final-channel boundary triplet while the experts that produce it still differ across Low, Medium, and High: the prompt-level reasoning-effort directive changes the hidden state entering the router before the identical protocol boundary is emitted.

Implication for the routing representation. The three regimes differ in how "thick" a per-token route vector is: gpt-oss-20B's small pool causes higher route-id collisions across rollouts, Qwen3-30B-A3B sits at the moderate sweet spot, and Qwen3-Next's ultra-sparse routing requires careful handling of the shared expert (we exclude it from the WJ histogram so it does not act as a constant noise floor that artificially inflates pairwise similarity). Despite these architectural differences, the same WJ-KNN agreement rule (Eq. 5) drives RAD on all 10 pools.

Table 8: **MoE architectural diversity in our pool.** The three families differ in expert pool size, top-k routing density, attention block, and the presence of a shared expert; RAD's WJ-KNN agreement rule applies to all of them without modification.

Family	Total / Active	Layers	Experts	Top-k	Attention
gpt-oss-20B	21B / 3.6B	24	32	4	alternating full / banded + GQA
gpt-oss-120B	117B / 5.1B	36	128	4	alternating full / banded + GQA
Qwen3-30B-A3B	30.5B / 3.3B	48	128	8	GQA (32 Q / 4 KV)
Qwen3-Next-80B-A3B	80B / 3B	48	512 + 1 shared	10 + shared	Gated DeltaNet + Gated Attention

All weights are stored at float16 precision regardless of the model's inference quantization (e.g., mxfp4 on gpt-oss).

Capture cost. After the layer-id-cache and `clear_buffer`-skip optimizations in the patched serving runtime, the routing capture path adds $\sim$0.05 ms per generated token ($\sim$32 s for a 6×10^5 token batch), about a 7% overhead relative to baseline generation on our hardware. Memory-mapped reads keep per-shard runtime memory under 8 GB. Total disk usage for the 310,400-run, 4,850-cell pool is approximately 1.2 TB, dominated by routing weights (the ids column compresses $\sim2\times$ better than the weights column). The 1.2 TB is the full-token offline audit cache, not RAD's deployment requirement: storing only the default 16-token anchor windows of the anchor-found rollouts takes 4.49 GB raw and an estimated 2.08–2.55 GB compressed ($\approx$470–580$\times$ smaller, $\approx$0.5 MB per problem), and at serving time RAD holds only the 64 anchor-window vectors of the current cohort, discardable after selection.

Selection cost. RAD reduces each problem to a single Weighted-Jaccard K-NN density argmax over N=64 completed rollouts. With the sparse top-K representation, each pairwise WJ is $O(L \cdot s)$ ($L \leq 48$ MoE layers; $s \leq \min(E, WK)$ the per-layer support after window-averaging, a small

constant at the deployed $W{=}16$, larger on SWE-bench where it unions over boundary positions), so per-problem cost is dominated by the 64×64 similarity matrix and runs in tens of milliseconds on a single CPU thread; the full 4,850-cell analysis pass completes in well under a minute end-to-end.

A.6 K-NN BASIN PURITY PROTOCOL

This appendix documents the exact computation of $A_{K_{\mathrm{nn}},r}$ (Eq. 2).

Cohort. Per problem p we collect $N{=}64$ rollouts. A rollout is included iff (i) its boxed or code-output answer can be extracted and dataset-aware normalized, and (ii) the routing readouts required by the analysis are present: for the window-readout diagnostic (Fig. 2), valid delimiter-anchored windows; for the round-indexed trajectory (Fig. 5), at least one occurrence of the relevant trajectory-anchor token strictly before the delimiter marker. The effective neighbor count is $k_{\mathrm{eff}} = \min(K_{\mathrm{nn}}, N_{\mathrm{valid}} - 1)$; we use $K_{\mathrm{nn}}{=}1$ for the window-readout basin-purity diagnostic (Fig. 2) and the purity–accuracy trajectory plot, and $K_{\mathrm{nn}}{=}8$ for the round-indexed anchor trajectory (Fig. 5). Problems with fewer than two valid rollouts or fewer than two distinct normalized answers are excluded; single-basin problems ($n_a{=}N$ for some a) yield $b{=}1$, so $A_{K_{\mathrm{nn}},r}$ is undefined and the problem is excluded. The $N{=}1{,}180$ basin-purity cells (aime25 138, brumo25 129, gpqa 913) are not fully independent (each problem is shared across the eight models) so the per-bar bootstrap (resampling cells) mildly underestimates the CI; a problem-clustered bootstrap widens the whiskers by $\sim 5\%$ and leaves the window ordering unchanged.

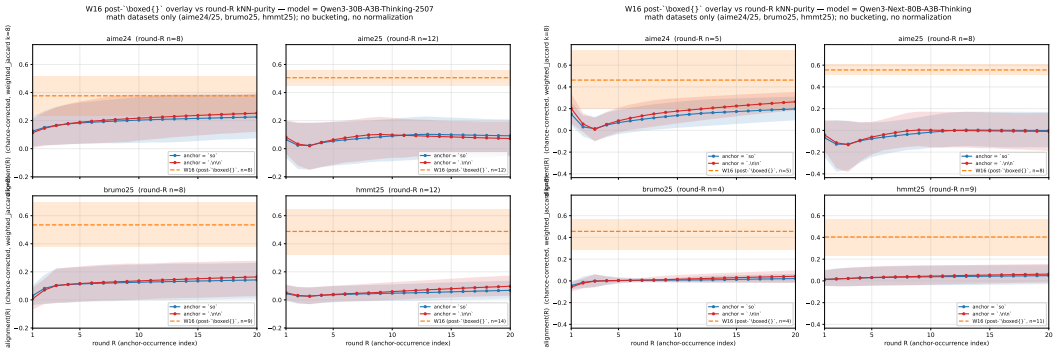
Routing feature. The per-rollout routing vector at anchor round r is the $L \cdot E$-dimensional sparse expert-weight histogram described in §3.1, accumulated over the readout window and divided by the window's token count, i.e. the per-token mean of the routed expert mass. This per-token averaging matters when rollouts in the *same* graph have different valid window lengths (from multi-token anchors, fallback, or truncation at the rollout's end) so a rollout with a shorter valid window is not penalized for carrying less total mass. (For a fixed window length it does not change the neighbor ranking, since Weighted Jaccard is invariant to a common rescaling of *both* vectors, $\mathrm{WJ}(cu, cv){=}\mathrm{WJ}(u, v)$, and each readout's K-NN graph is built separately.) We use Weighted Jaccard throughout (for the selector, the round-indexed analyses, and the window-readout basin-purity diagnostic), having empirically verified that Weighted Jaccard and cosine produce qualitatively identical neighborhood rankings on this representation.

Basin label. The primary label space is the normalized final-answer string under dataset-aware normalization (AIME $\rightarrow$ integer; GPQA $\rightarrow$ first uppercase letter; LiveCodeBench $\rightarrow$ raw string after canonicalization, which is near-unique across rollouts, usable for basin grouping but degenerate as a majority vote, which is why exact-string Majority is shown as "–" on code). Two diagnostic alternatives are reported in supplementary tables: binary correctness ($\{0, 1\}$) and first-character of the normalized answer (coarsest, AIME-only).

Anchor-round trajectory (deferred from §3.3). Figure 5 traces $A_{K_{\mathrm{nn}},r}$ at successive trajectory-anchor occurrences and overlays the `delimiter@16` RAD window line, exposing the early-steep / middle-plateau / delimiter-jump three-regime structure that the main-text Figure 2 only summarises in aggregate.

$A_{K_{\mathrm{nn}}}$–accuracy trajectory plot. Figure 6 presents the same per-t data as the main-text Figure 2 (right panel), reprojected from the dual-axis-vs-t view into the (purity, accuracy) diagnostic plane: each scatter point is one per-rollout token budget $16t$, $t \in [4, 64]$ (the rollout count is fixed at $N{=}64$), the viridis color encodes t, and the connecting curve traces the joint trajectory. The northeast walk of the trajectory is the alignment-vs-accuracy relationship made explicit: a unit increase in routing-neighbour basin purity corresponds to a measurable increase in selector accuracy on the same problems, consistent with purity and accuracy moving together as t grows (this view does not separate the two from their shared dependence on t). Saturation lives at the upper-right end of the curve at $t \approx 32$–64.

Ground-truth vs. majority pseudo-label. The three-regime curve in Figure 5 uses the textual majority-vote answer as a label-free pseudo-basin so that $A_{K_{\mathrm{nn}},r}$ does not require ground-truth access.



(a) Qwen3-30B-A3B-Thinking-2507. (b) Qwen3-Next-80B-A3B-Thinking.

Figure 5: **Anchor-round trajectory of $A_{K_{\mathrm{nn}},r}$ (K_{nn}=8).** Lines per *trajectory*-anchor surface form (shown in the in-panel legend under the heading "anchor"): `so` in blue, `.\n\n` in red. The horizontal orange line (printed in-panel with the earlier label "post-\boxed") is the `delimiter@16` RAD window. Shaded bands 95% bootstrap CIs (n_{boot}=1000). Two Qwen Thinking checkpoints; four math datasets per subfigure (AIME24, AIME25, BRUMO25, HMMT25). The early steep rise, middle stable plateau, and delimiter-window jump are visible across both models and all four datasets.

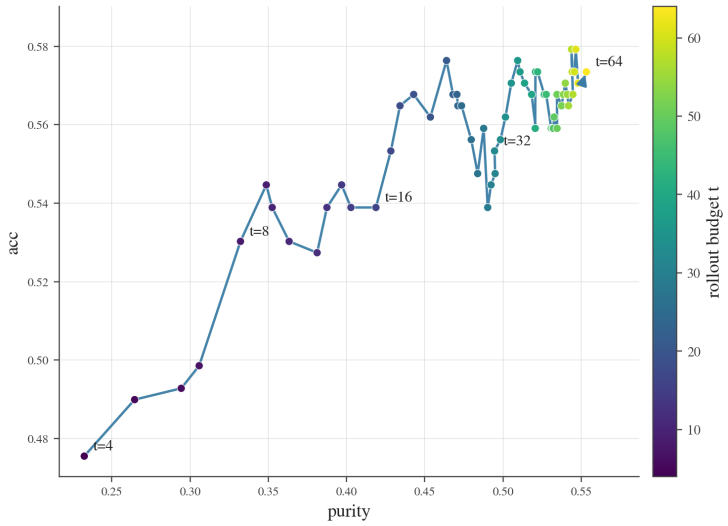


Figure 6: **Purity–accuracy trajectory parameterised by t.** Same balanced-panel data as Figure 2 right (347 math problems, 8 MoE models, K_{nn}=1 purity, KNN density-vote selector). Each scatter point is one $t \in \{4, \ldots, 64\}$; viridis color encodes t from dark purple (t=4) to bright yellow (t=64); arrow shows direction of increasing t; anchor labels at $t \in \{4, 8, 16, 32, 64\}$. The curve walks northeast: each unit gain in $A_{K_{\mathrm{nn}}}$ corresponds to a roughly co-linear gain in selector accuracy.

We separately verify that using the held-out ground-truth answer as basin label yields qualitatively identical curves: the early-steep / middle-plateau / delimiter-jump shape and the relative gap between regimes are preserved within bootstrap CI on every model-dataset cell we checked. Majority-labelled $A_{K_{\mathrm{nn}},r}$ is therefore a faithful proxy for ground-truth-labelled $A_{K_{\mathrm{nn}},r}$ for the diagnostic claim of §3.3.

Chance baseline forms. The implementation uses the leave-one-out form $b_{\mathrm{LOO}} = \sum_a n_a(n_a - 1)/\big(N(N-1)\big)$ printed in Eq. 2 for both the window-readout basin-purity diagnostic and the round-indexed analysis. The plug-in alternative $b_{\mathrm{PI}} = \sum_a (n_a/N)^2$ differs by $O(1/N)$ and yields qualitatively identical chance-corrected purity at N=64; we report the leave-one-out form throughout.

Cross-problem aggregation. Per-problem chance-corrected purity is $A_p = (\bar{p}_p - b_p)/(1 - b_p)$. The dataset-level value is the arithmetic mean across problems; a 95% confidence interval is reported via problem-level percentile bootstrap with n_{boot}=2000 resamples, resampling problems with replacement and reporting the 2.5$^{\text{th}}$–97.5$^{\text{th}}$ percentile of bootstrap means. For paired comparisons across two anchor rounds, the same bootstrap resamples the per-problem pair jointly and percentiles the resampled differences.

A.7 Method details and fallbacks

Anchor extraction. Trajectory, boundary, and delimiter anchors are located by binary `tok_token_ids` matching against a per-model token-id family (Appendix A.5). For each anchor, the family is built once per tokenizer by scanning the vocabulary and collecting every token id whose decoded surface form satisfies a regex defining the anchor: e.g., `so` uses `(?i)(^|[^A-Za-z])so($|[^A-Za-z])` so that `'so'`, `'So'`, and `'\nSo'` all count as the same anchor occurrence regardless of how the BPE tokenizer happened to split surrounding context. The trajectory-anchor extractor matches the discourse-marker families `.\n\n`, `so`, `now`, `wait`, `.`, and `:`; the reported analyses use a *single* such anchor per readout, by default `so` (with `now`, and the paragraph break `.\n\n` in Fig. 5, as the alternatives compared in App. A.9), never combined. Delimiter anchors are dataset-specific: `\boxed{` for math (AIME / BRUMO / HMMT) and GPQA, and the literal three-backtick code fence for LiveCodeBench. Boundary anchors use final-response transition markers such as the OSS Harmony triplet `<|channel|>final<|message|>` or end-of-thinking markers when present.

Delimiter-anchor location, eligibility, and baselines. For each rollout the controller locates the answer-opening delimiter, in order: (i) `\boxed{` on math/GPQA; (ii) the dataset-specific code-fence on LCB; (iii) the OSS boundary triplet on OSS rollouts. RAD scores a rollout only when its anchor (and the 16-token window) is located; per problem this gives three cases: **(a)** all N located (≈75% of well-posed problems), the candidate set is the full pool; **(b)** a strict subset located ($n_{\text{cohort}} < N$, ≈24%), the anchor-missing rollouts have no routing readout and are dropped from the candidate set, RAD selecting among the located ones; **(c)** fewer than two located (∼0.8%), RAD abstains and the problem is scored *incorrect*. Mean anchor coverage is ≈93% (per-model 0.73–1.00). *Baselines are handled the same way:* textual Majority and the DeepConf-vote act only on rollouts with a parseable answer (a rollout with none casts no vote), while the Avg@N floor counts all N unconditionally. RAD, Majority, and DeepConf are therefore scored on the *same per-problem denominator* (cases (a)–(c) included), so RAD's anchor requirement is comparable to answer extraction, not an extra restriction. Restricting to anchor-located rollouts is a mild self-selection toward rollouts that reached the answer; it contributes the mechanical Δ_{anchor} part of RAD's lift over Avg@64, with the genuine selection effect isolated as Δ_{sel} (per-model in Table 20).

EOS-aligned fallback. Commit markers are not required once generation has completed. On the same fixed 64-rollout pools (six datasets, 4,850 cells), random selection obtains 64.61%, marker-based RAD 71.32% (+6.71 pp), an EOS-aligned readout using each rollout's final 16 routing rows 70.37% (+5.76 pp), and the marker-with-EOS-fallback rule 70.54% (+5.93 pp): EOS preserves ≈86% of the marker-based gain. Pre-completion selection without a phase marker or EOS is outside the validated scope.

Answer extraction, normalization, and tie handling. Final answers for Majority and DeepConf-vote are extracted and normalized with the open-source `math_verify` toolkit (Kydlíček et al., 2025) and SymPy (equivalent expressions merged before counting and before comparison with the ground truth; GPQA: first uppercase option letter; LiveCodeBench: canonicalized raw program strings, near-unique across rollouts). Majority counts rollouts per normalized class; DeepConf-vote follows the open-source DeepConf implementation. Ties break to the earliest class in the fixed rollout order; an instance with no valid answer is marked incorrect and stays in the denominator. RAD never reads these strings (App. A.1); they only score its selection after the fact.

Weighted Jaccard computation. $\text{WJ}(u, v) = \sum_e \min(u_e, v_e) / \sum_e \max(u_e, v_e)$ is computed on the sparse $L \cdot E$-dimensional histogram representation; only experts in the union of the two rollouts'

(window-averaged) supports contribute, so the per-pair cost is $O(L \cdot s)$, s the per-layer sparse support size, $s \leq \min(E, WK)$, rather than $O(L \cdot E)$.

A.8 Exact random baselines

Setup. For each shard, P problems are sampled and each problem has $n_{\text{runs}} = 64$ rollouts. Let k_p be the number of correct rollouts on problem p and $\hat{p}_p = k_p/n_{\text{runs}}$ its empirical correct rate. The random single-run selector picks one rollout uniformly at random per problem; we report its expected pass@1 $\mu_{\text{rand}} = (1/P)\sum_p \hat{p}_p$, its 95% upper envelope $U_{95}^{\text{rand-1}}$, and the lift $\Delta = \text{Acc}_{\text{method}} - U_{95}^{\text{rand-1}}$.

Selector-variance Wald CI. Treating the corpus $\{\hat{p}_p\}_{p=1}^{P}$ as fixed and the random selector as the only source of randomness, the per-problem outcomes $X_p \sim \text{Bern}(\hat{p}_p)$ are independent and the random pass@1 has variance

$$\text{Var}\left[\frac{1}{P}\sum_p X_p\right] \;=\; \frac{1}{P^2}\sum_p \hat{p}_p(1-\hat{p}_p), \tag{6}$$

which yields the Wald 95% confidence interval

$$\text{CI}_{95} \;=\; \mu_{\text{rand}} \pm 1.96 \cdot \sqrt{\tfrac{1}{P^2}\sum_p \hat{p}_p(1-\hat{p}_p)}. \tag{7}$$

We use the upper bound $U_{95}^{\text{rand-1}}$ as the comparison line in the main text. This CI answers a fixed-corpus question ("how wide is the random selector's distribution on *these* problems?") rather than a cross-corpus question ("does the gap generalize to a new problem set?"); for the latter we additionally report problem-level paired bootstrap intervals on selected Δ values, but these are not the canonical envelope.

A.9 Cost–certainty frontier and the routing window

This appendix gives the full analysis behind the cost–certainty frontier of §5.3. We score a problem's rollouts by their local density in a k-NN graph (K=10) over the weighted-Jaccard similarity of the per-token-mean expert-activation histogram read in a short window at an anchor token, and return the densest rollout's correctness (`dens`). *Trajectory* anchors are the recurring discourse markers `So` (default) and `Now`, each read as a single anchor (read early, near-universal; the recency-optimized variant `So`* uses the latest in-budget `So`); *Boundary* anchors are end-of-thinking or final-channel markers, while *delimiter* anchors are fixed answer-opening delimiters such as `\boxed{}` and code fences; both are read late and usually once. Per problem we compare `dens` against `avg@coh` (mean correctness over the anchor-found cohort) and the anchor-free `avg@full` ($= n_{\text{correct}}/64$ over all rollouts), giving $\Delta_{\text{anchor}} =$ `avg@coh` $-$ `avg@full` (mechanical, kernel-independent), $\Delta_{\text{sel}} =$ `dens` $-$ `avg@coh` (pure selector intelligence), and $\Delta_{\text{full}} = \Delta_{\text{anchor}} + \Delta_{\text{sel}}$. Significance is a Poisson-binomial exact test on the per-problem baseline rates (reported as z, one-sided p). We use the anchor-free `avg@full`; a deceptively common fallback (`avg64` $\times\, n_{\text{graded}}/64$) zeroes out correct-but-anchor-absent rollouts and inflates the lift ≈3.3× at low coverage, which we avoid.

The frontier. Table 9 gives the per-budget Δ_{full} and coverage for all five anchors. Trajectory anchors are usable from a small budget (coverage ≈ 1 throughout); boundary/delimiter anchors cover few problems until high budget but then dominate on accuracy, the two crossing at $B \approx$ 2048–4096. The families also differ in *kind*: trajectory anchors carry $\Delta_{\text{anchor}} \approx 0$ (the markers are ubiquitous, so reaching one says nothing about correctness), so their lift is entirely selector intelligence, whereas boundary/delimiter anchors accrue Δ_{anchor} up to +3–5 pp because reaching the end-of-thinking point is itself a correctness signal, roughly half of their headline lift.

A lexical control: the discourse token `So`. Aligning rollouts at the last in-budget occurrence of `So` with the whole readout window inside the budget B, selection on identical anchor-found units is near random for $B \leq$ 1,024, reaches 66.49% vs. 64.92% random at B=2,048, and 68.36% vs. 64.92% at full budget (coverage 67.3% → 99.2%); the post-commit readout reaches 71.3% on the same six-dataset pool. Context-dependent routing is present at ordinary tokens and becomes substantially clearer near commitment.

Table 9: **Cost–certainty frontier (weighted Jaccard):** uncertainty reduction Δ_{full} in pp (PB-exact z in parentheses) vs. per-rollout token budget B, for the five anchors. * marks $|z| \geq 1.96$ (one-sided $p < 0.025$). cov is the late-anchor coverage (fraction of problems with the boundary/delimiter anchor in budget); trajectory coverage is ≈ 1 from low B. Low-B boundary/delimiter rows sit on a tiny short-trace subset (selection bias; flagged, not interpreted).

B	trajectory So	trajectory So*	trajectory Now	boundary	delimiter	cov(late)
256	-0.38 (-0.8)	$+1.58$ (2.9)*	-1.27 (-1.5)	-1.17 (-1.1)	-1.06 (-1.0)	0.19
1024	$+0.70$ (1.5)	$+0.71$ (1.4)	$+0.06$ (0.1)	$+0.45$ (0.7)	$+0.87$ (1.4)	0.45
4096	$+1.71$ (3.7)*	$+1.26$ (2.6)*	$+0.39$ (0.8)	$+2.35$ (4.4)*	$+3.69$ (7.0)*	0.68
16384	$+2.77$ (6.1)*	$+2.45$ (5.1)*	$+2.37$ (4.6)*	$+4.26$ (9.0)*	$+5.83$ (12.4)*	0.92
32768	$+3.45$ (7.5)*	$+3.55$ (7.4)*	$+3.98$ (7.7)*	$+6.11$ (13.2)*	$+7.65$ (16.5)*	0.98

How much to read: the budget-honest window. Holding the budget fixed, the routing window has a budget-honest optimum (Table 10, Figure 7): the whole window must fit in $[0, B)$ or the rollout is dropped. A small window ($W \approx 8$–16) is best at mid budgets, and only the largest budget favours the widest fitting window ($W{=}512$, $+4.06$ pp, $z{=}8.5$); the per-budget argmax is noisy between. A naive sweep that lets the window read past B spuriously favours a mid window, a budget leak this rule removes. The deployed RAD reads a 16-token window (§4.1), which sits in this flat near-optimal region at typical budgets.

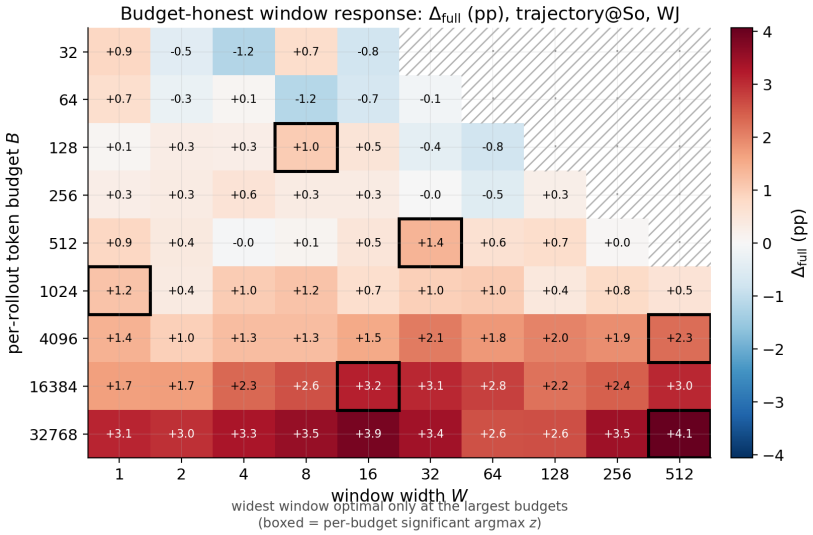


Figure 7: **Budget-honest window grid (trajectory So, weighted Jaccard).** Δ_{full} (pp) for each (budget B, window width W); hatched cells cannot fit the budget. Boxed cells are the per-budget significant argmax ($z \geq 2$). A small window is optimal at mid budgets; only the largest budget favours $W{=}512$.

Table 10: **Budget-honest window grid (trajectory So, weighted Jaccard):** Δ_{full} in pp for each (budget B, window width W); "–" marks windows that cannot fit the budget (rollout dropped).

$B\backslash W$	1	2	4	8	16	32	64	128	256	512
32	$+0.9$	-0.5	-1.2	$+0.7$	-0.8	–	–	–	–	–
64	$+0.7$	-0.3	$+0.1$	-1.2	-0.7	-0.1	–	–	–	–
128	$+0.1$	$+0.3$	$+0.3$	$+1.0$	$+0.5$	-0.4	-0.8	–	–	–
256	$+0.3$	$+0.3$	$+0.6$	$+0.3$	$+0.3$	-0.0	-0.5	$+0.3$	–	–
512	$+0.9$	$+0.4$	-0.0	$+0.1$	$+0.5$	$+1.4$	$+0.6$	$+0.7$	$+0.0$	–
1024	$+1.2$	$+0.4$	$+1.0$	$+1.2$	$+0.7$	$+1.0$	$+1.0$	$+0.4$	$+0.8$	$+0.5$
4096	$+1.4$	$+1.0$	$+1.3$	$+1.3$	$+1.5$	$+2.1$	$+1.8$	$+2.0$	$+1.9$	$+2.3$
16384	$+1.7$	$+1.7$	$+2.3$	$+2.6$	$+3.2$	$+3.1$	$+2.8$	$+2.2$	$+2.4$	$+3.0$
32768	$+3.1$	$+3.0$	$+3.3$	$+3.5$	$+3.9$	$+3.4$	$+2.6$	$+2.6$	$+3.5$	$+4.1$

Width versus recency. The window helps through two separable mechanisms (Table 11, Figure 8). On a common subset, fixing the anchor and widening the window alone (*width*) lifts Δ_{full} from $+1.81$ to $+4.06$ pp, the dominant driver. Using the latest in-budget anchor adds a *recency* bonus that is largest at small W ($+1.29$ at $W{=}1$) and vanishes by $W{=}512$: small windows ride recency, wide windows ride width.

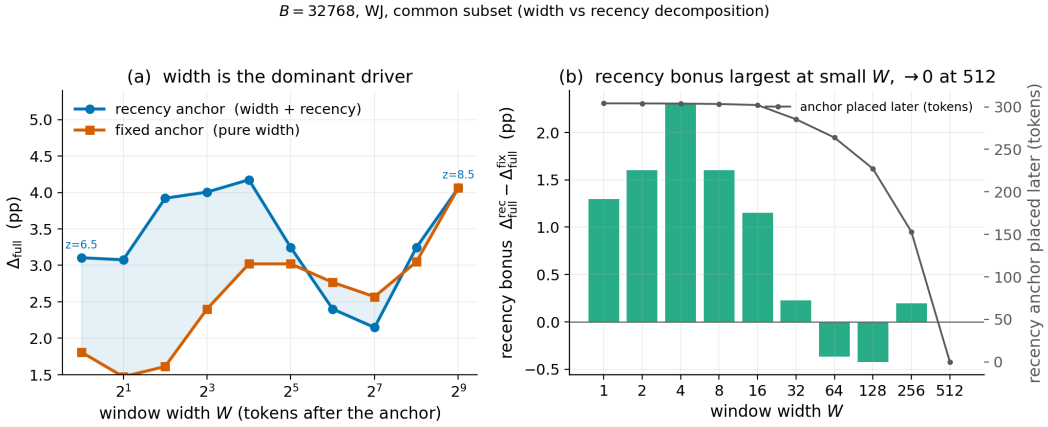


Figure 8: **Width vs. recency (trajectory `So`, $B{=}32768$, weighted Jaccard).** *Left:* Δ_{full} vs. window width for the `fixed` (pure-width) and `recency` (latest-anchor) strategies on a common subset. *Right:* the recency bonus (`recency−fixed`) and the mean anchor depth per window. Width is the dominant driver ($+1.81 \rightarrow +4.06$ pp); the recency bonus is largest at small W and vanishes by $W{=}512$.

Table 11: **Width vs. recency (trajectory `So`, $B{=}32768$, weighted Jaccard, common subset):** Δ_{full} in pp (z). `fixed` holds the anchor and varies only the window (pure width); `recency` uses the latest in-budget anchor per window; `bonus=recency−fixed`.

W	fixed (width)	recency	bonus
1	+1.81 (3.8)	+3.10 (6.5)	+1.29
2	+1.47 (3.1)	+3.08 (6.4)	+1.60
4	+1.61 (3.4)	+3.92 (8.2)	+2.31
8	+2.40 (5.0)	+4.01 (8.3)	+1.60
16	+3.02 (6.3)	+4.17 (8.7)	+1.15
32	+3.02 (6.3)	+3.25 (6.8)	+0.23
64	+2.77 (5.8)	+2.40 (5.0)	−0.37
128	+2.57 (5.4)	+2.15 (4.5)	−0.42
256	+3.05 (6.4)	+3.25 (6.8)	+0.20
512	+4.06 (8.5)	+4.06 (8.5)	+0.00

Per-model and coverage caveats. The trajectory-anchor lift is positive on all 8 models (6/8 significant at $B{=}32768$) with $\Delta_{\text{anchor}} \approx 0$ throughout, whereas boundary/delimiter Δ_{anchor} tracks reasoning effort / trace length (e.g. OSS-20B-High $+14.3$ pp of its $+16.3$, OSS-120B-High $+6.8$ of $+8.2$), a property of *whether the rollout finishes*, not of the selector. At low budget the boundary/delimiter cells sit on a tiny short-trace, low-effort subset (coverage 1–19%); those rows are selection-bias artifacts and are flagged, not interpreted. The kernel choice is within noise at the standard $W{=}16$ (Appendix A.13).

Domain dependence. Selector intelligence is strongly domain-dependent (Table 12): it orders math > science > code for the trajectory and boundary/delimiter anchors, and on code the trajectory (`So`*) selector is null ($\Delta_{\text{sel}} = +1.00$ pp, $z{=}1.1$). The $+11$–12 pp lift sometimes quoted for routing selectors exists only for boundary/delimiter×math at high budget, and a large share of it is the mechanical Δ_{anchor}, not selector quality.

Table 12: **Per-domain selector intelligence at B=32768 (weighted Jaccard):** $\Delta_{\text{sel}} = \texttt{dens} - \texttt{avg@coh}$ (pure selector) and $\Delta_{\text{full}} = \Delta_{\text{anchor}} + \Delta_{\text{sel}}$ in pp (z). The trajectory ($\texttt{So}^*$) selector is *null* on code; the +11–12 pp headline lives only in boundary/delimiter×math, of which the mechanical Δ_{anchor} is a large share.

domain	$\texttt{So}^*$ Δ_{sel}	$\texttt{So}^*$ Δ_{full}	boundary Δ_{sel}	boundary Δ_{full}	boundary Δ_{anchor}
math	+6.88 (7.7)*	+6.95 (7.7)*	+6.17 (7.4)*	+10.99 (12.0)*	+4.82
science	+2.95 (4.1)*	+3.09 (4.2)*	+2.05 (2.8)*	+3.74 (5.2)*	+1.69
code	+1.00 (1.1)	+1.10 (1.2)	+1.77 (2.3)*	+5.46 (6.8)*	+3.69

A.10 Agentic best-of-N: SWE-bench Verified

This appendix details the agentic best-of-N result reported with Table 2.

Setup. For each of 500 SWE-bench Verified problems a coding agent samples N=16 independent rollouts (8,000 rollouts per model); each rollout emits a code patch whose correctness $y_i \in \{0, 1\}$ is decided by the hidden test suite. The selector must pick one rollout *without* executing tests, reading the reference patch, or inspecting y_i. It uses only the rollout's MoE routing, exactly as in the main pool. We anchor at the *boundary* token marking the thinking→action transition: `</think>` for Qwen3.6-35B-A3B (ChatML thinking template) and the action-turn opener `<|start|>` for the gpt-oss Harmony template (the multi-turn agentic action boundary). Both denote the same event (the switch from internal reasoning to emitting the patch) and occur exactly once per reasoning step (full coverage). For each rollout we take the union, over its boundary positions, of the activated (layer, expert) routing pairs into a binary fingerprint S_i, score each rollout by its mean binary Jaccard to its k=4 nearest neighbours, and select the densest center. The three backbones span distinct MoE widths (gpt-oss architectures in Appendix A.5; Qwen3.6-35B-A3B is a separate configuration with 40 layers × 256 routed experts). The baseline is random rollout selection (*Avg@16*), whose expectation equals the per-problem mean pass rate with zero selector variance, matching the Avg@N floor used for the main pool (Appendix A.8).

Per-model agentic harness and configuration. Each backbone is run inside the multi-step coding-agent harness it is in-distribution for, rather than a single shared scaffold, so the routing recurs at a genuine action boundary in each model's native trajectory format. All three sample N=16 independent rollouts on the 500-problem **SWE-bench Verified** (Jimenez et al., 2024; OpenAI, 2024) split. *Qwen3.6-35B-A3B* runs under a **mini-swe-agent**-style loop (Yang et al., 2024): a linear message history in which the model alternates a `THOUGHT` block and a single `ACTION bash` command per step and terminates by issuing `submit`. Sampling is temperature 0.6, top-p 0.95, up to 50 steps, with a 60 s per-command execution timeout in a per-instance Docker environment; the candidate patch is recovered as `git diff` against the task base commit. The thinking→action boundary is the ChatML `</think>` token that closes each step's reasoning span. *gpt-oss 20B and 120B* run under **harmonyagent** (Mavrin, 2026), a from-scratch reconstruction of an in-distribution gpt-oss agent in the OpenAI **Harmony** response format (Kundel, 2025), with a Harmony tool set (`apply_patch`, `repo_browser.{print_tree,search,open_file}`, `container.exec`) and the analysis/final channels. Both sizes are run at a single fixed `Reasoning: high` reasoning-effort setting with a 131,072-token context window and Harmony-default sampling (temperature 1.0, top-p 1.0); unlike the main pool, which sweeps gpt-oss Low/Medium/High as separate configurations (Appendix A.5), the agentic SWE-bench runs use only High for both models. Reasoning and tool calls are emitted in the `analysis` channel and each agentic action turn opens with the Harmony `<|start|>` control token, which is the boundary anchor. The candidate patch is recovered as `git add -A && git diff -cached`. Because the two harnesses extract patches under different conventions (base-commit `diff` vs. staged `diff -cached`, with different truncation and cleanup), *absolute* resolve rates are not strictly comparable across the Qwen and gpt-oss families. This does not affect RAD: selection is always made *within* one model's 16-rollout cohort under a single fixed harness, where all candidates share the same extraction convention, so each model's reported lift over its own Avg@16 floor is invariant to the cross-family extraction mismatch.

Table 13: **The boundary-fingerprint length confound is inert at the selector level** (SWE-bench, N=16 rollouts; pooled correlations over all rollouts, within-problem correlation and lifts on the decidable subset, in pp over Avg@16). Per-rollout step counts vary widely (e.g. Qwen3.6 1–166 thinking→action boundaries) and fingerprint cardinality $|S_i|$ scales with them. On Qwen3.6 $|S_i|$ is pooled-anticorrelated with correctness (-0.23), but this is an across-problem difficulty effect (harder problems run longer and fail more): it *vanishes within problem* ($|r| \leq 0.06$ on all three), and the within-problem cohort is the only set the k-NN ranks. Accordingly the best step-count-only selector gives no positive lift, whereas RAD lifts $+4.7/+4.8/+5.0$ pp. The agentic gain is routing agreement, not trajectory length.

Model	$r(\|S_i\|, \text{step})$	$r(\|S_i\|, \text{correct})$		step-count	RAD
	(pooled)	pooled	within-prob.	selector	lift
Qwen3.6-35B-A3B	$+0.60$	-0.23	-0.00	-3.9	$+\mathbf{4.7}$
gpt-oss-120B	$+0.23$	$+0.01$	$+0.03$	$+0.8$	$+\mathbf{4.8}$
gpt-oss-20B	$+0.15$	$+0.04$	$+0.06$	$+0.5$	$+\mathbf{5.0}$

Metrics. *Mixed sel@1* is accuracy on the *decidable* subset: problems whose 16 rollouts are split between passing and failing, the only problems on which a selector can change the outcome ($n_{\text{mix}} = 233/247/286$ for Qwen3.6-35B-A3B / gpt-oss-120B / gpt-oss-20B). *Avg@all* is end-to-end accuracy over all 500 problems. Because roughly half of all problems are unanimous (all rollouts pass or all fail) and are therefore inert under any selector, the end-to-end gain is close to half the decidable-subset gain: the $+4.7/+4.8/+5.0$ pp on mixed sel@1 dilute to $+2.2/+2.4/+2.9$ pp on Avg@all (Table 2).

Cross-architecture portability. The identical recipe (boundary anchor, pair-level binary Jaccard, k=4 density) yields $+4$ to $+5$ pp on the decidable subset of all three models, with no per-model tuning and no access to answer strings. The three backbones differ substantially in routing width and chat template, so the agreement signal at the action boundary is a portable property of MoE reasoning rather than an artefact of one architecture.

Why a binary fingerprint. We record only *which* (layer, expert) pairs fire at the boundary, not how often, so binarizing to set membership drops the per-pair *count* component (which grows with trajectory length) and keeps the portable "which experts were recruited" signal. The union over a rollout's per-step boundaries still leaves the fingerprint cardinality $|S_i|$ scaling with step count, but this length confound is inert at the selector level: *within* each problem's rollout cohort (the only set the k-NN ranks), $|S_i|$ is uncorrelated with correctness and a step-count-only selector gives no lift, so RAD's $+4.7/+4.8/+5.0$ pp is routing agreement, not trajectory length (Table 13).

Why a single anchor. The signal is concentrated at the single action-start boundary. Pooling routing rows from the boundary together with other token positions mixes distinct decision phases and dilutes the agreement signal; reading each reasoning step at its lone boundary token keeps the comparison phase-aligned. This is the agentic counterpart of the marker-only readouts analysed for the main pool in §5.3.

A.11 Cohort agreement, reasoning effort, and thinking vs. instruct

This appendix gives the full analysis behind the cohort-agreement result of §5.3, using the lift decomposition of Appendix A.9 (anchor-free `avg@full` $= n_{\text{correct}}/64$; `avg@coh` the anchor-found cohort mean; $\Delta_{\text{anchor}} =$ `avg@coh` $-$ `avg@full`, $\Delta_{\text{sel}} =$ `dens` $-$ `avg@coh` the selector intelligence vs. a random within-cohort pick, $\Delta_{\text{full}} = \Delta_{\text{anchor}} + \Delta_{\text{sel}}$). The pool spans ten models (gpt-oss 20B/120B at Low/Medium/High reasoning effort and Qwen3-30B-A3B / Qwen3-Next-80B-A3B in Instruct and Thinking variants) over the math, GPQA, and well-posed (math+GPQA) pools, at the deployed setting (delimiter anchor, 16-token window, K=10, weighted Jaccard; the kernel is a null axis). Significance is the Poisson-binomial exact macro test ($\Delta_{\text{sel}}/\Delta_{\text{full}}$, the random-within-cohort-pick null), exact McNemar (paired contrasts), and a per-size paired bootstrap (effort trend); $^{*}/^{**}/^{***}$ denote $p < 0.05/0.01/0.001$.

Cohort agreement (the law). The selector's gain concentrates on problems whose rollouts disagree. The cross-model correlation between Δ_{sel} and the cohort baseline avg@coh is $r = -0.90$ (Figure 4), but this is *largely structural*: dens and avg@coh are 99% collinear ($r = 0.993$), so Δ_{sel} is a residual bounded by the $1 - \texttt{avg@coh}$ headroom and a constant-headroom-fraction selector already yields slope ≈ -1. The more direct test not relying on the cross-model headroom is at the problem level, within model: a problem's selector advantage (RAD-correct minus its cohort pass rate) rises with its answer entropy, simple correlation $+0.19$ ($p = 1.3 \times 10^{-27}$, $n = 3152$) and *partial* correlation $+0.28$ ($p = 2.9 \times 10^{-59}$) after partialling out difficulty (Table 14), which cannot be a ceiling/part-whole artifact. The cross-model answer-entropy twin ($r = +0.84$) is shown directly in Figure 9.

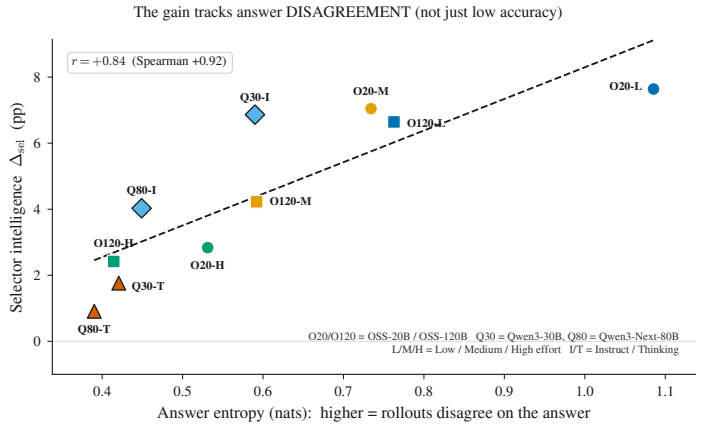


Figure 9: **The selector's gain tracks answer *disagreement*, not just low accuracy.** Per-model selector intelligence Δ_{sel} vs. full-pool answer entropy (well-posed; weighted Jaccard); cross-model $r = +0.84$. The difficulty-controlled problem-level test is Table 14.

Table 14: **Problem-level: the selector helps more on higher-disagreement problems (weighted Jaccard).** Within each model, per-problem selector advantage (RAD-correct − cohort pass rate) vs. answer entropy (well-posed, Pearson). Pooled = within-model demeaned; the last row additionally partials out per-problem difficulty (cohort pass rate). The positive partial correlation shows that the entropy–advantage association is not explained away by cohort pass rate.

Model	n	corr(entropy, advantage)
OSS-120B-High	316	+0.11
OSS-120B-Low	318	+0.21
OSS-120B-Medium	318	+0.16
OSS-20B-High	304	+0.18
OSS-20B-Low	318	+0.14
OSS-20B-Medium	318	+0.31
Qwen3-30B-A3B-Instruct-2507	318	+0.31
Qwen3-30B-A3B-Thinking-2507	316	+0.15
Qwen3-Next-80B-A3B-Instruct	318	+0.18
Qwen3-Next-80B-A3B-Thinking	308	+0.09
Pooled (within-model demeaned)	3152	**+0.19** (p=1.3×10^{-27})
+ difficulty partialled out	3152	**+0.28** (p=2.9×10^{-59})

Reasoning effort: a ceiling effect, not lost skill. As OSS effort rises Low→Medium→High the cohort converges (avg@coh 42 → 70 → 92%), so the absolute Δ_{sel} falls $+12.2 \to +11.2 \to +3.6$ pp (Table 15; paired-bootstrap Low−High $+9.3$ pp $[+2.8, +15.6]$ $p = 0.003$ for 20B, $+9.2$ $[+4.0, +14.3]$ $p < 0.001$ for 120B, Table 16). Normalised by headroom the selector captures an *equal-or-larger* fraction at higher effort (21 → 39 → 43%), so this is a ceiling effect, not lost skill. The operational lift Δ_{full} stays $\sim +12$–15 pp at every effort, but its source flips from genuine selection (Low: $\Delta_{\text{anchor}} \approx 0$) to the mechanical late-anchor filter (High: $\Delta_{\text{anchor}} \approx +11$, as high-effort runs

less reliably emit an in-window `\boxed`). RAD matches or edges textual majority at every effort (dens−maj@64 $= +0.8/0/+3.1$ pp).

Table 15: **RAD lift vs. OSS reasoning effort** (math, 20B+120B pooled, weighted Jaccard). Δ_{anchor} = late-anchor self-selection; Δ_{sel} = genuine selector intelligence (PB-exact); $\Delta_{\text{full}}=\Delta_{\text{anchor}}+\Delta_{\text{sel}}$. As effort rises the cohort converges (avg@coh 42→92%): the absolute Δ_{sel} shrinks but per-headroom it is preserved, and the lift's source flips from selection to the mechanical anchor filter.

Effort	avg@64	Cons@64	Δ_{anc}	Δ_{sel}	Δ_{full}	Δ_{sel}/HR	RAD	RAD+DC
Low	41.9	53.3	+0.1	$+12.2^{***}$	+12.2	21.0%	54.2	54.2
Medium	69.9	82.5	+1.4	$+11.2^{***}$	+12.6	39.1%	82.5	84.2
High	80.6	92.1	+11.0	$+3.6^{**}$	+14.6	43.1%	95.2	93.9

Table 16: **Effort trend, per-size paired bootstrap (5000 resamples, shared problems).** diff = lift(first) − lift(second); 95% CI; sign-based two-sided p. The trend is concentrated at Medium→High; Low→Medium is n.s.

Method	Size	Contrast	n	Δ_{full} diff [CI] (p)	Δ_{sel} diff [CI] (p)
math					
RAD	OSS-20B	Low-High	111	-3.6 [-12.1,+4.8] (0.387)	+9.3 [+2.8,+15.6] (0.003)
RAD+DC	OSS-20B	Low-High	111	-0.0 [-8.7,+8.3] (0.989)	+12.9 [+6.2,+19.3] (0.000)
RAD	OSS-20B	Medium-High	111	-0.1 [-5.5,+4.9] (0.996)	+10.4 [+5.3,+15.5] (0.000)
RAD+DC	OSS-20B	Medium-High	111	+1.7 [-4.4,+7.4] (0.558)	+12.2 [+6.4,+18.0] (0.000)
RAD	OSS-20B	Low-Medium	120	-3.6 [-11.0,+3.7] (0.347)	-1.0 [-8.0,+6.0] (0.798)
RAD+DC	OSS-20B	Low-Medium	120	-3.6 [-11.3,+4.2] (0.385)	-1.0 [-8.4,+6.3] (0.809)
RAD	OSS-120B	Low-High	118	+0.1 [-7.0,+6.9] (0.982)	+9.2 [+4.0,+14.3] (0.000)
RAD+DC	OSS-120B	Low-High	118	-0.7 [-8.1,+6.4] (0.833)	+8.3 [+2.8,+13.8] (0.003)
RAD	OSS-120B	Medium-High	118	-2.7 [-8.5,+2.8] (0.354)	+6.3 [+1.6,+10.9] (0.007)
RAD+DC	OSS-120B	Medium-High	118	-0.2 [-6.1,+5.1] (0.951)	+8.8 [+3.7,+14.0] (0.000)
RAD	OSS-120B	Low-Medium	120	+2.8 [-3.2,+8.7] (0.371)	+2.9 [-3.2,+8.8] (0.363)
RAD+DC	OSS-120B	Low-Medium	120	-0.5 [-7.5,+6.2] (0.882)	-0.5 [-7.4,+6.3] (0.897)
well-posed					
RAD	OSS-20B	Low-High	305	-4.2 [-8.9,+0.2] (0.064)	+5.0 [+1.1,+8.7] (0.012)
RAD+DC	OSS-20B	Low-High	305	-2.9 [-7.5,+1.4] (0.194)	+6.3 [+2.3,+10.1] (0.002)
RAD	OSS-20B	Medium-High	305	-3.4 [-7.0,+0.0] (0.052)	+4.5 [+1.3,+7.7] (0.008)
RAD+DC	OSS-20B	Medium-High	305	-3.1 [-6.8,+0.5] (0.094)	+4.8 [+1.4,+8.3] (0.008)
RAD	OSS-20B	Low-Medium	318	-0.7 [-4.9,+3.2] (0.702)	+0.6 [-3.4,+4.5] (0.792)
RAD+DC	OSS-20B	Low-Medium	318	-0.1 [-4.4,+4.0] (0.949)	+1.2 [-2.8,+5.2] (0.558)
RAD	OSS-120B	Low-High	316	-1.2 [-5.1,+2.4] (0.517)	+4.3 [+1.2,+7.4] (0.007)
RAD+DC	OSS-120B	Low-High	316	-2.5 [-6.2,+1.1] (0.183)	+3.0 [-0.1,+6.2] (0.059)
RAD	OSS-120B	Medium-High	316	-3.6 [-6.9,-0.5] (0.019)	+1.9 [-0.9,+4.6] (0.177)
RAD+DC	OSS-120B	Medium-High	316	-3.0 [-6.3,+0.1] (0.062)	+2.5 [-0.4,+5.3] (0.092)
RAD	OSS-120B	Low-Medium	318	+2.4 [-1.1,+6.0] (0.170)	+2.4 [-1.1,+6.0] (0.167)
RAD+DC	OSS-120B	Low-Medium	318	+0.5 [-3.3,+4.2] (0.818)	+0.5 [-3.3,+4.2] (0.808)

Thinking vs. instruct. The same mechanism explains the thinking/instruct gap. On the selectorable *math* set both variants show genuine selector intelligence, but 2–3× larger on instruct (Table 17, Figure 10): Qwen3-30B Instruct $+12.3^{***}$ vs. Thinking $+4.5^{*}$; Qwen3-Next-80B Instruct $+7.1^{**}$ vs. Thinking $+2.2$ (n.s.; RAD+DC $+4.9^{***}$), because thinking cohorts converge (avg@coh 87–92%) while instruct cohorts stay diverse (61–69%). On GPQA the selector is near-null for all four ($\Delta_{\text{sel}} = +0.1$ to $+3.6$, n.s.); since GPQA is ≈63% of the well-posed problems, the well-posed problem-weighted average flips Qwen3-30B-Thinking from a significant math effect ($+4.5^{*}$) to a misleading n.s. ($+1.8$), so we report *math* for this contrast. The higher *pick* accuracy on thinking models (+14–17 pp) is base capability, not selector quality.

DeepConf and fusion. Adding DeepConf confidence weighting (RAD+DC) does not significantly change RAD in any well-posed or math cell after Holm correction (paired McNemar, Table 18; pooled-by-effort $p = 1.00/0.71/1.00$; the one nominally significant cell, Next-80B-Thinking well-posed $+1.9$ pp $p = 0.031$, does not survive Holm); its small help concentrates where routing density saturates (converged thinking cohorts), and under WJ no cell is significantly harmful. The two fusion variants (conf50, the deployed top-50% cBW filter then density, vs. rank-sum) are themselves indistinguishable (Table 19), so the fusion choice is not a design axis.

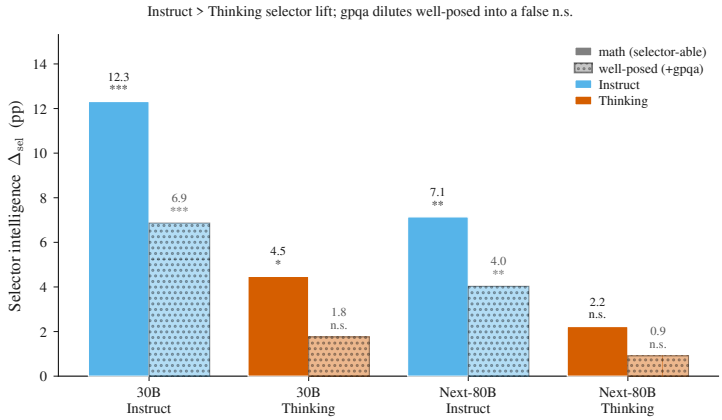


Figure 10: **Instruct > thinking selector lift, and GPQA dilutes it.** Δ_{sel} per Qwen model on math vs. well-posed (math+GPQA, weighted Jaccard). The selector is genuine on both variants but 2–3× larger on instruct (diverse cohorts); GPQA is near-null for all, so the well-posed problem-weighted average flips Qwen3-30B-Thinking from significant ($+4.5^{*}$) to a false n.s.

Table 17: **Thinking vs. instruct: the selector's intelligence is larger on instruct than thinking models, and GPQA dilutes it.** Δ_{sel} (PB-exact). On math, instruct > thinking; GPQA is near-null for all, so the well-posed problem-weighted average flips Qwen3-30B-Thinking to a misleading n.s.

	avg@64 (math)	avg@coh (math)	Δ_{sel} (pp)			RAD dens
Model			math	gpqa	well-posed	(math)
Qwen3-30B Instruct	60.9	61.0	$+12.3^{***}$	+3.6	$+6.9^{***}$	73.3
Qwen3-30B Thinking	80.0	87.1	$+4.5^{*}$	+0.1	+1.8	91.5
Qwen3-Next-80B Instruct	68.6	68.7	$+7.1^{**}$	+2.1	$+4.0^{**}$	75.8
Qwen3-Next-80B Thinking	83.6	91.6	+2.2	+0.1	+0.9	93.9

Table 18: **RAD+DC vs. RAD, paired exact McNemar (weighted Jaccard).** Δ = RAD+DC − RAD (pp); *rad only*/*dc only* are the discordant counts. No well-posed/math cell is significant after Holm correction (one cell nominally p=0.031); DC only nudges narrow high-convergence cells.

Group	n	RAD	RAD+DC	Δ	rad only	dc only	McNemar p
math							
OSS-20B-Low	120	48.3	49.2	+0.8	3	4	1.000
OSS-20B-Medium	120	83.3	84.2	+0.8	2	3	1.000
OSS-20B-High	111	94.6	91.9	-2.7	3	0	0.250
OSS-120B-Low	120	60.0	59.2	-0.8	4	3	1.000
OSS-120B-Medium	120	81.7	84.2	+2.5	2	5	0.453
OSS-120B-High	118	95.8	95.8	+0.0	1	1	1.000
OSS-pooled-Low	240	54.2	54.2	+0.0	7	7	1.000
OSS-pooled-Medium	240	82.5	84.2	+1.7	4	8	0.388
OSS-pooled-High	229	95.2	93.9	-1.3	4	1	0.375
30B-Thinking	118	91.5	91.5	+0.0	1	1	1.000
30B-Instruct	120	73.3	74.2	+0.8	0	1	1.000
Next-80B-Thinking	114	93.9	96.5	+2.6	0	3	0.250
Next-80B-Instruct	120	75.8	77.5	+1.7	1	3	0.625
gpqa							
OSS-20B-Low	198	60.6	61.6	+1.0	4	6	0.754
OSS-20B-Medium	198	67.7	67.7	+0.0	5	5	1.000
OSS-20B-High	194	69.1	70.1	+1.0	6	8	0.791
OSS-120B-Low	198	69.2	67.7	-1.5	5	2	0.453
OSS-120B-Medium	198	72.7	72.2	-0.5	4	3	1.000
OSS-120B-High	198	81.3	81.3	+0.0	2	2	1.000
OSS-pooled-Low	396	64.9	64.6	-0.3	9	8	1.000
OSS-pooled-Medium	396	70.2	69.9	-0.3	9	8	1.000
OSS-pooled-High	392	75.3	75.8	+0.5	8	10	0.815
30B-Thinking	198	71.2	72.2	+1.0	2	4	0.688

Group	n	RAD	RAD+DC	Δ	rad only	dc only	McNemar p
30B-Instruct	198	70.2	68.2	-2.0	7	3	0.344
Next-80B-Thinking	194	77.8	79.4	+1.5	0	3	0.250
Next-80B-Instruct	198	74.7	75.3	+0.5	2	3	1.000
well-posed							
OSS-20B-Low	318	56.0	56.9	+0.9	7	10	0.629
OSS-20B-Medium	318	73.6	73.9	+0.3	7	8	1.000
OSS-20B-High	305	78.4	78.0	-0.3	9	8	1.000
OSS-120B-Low	318	65.7	64.5	-1.3	9	5	0.424
OSS-120B-Medium	318	76.1	76.7	+0.6	6	8	0.791
OSS-120B-High	316	86.7	86.7	+0.0	3	3	1.000
OSS-pooled-Low	636	60.8	60.7	-0.2	16	15	1.000
OSS-pooled-Medium	636	74.8	75.3	+0.5	13	16	0.711
OSS-pooled-High	621	82.6	82.4	-0.2	12	11	1.000
30B-Thinking	316	78.8	79.4	+0.6	3	5	0.727
30B-Instruct	318	71.4	70.4	-0.9	7	4	0.549
Next-80B-Thinking	308	83.8	85.7	+1.9	0	6	0.031
Next-80B-Instruct	318	75.2	76.1	+0.9	3	6	0.508
code							
OSS-20B-Low	167	38.9	41.9	+3.0	11	16	0.442
OSS-20B-Medium	167	72.5	73.7	+1.2	6	8	0.791
OSS-20B-High	139	84.2	82.0	-2.2	9	6	0.607
OSS-120B-Low	167	70.1	67.7	-2.4	7	3	0.344
OSS-120B-Medium	167	79.0	78.4	-0.6	4	3	1.000
OSS-120B-High	155	88.4	89.7	+1.3	1	3	0.625
OSS-pooled-Low	334	54.5	54.8	+0.3	18	19	1.000
OSS-pooled-Medium	334	75.7	76.0	+0.3	10	11	1.000
OSS-pooled-High	294	86.4	86.1	-0.3	10	9	1.000
30B-Thinking	162	74.7	77.2	+2.5	4	8	0.388
30B-Instruct	167	45.5	44.9	-0.6	5	4	1.000
Next-80B-Thinking	167	70.7	72.5	+1.8	4	7	0.549
Next-80B-Instruct	167	63.5	65.9	+2.4	2	6	0.289

Table 19: **DeepConf fusion robustness: top-50% cBW filter (conf50) vs. rank-sum (weighted Jaccard).** conf50 keeps the top-50% of rollouts by DeepConf cBW, then picks the densest by routing K-NN; Δ = rankfuse $-$ conf50 (pp), paired exact McNemar. The two fusions agree within noise everywhere, so the fusion choice is not a design axis.

Group	n	conf50	rankfuse	Δ	c50 only	rf only	McNemar p
math							
OSS-20B-Low	120	49.2	48.3	-0.8	3	2	1.000
OSS-20B-Medium	120	84.2	83.3	-0.8	6	5	1.000
OSS-20B-High	111	91.9	93.7	+1.8	0	2	0.500
OSS-120B-Low	120	59.2	61.7	+2.5	1	4	0.375
OSS-120B-Medium	120	84.2	84.2	+0.0	2	2	1.000
OSS-120B-High	118	95.8	94.9	-0.8	2	1	1.000
OSS-pooled-Low	240	54.2	55.0	+0.8	4	6	0.754
OSS-pooled-Medium	240	84.2	83.8	-0.4	8	7	1.000
OSS-pooled-High	229	93.9	94.3	+0.4	2	3	1.000
30B-Thinking	118	91.5	91.5	+0.0	0	0	1.000
30B-Instruct	120	74.2	72.5	-1.7	2	0	0.500
Next-80B-Thinking	114	96.5	93.9	-2.6	3	0	0.250
Next-80B-Instruct	120	77.5	77.5	+0.0	1	1	1.000
gpqa							
OSS-20B-Low	198	61.6	59.6	-2.0	9	5	0.424
OSS-20B-Medium	198	67.7	67.2	-0.5	5	4	1.000
OSS-20B-High	194	70.1	69.6	-0.5	5	4	1.000
OSS-120B-Low	198	67.7	69.7	+2.0	1	5	0.219
OSS-120B-Medium	198	72.2	72.7	+0.5	4	5	1.000
OSS-120B-High	198	81.3	81.8	+0.5	1	2	1.000
OSS-pooled-Low	396	64.6	64.6	+0.0	10	10	1.000
OSS-pooled-Medium	396	69.9	69.9	+0.0	9	9	1.000
OSS-pooled-High	392	75.8	75.8	+0.0	6	6	1.000
30B-Thinking	198	72.2	72.2	+0.0	3	3	1.000
30B-Instruct	198	68.2	68.7	+0.5	3	4	1.000
Next-80B-Thinking	194	79.4	78.4	-1.0	2	0	0.500
Next-80B-Instruct	198	75.3	74.7	-0.5	2	1	1.000
well-posed							
OSS-20B-Low	318	56.9	55.3	-1.6	12	7	0.359
OSS-20B-Medium	318	73.9	73.3	-0.6	11	9	0.824
OSS-20B-High	305	78.0	78.4	+0.3	5	6	1.000
OSS-120B-Low	318	64.5	66.7	+2.2	2	9	0.065
OSS-120B-Medium	318	76.7	77.0	+0.3	6	7	1.000

Group	n	conf50	rankfuse	Δ	c50 only	rf only	McNemar p
OSS-120B-High	316	86.7	86.7	+0.0	3	3	1.000
OSS-pooled-Low	636	60.7	61.0	+0.3	14	16	0.856
OSS-pooled-Medium	636	75.3	75.2	-0.2	17	16	1.000
OSS-pooled-High	621	82.4	82.6	+0.2	8	9	1.000
30B-Thinking	316	79.4	79.4	+0.0	3	3	1.000
30B-Instruct	318	70.4	70.1	-0.3	5	4	1.000
Next-80B-Thinking	308	85.7	84.1	-1.6	5	0	0.062
Next-80B-Instruct	318	76.1	75.8	-0.3	3	2	1.000
code							
OSS-20B-Low	167	41.9	41.9	+0.0	21	21	1.000
OSS-20B-Medium	167	73.7	74.3	+0.6	7	8	1.000
OSS-20B-High	139	82.0	81.3	-0.7	8	7	1.000
OSS-120B-Low	167	67.7	67.1	-0.6	5	4	1.000
OSS-120B-Medium	167	78.4	80.2	+1.8	2	5	0.453
OSS-120B-High	155	89.7	91.0	+1.3	2	4	0.688
OSS-pooled-Low	334	54.8	54.5	-0.3	26	25	1.000
OSS-pooled-Medium	334	76.0	77.2	+1.2	9	13	0.523
OSS-pooled-High	294	86.1	86.4	+0.3	10	11	1.000
30B-Thinking	162	77.2	77.8	+0.6	4	5	1.000
30B-Instruct	167	44.9	47.3	+2.4	6	10	0.454
Next-80B-Thinking	167	72.5	73.1	+0.6	9	10	1.000
Next-80B-Instruct	167	65.9	66.5	+0.6	9	10	1.000

Per-model decomposition. Table 20 gives the full decomposition for all ten models across the four pools. On code (LiveCodeBench) graded rollouts are a subset of anchor-found ones, so Δ_{anchor} is large and mechanical, a format/completion-compliance signal (emitting a code fence marks a complete, more-often-correct rollout, a correctness proxy rather than answer content), and string-voting baselines degenerate; only the routing density Δ_{sel} is a genuine selector measure there. Even so, against the Avg@64 floor reported in Table 1 RAD's code selection is positive for every model family; Δ_{sel} is the stricter test (its baseline is the within-cohort random pick `avg@coh`, which already contains the mechanical Δ_{anchor}) so a small or occasionally negative Δ_{sel} measures selection *on top of* an already-elevated cohort, not a loss against the reported floor.

Table 20: **Per-model full decomposition (RAD, weighted Jaccard).** All pools; n is the *cohort-eligible* count (problems with ≥ 2 anchor-located rollouts), a subset of the full sets reported by the headline Tables 1/3: these n sum to 3153 of the 3180 well-posed and 1625 of the 1670 code problems (headline accuracies use the full sets; this decomposition uses the cohort-eligible subset). Δ_{sel} PB-exact. On code, graded $\subseteq$ anchor-found so maj degenerates ($-$) and only Δ_{sel} is meaningful.

Model	n	avg@full	avg@coh	maj	dens	Δ_{anc}	Δ_{sel}	Δ_{full}	Δ_{sel}/HR
math									
OSS-120B-High	118	84.4	93.4	95.8	95.8	+9.1	+2.3	+11.4	35.3%
OSS-120B-Low	120	48.8	48.8	58.3	60.0	+0.0	+11.2***	+11.2	21.9%
OSS-120B-Medium	120	73.3	73.3	84.2	81.7	+0.1	+8.3**	+8.4	31.3%
OSS-20B-High	111	76.5	89.6	88.3	94.6	+13.1	+5.0**	+18.1	48.2%
OSS-20B-Low	120	35.1	35.2	48.3	48.3	+0.1	+13.1***	+13.3	20.3%
OSS-20B-Medium	120	66.5	69.2	80.8	83.3	+2.7	+14.2***	+16.9	45.9%
Qwen3-30B-A3B-Instruct-2507	120	60.9	61.0	70.8	73.3	+0.1	+12.3***	+12.4	31.6%
Qwen3-30B-A3B-Thinking-2507	118	80.0	87.1	88.1	91.5	+7.0	+4.5*	+11.5	34.5%
Qwen3-Next-80B-A3B-Instruct	120	68.6	68.7	74.2	75.8	+0.0	+7.1**	+7.2	22.8%
Qwen3-Next-80B-A3B-Thinking	114	83.6	91.6	90.4	93.9	+8.0	+2.2	+10.2	26.6%
gpqa									
OSS-120B-High	198	75.4	78.8	83.3	81.3	+3.4	+2.5	+5.9	11.7%
OSS-120B-Low	198	65.3	65.3	65.7	69.2	+0.0	+3.9	+3.9	11.2%
OSS-120B-Medium	198	71.0	71.0	73.2	72.7	+0.0	+1.7	+1.7	6.0%
OSS-20B-High	194	60.5	67.5	72.2	69.1	+7.0	+1.6	+8.6	4.9%
OSS-20B-Low	198	56.3	56.3	61.6	60.6	+0.0	+4.3	+4.3	9.9%
OSS-20B-Medium	198	64.4	64.9	68.2	67.7	+0.6	+2.7	+3.3	7.8%
Qwen3-30B-A3B-Instruct-2507	198	66.2	66.6	69.2	70.2	+0.5	+3.6	+4.0	10.7%
Qwen3-30B-A3B-Thinking-2507	198	71.1	71.1	71.2	71.2	+0.0	+0.1	+0.1	0.5%
Qwen3-Next-80B-A3B-Instruct	198	72.2	72.6	75.3	74.7	+0.4	+2.1	+2.5	7.8%
Qwen3-Next-80B-A3B-Thinking	194	62.0	77.7	79.9	77.8	+15.7	+0.1	+15.8	0.6%
well-posed									
OSS-120B-High	316	78.7	84.3	88.0	86.7	+5.6	+2.4	+8.0	15.4%
OSS-120B-Low	318	59.1	59.1	62.9	65.7	+0.0	+6.6***	+6.6	16.2%
OSS-120B-Medium	318	71.9	71.9	77.4	76.1	+0.0	+4.2**	+4.3	15.0%
OSS-20B-High	305	66.3	75.5	78.0	78.4	+9.2	+2.8*	+12.0	11.6%
OSS-20B-Low	318	48.3	48.3	56.6	56.0	+0.1	+7.6***	+7.7	14.8%

Model	n	avg@full	avg@coh	maj	dens	Δ_{anc}	Δ_{sel}	Δ_{full}	Δ_{sel}/HR
OSS-20B-Medium	318	65.2	66.5	73.0	73.6	+1.4	+7.0***	+8.4	21.1%
Qwen3-30B-A3B-Instruct-2507	318	64.2	64.5	69.8	71.4	+0.3	+6.9***	+7.2	19.3%
Qwen3-30B-A3B-Thinking-2507	316	74.4	77.0	77.5	78.8	+2.6	+1.8	+4.4	7.7%
Qwen3-Next-80B-A3B-Instruct	318	70.9	71.1	74.8	75.2	+0.3	+4.0**	+4.3	14.0%
Qwen3-Next-80B-A3B-Thinking	308	70.0	82.9	83.8	83.8	+12.8	+0.9	+13.7	5.3%
code									
OSS-120B-High	155	76.9	85.5	–	88.4	+8.7	+2.9	+11.5	19.7%
OSS-120B-Low	167	64.3	64.3	–	70.1	+0.0	+5.7**	+5.7	16.1%
OSS-120B-Medium	167	76.1	76.2	–	79.0	+0.1	+2.9	+3.0	12.1%
OSS-20B-High	139	61.7	77.3	–	84.2	+15.6	+6.9*	+22.4	30.3%
OSS-20B-Low	167	33.5	33.8	–	38.9	+0.3	+5.1	+5.4	7.7%
OSS-20B-Medium	167	67.3	68.3	–	72.5	+1.0	+4.1	+5.1	13.0%
Qwen3-30B-A3B-Instruct-2507	167	47.1	47.1	–	45.5	+0.0	-1.6	-1.5	-3.0%
Qwen3-30B-A3B-Thinking-2507	162	71.4	73.8	–	74.7	+2.3	+0.9	+3.3	3.6%
Qwen3-Next-80B-A3B-Instruct	167	62.8	62.9	–	63.5	+0.0	+0.6	+0.6	1.6%
Qwen3-Next-80B-A3B-Thinking	167	70.1	71.2	–	70.7	+1.1	-0.5	+0.5	-1.8%

A.12 Sensitivity to the neighborhood size k_{rad} and window width W

The deployed selector uses k_{rad}=10, W=16 with no per-dataset tuning. Table 21 sweeps k_{rad} at W=16: overall accuracy moves by 0.6 pp (70.7–71.3) and the per-task optima differ. The full grid $k_{\text{rad}} \in \{2, 4, 6, 8, 10, 12, 16, 24\} \times W \in \{4, 8, 16, 32\}$ (32 settings) stays within 70.4–71.5%; at W=16 varying k_{rad} changes accuracy by 0.6 pp, at k_{rad}=10 varying W by 0.7 pp, and the default (71.3%) is 0.2 pp below the best setting (trajectory-anchor window analysis: App. A.9).

Table 21: k_{rad} **sweep at** W=16 (accuracy %, ten configurations; Math = four math sets, 1,200 cells; LCBv5 1,670; GPQA 1,980; Overall = problem-weighted over all 4,850 cells, i.e. the six-dataset pool, not Table 1's math+GPQA average).

k_{rad}	Math	LCBv5	GPQA	Overall
2	77.2	65.6	71.1	70.7
4	77.8	66.0	71.4	71.1
6	77.6	66.0	71.2	71.0
8	78.0	66.2	71.3	71.2
10 (default)	**78.3**	**66.5**	71.2	**71.3**
12	78.0	66.5	71.1	71.2
16	77.8	66.0	70.9	70.9
24	78.0	65.9	70.8	70.9

Reading routing before the answer. Using only the routing of the first 32, 64, 128, or 256 generated tokens (same pool and selector) gives 64.8, 65.2, 65.0, and 65.1% against 71.3% for the post-commit readout: the answer basin is not reliably recoverable early.

Density as a confidence score. On GPQA, local routing density ranks rollouts usefully (AUROC 0.743) but is poorly calibrated (ECE 0.203; 0.078 after per-question Platt scaling); at the option level route mass is nearly redundant with the softmax over options and combining them adds nothing, so we do not claim independent uncertainty information.

A.13 Binary vs. weighted Jaccard kernel

RAD scores agreement with a k-NN density in routing space; the similarity kernel can be either binary Jaccard (JA, set overlap of activated experts) or weighted Jaccard (WJ, $\sum_e \min(u_e, v_e) / \sum_e \max(u_e, v_e)$ of routing weights). We test whether the routing *weights* matter with a within-cell paired ablation that holds the anchor, cohort, baselines, and all 15,188 problem cells (8 thinking models × 6 datasets, cohort ≥ 2) fixed and varies *only* the kernel between two otherwise-identical runs. The two kernels pick a different rollout on just 816 cells (5.4%); only these carry kernel signal.

The difference is non-significant in every anchor mode and all Holm-adjusted p-values are 1.000 (Table 22); the pooled gap is WJ−JA = +0.22 pp with a shard cluster-bootstrap 95% CI $[-0.18, +0.59]$

Table 22: **Binary (JA) vs. weighted (WJ) Jaccard kernel, per anchor mode.** Within-cell paired ablation over 15,188 problem cells (8 thinking models × 6 datasets, cohort ≥ 2); only the kNN similarity kernel differs between the two runs. *b*/*c* are the discordant picks (JA-only-correct / WJ-only-correct); the exact two-sided McNemar test acts on them. All four anchor modes are non-significant and Holm-adjust to 1.000; pooled WJ−JA = +0.22 pp (95% CI [−0.18, +0.59], cluster bootstrap over shards). *No detectable difference under the primary paired test.*

Anchor mode	n	JA (%)	WJ (%)	WJ−JA	b	c	McNemar p	Holm p
`boundary@marker`	3786	70.71	71.16	+0.45	103	120	0.284	1.000
`delimiter@marker`	3808	71.43	71.85	+0.42	109	125	0.327	1.000
`delimiter@16`	3808	73.58	73.82	+0.24	72	81	0.518	1.000
`boundary@16`	3786	72.82	72.61	−0.21	107	99	0.626	1.000
Pooled	**15188**	**72.14**	**72.36**	+0.22	**391**	**425**	**0.248**	—

Table 23: **JA vs. WJ by category, per anchor mode.** math = {aime24, aime25, brumo25, hmmt25}; code = {livecodebench_v5}; science = {gpqa}. No cell is significant; the largest gap (`delimiter@marker`×math, +1.17 pp) is non-significant ($p = 0.108$) and does not survive FDR (App. A.13, BH $q = 0.70$).

Anchor mode	Category	n	JA (%)	WJ (%)	WJ−JA	b	c
`boundary@marker`	math	934	75.91	76.66	+0.75	15	22
`boundary@marker`	code	1278	69.41	70.34	+0.94	35	47
`boundary@marker`	science	1574	68.68	68.55	−0.13	53	51
`delimiter@marker`	math	941	75.56	76.73	+1.17	14	25
`delimiter@marker`	code	1291	70.49	70.64	+0.15	49	51
`delimiter@marker`	science	1576	69.73	69.92	+0.19	46	49
`delimiter@16`	math	941	80.77	80.87	+0.11	6	7
`delimiter@16`	code	1291	71.88	71.88	+0.00	51	51
`delimiter@16`	science	1576	70.69	71.19	+0.51	15	23
`boundary@16`	math	934	80.73	79.76	−0.96	16	7
`boundary@16`	code	1278	70.50	71.13	+0.63	51	59
`boundary@16`	science	1574	70.01	69.57	−0.44	40	33

that crosses zero. No category cell is significant (Table 23), and no per-model / per-dataset / category cut survives BH-FDR (0/26); the two Qwen models lean in opposite directions, consistent with noise rather than a systematic edge. The ablation is adequately powered (Table 24): pooled power to detect a true 1 pp effect is ≈ 1.00 and the pooled significance boundary (0.38 pp) exceeds the pooled observed gap (0.22 pp), so a paper-relevant (≥ 1 pp) *pooled* kernel effect would almost certainly have been detected, an informative null, not a false negative. The kernel is therefore not load-bearing: the agreement signal lives in the routing geometry, not in the weighting, so binary and weighted Jaccard are interchangeable in practice. Numbers reproduce one-click via `wj_ja_compare.py`.

A.14 Layer-depth profile of the routing signal

This appendix gives the full layer-depth ablation summarised in §5.3. We profile the selector at depth-decile resolution: the ten deciles of the MoE stack indexed from the shallowest, each ablated three ways, a single decile alone ($s_0 \ldots s_9$), the cumulative shallow band ($c_0 \ldots c_8$, growing from the top), and leave-one-decile-out ($x_0 \ldots x_9$), under all four anchor configurations (boundary vs. delimiter anchor × anchor-only vs. 16-token window) and both the weighted (WJ) and binary (JA) Jaccard kernels. Picks reproduce an independent band-level recomputation bit-for-bit (45,608 problem-level cells, zero mismatches), so the decile profile and the coarser band picks are the same underlying selection. Significance uses the exact paired McNemar test (full vs. each band, Δ = full − band) with Holm correction within each configuration, plus an omnibus Cochran's Q over each band family; all numbers regenerate via `claim_audit.py`.

Table 24: **The kernel ablation is adequately powered.** Exact *conditional* McNemar power (conditioning on the discordant count $n_{\text{disc}} = b + c$) at $\alpha = 0.05$ (two-sided), $1 - \beta = 0.80$. MDE@80% and the post-hoc significance boundary are WJ−JA accuracy gaps (pp). Pooled power to detect a true 1 pp effect is ≈ 1.00 and the pooled significance boundary (0.38 pp) exceeds the pooled observed gap (0.22 pp), so a paper-relevant (≥ 1 pp) *pooled* kernel effect would almost certainly have been detected. The null in Table 22 is informative, not a false negative. Per anchor mode, power@1pp is 0.68–0.84, so a ≥ 1 pp effect confined to one mode could occasionally be missed.

Scope	n_{disc}	MDE@80%	sig. bound	power@1pp	power@0.5pp	obs. |WJ−JA|
`boundary@marker`	223	1.12	0.82	0.70	0.23	0.45
`delimiter@marker`	234	1.15	0.84	0.68	0.22	0.42
`delimiter@16`	153	0.95	0.71	0.84	0.29	0.24
`boundary@16`	206	1.08	0.79	0.74	0.24	0.21
Pooled	**816**	**0.53**	**0.38**	**≈1.00**	**0.75**	**0.22**

Necessity and sufficiency. Across all four configurations and both kernels, no single decile is necessary: every leave-one-decile-out change is ≤ 0.47 pp and $0/10$ survive Holm in each cell. The shallowest tenth alone (cumulative band c_0) is within 0.7 pp of the full network at the point estimate (95% deficit upper bound ≤ 1.7 pp, ≤ 1.2 pp for the windowed configurations) and is never significantly worse (per-config McNemar $p \geq 0.23$; pooled $p = 0.19$). For the marker-only (anchor-token) configurations no single decile differs significantly from the full network (every per-config McNemar Holm-adjusts to n.s.); for the boundary anchor the single-decile omnibus is also not significant (Cochran's Q $p = 0.67$, Table 25). So the sufficiency of a small slice is *not* a shallow-only effect.

Table 25: **Single-decile layer profile for the marker-only (boundary anchor) configuration** (weighted Jaccard, $n = 3786$ problems, full-network accuracy 71.16%). Each row uses *only* one tenth of the MoE layers, indexed from the shallowest (s_0) to the deepest (s_9); Δ is full − decile (positive = full better), with the exact paired McNemar p and its Holm-adjusted value. No single decile (shallow, middle, or deep) differs significantly from the full network (omnibus Cochran's $Q = 6.72$, df $= 9$, $p = 0.67$; $0/10$ survive Holm), so a small slice suffices regardless of depth.

Decile	Acc. (%)	Δ (pp)	McNemar p	Holm
s_0 (shallowest)	70.73	+0.42	0.456	1.000
s_1	71.24	−0.08	0.921	1.000
s_2	70.31	+0.85	0.134	1.000
s_3	70.05	+1.11	0.045	0.448
s_4	70.73	+0.42	0.451	1.000
s_5	70.81	+0.34	0.560	1.000
s_6	70.71	+0.45	0.422	1.000
s_7	70.36	+0.79	0.168	1.000
s_8	70.36	+0.79	0.164	1.000
s_9 (deepest)	70.29	+0.87	0.130	1.000
full	71.16	—	—	—

Where a single-decile preference does appear, it is GPQA-specific. At decile resolution the two *windowed* configurations do show a shallow-favoring localization (single-decile omnibus Cochran's Q significant: boundary+16-token $p = 6.5 \times 10^{-3}$ WJ / 2×10^{-4} JA). This significance is driven by GPQA: the shallow−middle gap is +2.9 pp (WJ) / +3.3 pp (JA) on GPQA (Q $p < 10^{-4}$) but only +0.2 pp (WJ, Q $p = 0.25$) / +0.7 pp (JA, Q $p = 0.30$) outside GPQA, not significant in either kernel (Figure 11). It does not raise full-network accuracy and is best read as a GPQA-specific interpretability observation, not a general depth-localization law. Figure 12 gives the per-configuration shallowest-tenth-vs-full comparison underlying the main-text sufficiency result.

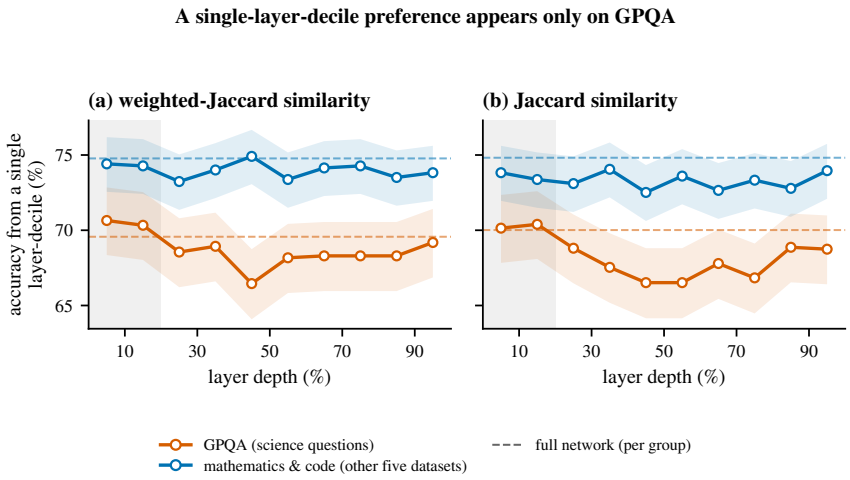


Figure 11: **The single-layer-decile preference is specific to GPQA.** Selection accuracy using *only one* layer-decile of routing, as a function of layer depth, for the boundary anchor with a 16-token routing window, split into GPQA (graduate-level science questions) vs. the five mathematics and code datasets, under the **(a)** weighted-Jaccard (primary) and **(b)** Jaccard similarity kernel; dashed lines mark each group's full-network accuracy. On GPQA the shallow deciles match the full network while the middle deciles fall about 3 percentage points lower (single-decile omnibus Cochran's-Q test $p < 10^{-4}$ under both kernels); on mathematics and code the profile is flat ($p = 0.25$ weighted-Jaccard, 0.30 Jaccard; not significant). The pooled preference is therefore driven by GPQA and is not a general depth-localization law; it does not raise full-network accuracy.

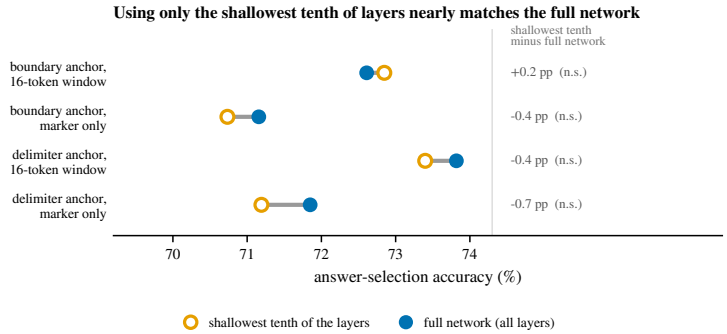


Figure 12: **Using only the shallowest tenth of the layers nearly matches the full network, for every selector setting.** For each of the four settings (anchor point × routing window), open markers give the answer-selection accuracy when the selector reads only the shallowest tenth of the mixture-of-experts layers, and filled markers give the full-network accuracy (weighted-Jaccard similarity kernel); the bar links the two. The shallowest tenth is within 0.7 percentage points of the full network in every setting (for the boundary+16-token setting nominally 0.2 pp above), and the difference is not statistically significant in any of them (paired McNemar exact test, $p \geq 0.23$). The Jaccard kernel reproduces this (gaps ≤ 0.8 percentage points; not significant), and the audit script regenerates all values.

A.15 Composition control: what the routing selector selects

This appendix details the composition control that dissects the RAD selector (§5.3, Figure 4). We pool the eight thinking-class model configurations (gpt-oss-20B/120B at Low/Medium/High effort, Qwen3-30B-Thinking, Qwen3-Next-80B-Thinking) over the six datasets at N=64 rollouts per problem and study two things: (i) how accuracy scales with the cohort size $m \in \{2, 4, 8, 16, 32, 64\}$, and (ii) a composition control that fixes m and injects exactly c correct rollouts, sweeping the correct fraction f=c/m. RAD uses the weighted-Jaccard k-NN density (k=10, window 16); RAD+DC

additionally filters by DeepConf confidence (top-50% by cBW, then density within the kept set); maj is the answer-aware plurality reference and oracle is pass@m. Every selector is answer-string-free except maj. We report both the *boundary* anchor (end-of-thinking / final-response marker; n=2832 at pool≥64) and the deployed *delimiter* anchor (`\boxed` / code-fence marker; n=2636); the paper default is the delimiter anchor (Table 1). Uncertainty is a two-stage bootstrap (B=200 sub-cohort draws per cell, then a 2000-resample cluster bootstrap over problems). **Eval-conditioning.** The pool is the anchor-found ∧ labeled ∧ feature-present rollouts (mean anchor coverage ≈0.90); the *absolute* accuracies below are therefore eval-conditioned and sit several points above an all-problems full-64 deployment. They evaluate the selector on its applicable subset, not a deployment number. The lift-over-chance composition results (controlled f) and all paired CIs are unaffected by this conditioning.

Scaling with the rollout count. RAD's pick accuracy rises monotonically with m (Table 26, Figure 13): the paired slope from m=2 to 64 is +3.26 pp at the delimiter anchor (+2.90 boundary), with a 95% CI that excludes 0. DeepConf does *not* scale (slope CI includes 0 at both anchors; it peaks by m≈8 and mildly declines thereafter), the signature of a per-rollout quality signal, not a consensus one. Fused, RAD+DC scales and at m=64 statistically ties the answer-aware Majority: the paired gap RAD+DC−maj is +0.11 pp [−0.83, +1.06] at the delimiter anchor and −0.39 pp [−1.45, +0.64] at the boundary anchor, both including 0 (the binary-Jaccard kernel reproduces the tie, −0.64 pp [−1.66, +0.35]). The per-domain split is Figure 14.

Table 26: **Pick accuracy (%) vs. cohort size** m (pool≥64; weighted Jaccard; answer-string-free selectors except maj). Paired slopes m=2 → 64 (pp): boundary RAD +2.90 [+2.04, +3.74], DC +0.37 [−0.56, +1.35], RAD+DC +3.30 [+2.53, +4.10]; delimiter RAD +3.26 [+2.38, +4.09], DC −0.36 [−1.33, +0.60], RAD+DC +2.98 [+2.27, +3.75]. Paired gap RAD+DC−maj at m=64: boundary −0.39 [−1.45, +0.64], delimiter +0.11 [−0.83, +1.06] (both include 0).

m	chance	RAD	DC	RAD+DC	maj	oracle
Boundary anchor (n = 2832)						
2	71.78	71.75	72.73	72.73	73.51	78.99
4	71.79	72.85	73.20	73.20	74.61	84.21
8	71.81	73.56	73.48	73.94	75.67	87.92
16	71.78	74.00	73.48	74.53	76.29	90.52
32	71.79	74.23	73.37	75.09	76.47	92.36
64	71.79	74.65	73.09	76.02	76.41	93.61
Delimiter anchor (n = 2636, deployed)						
2	74.38	74.36	74.98	74.98	75.57	81.10
4	74.38	75.83	75.25	75.25	76.42	85.90
8	74.40	76.48	75.41	76.45	77.32	89.32
16	74.37	76.79	75.30	76.99	77.82	91.68
32	74.38	77.05	75.06	77.31	77.93	93.38
64	74.39	77.62	74.62	77.96	77.85	94.50

Cohort size on the natural pool. Table 26 is eval-conditioned. Table 27 recomputes RAD and Cons (textual Majority) on exactly the same sampled rollouts for $m \in \{4, 8, 16, 32, 64\}$ over all 3,180 math+GPQA cells with the extraction, fallback, and tie rules of Table 1; for $m < 64$ each point averages 200 seed-fixed paired random sub-cohorts. Both improve with m (RAD +4.27 pp from m=4 to 64, Cons +3.21 pp) and the gap narrows monotonically, but RAD trails Cons through m=32: this establishes pool-size improvement, not superior sample efficiency.

Composition control. Fixing m=16 on a difficulty-balanced fixed set (≥ 15 correct ∧ ≥ 15 incorrect, n=703 boundary / 664 delimiter) and injecting c correct rollouts, RAD's lift over chance is a clean S-curve in f (Table 28, Figures 4 and 15): strongly negative when correct answers are a minority (−8.2 pp at f=0.25), crossing zero near f≈0.4, and large-positive once a majority (+20 boundary / +24 delimiter pp at f=0.62). DeepConf is the complementary signal, an inverted-U positive in the rare-correct regime where RAD is negative (peak +9.7 pp at f=0.25), with no sign flip. So RAD+DC inherits both (e.g. boundary c=8: RAD +10.9 → RAD+DC +12.5; c=1: −3.7 → −2.9). Beyond this regime-complementarity, the two are near-orthogonal as per-rollout

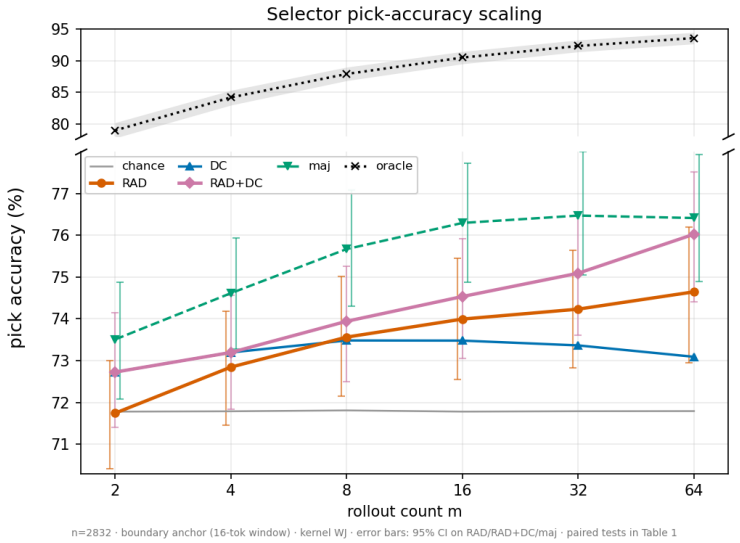


Figure 13: **Pick accuracy scales with the rollout count** m (boundary anchor, n=2832, weighted Jaccard; broken y-axis: the ≈70–78% selector comparison below, the 78–95% oracle ceiling above). RAD rises with m while DeepConf saturates by m≈8; fused, RAD+DC tracks the answer-aware Majority. Error bars are 95% cluster-bootstrap CIs on RAD/RAD+DC/maj (paired tests in Table 26).

Table 27: **RAD vs. Cons by cohort size** m **on the natural pool** (accuracy %, math+GPQA, n=3,180 cells, same denominator as Table 1).

m	RAD	Cons	RAD−Cons
4	69.60	70.37	−0.77
8	71.35	72.09	−0.74
16	72.50	72.94	−0.44
32	73.21	73.41	−0.20
64	73.87	73.58	+0.28

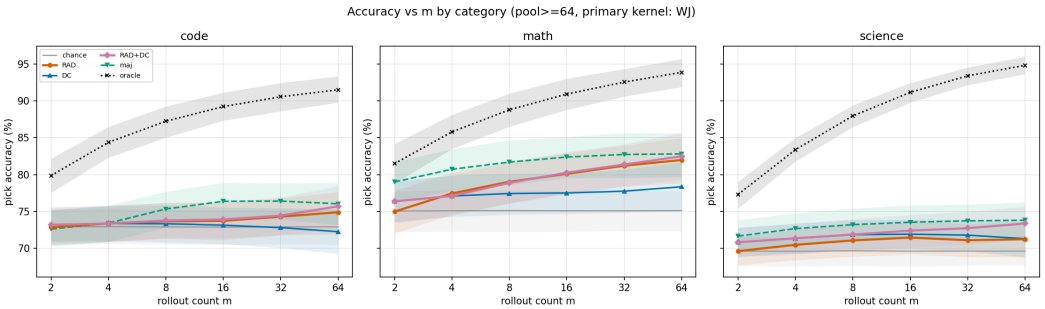


Figure 14: **Accuracy vs.** m **by domain** (pool≥64, weighted Jaccard, boundary anchor). The selector comparison (RAD/DC/RAD+DC vs. Majority and the oracle ceiling) on code, math, and science; the RAD-to-Majority gap and the oracle headroom are largest on mathematics.

signals (mean within-problem Pearson $r = 0.03$ between the DeepConf cBW scalar and the K-NN routing density, under both kernels; Table 31), so RAD+DC fuses non-redundant axes rather than re-weighting one. The answer-aware maj c=1 entry is a lowest-id tie-break artifact (it flips +2.5 boundary → −1.4 delimiter) and is omitted. The raw pick accuracies behind these lifts (with the y=f chance diagonal) are in Figure 16, and the S-curve amplitude grows with m (Figure 17).

Table 28: **Composition control: lift over chance (pp) at** m=16 on the difficulty-balanced fixed set (inject c correct rollouts, f=$c/16$; n=703 boundary / 664 delimiter). RAD's S-curve (negative when correct is a minority, positive once a majority) is the consensus signature; DeepConf is positive in the rare-correct (low-f) regime where RAD is negative.

		Boundary anchor			Delimiter anchor		
c	f	RAD	DC	RAD+DC	RAD	DC	RAD+DC
1	0.06	−3.70	+5.85	−2.90	−3.81	+3.55	−3.02
2	0.12	−6.47	+8.55	−4.22	−6.69	+5.39	−4.53
4	0.25	−8.16	+9.65	−1.65	−8.16	+6.07	−0.95
6	0.38	−1.77	+8.91	+6.58	−0.01	+5.47	+6.36
8	0.50	+10.89	+6.61	+12.53	+17.23	+3.57	+13.45
10	0.62	+20.11	+4.56	+15.15	+23.77	+2.07	+16.86
12	0.75	+16.49	+2.06	+14.12	+18.08	+0.31	+15.72
14	0.88	+8.82	+0.15	+8.30	+9.43	−0.84	+9.07
15	0.94	+4.51	−0.30	+4.27	+4.77	−0.84	+4.57

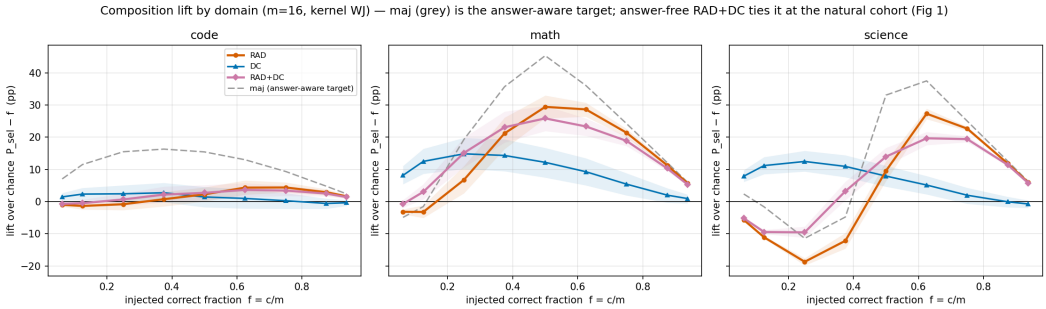


Figure 15: **Composition lift by domain** (m=16, weighted Jaccard, boundary anchor): lift over chance vs. injected correct fraction f, with the answer-aware Majority (grey) as the target. The RAD consensus S-curve is strongest on mathematics and flattest on code, matching the domain ordering of Table 30.

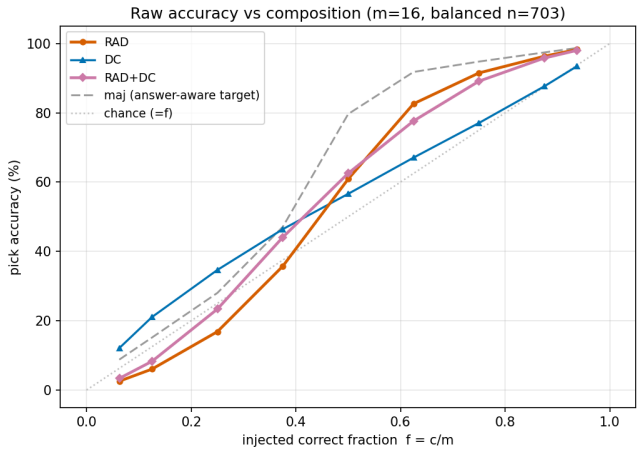


Figure 16: **Raw pick accuracy behind the composition lift** (m=16 balanced, n=703, boundary anchor): absolute pick accuracy vs. f, with the y=f chance diagonal. RAD overtakes DeepConf once correct answers approach a majority ($f \gtrsim 0.4$); below that DeepConf is higher.

The single-needle regime. When exactly one rollout is correct (c=1, f=$1/m$) the dichotomy is sharpest (Table 29): DeepConf finds the needle above chance at every m ($+7.2/+5.2/+2.9$ pp boundary at m=8/16/32) while RAD is below chance ($-6.8/-3.9/-2.3$), since a lone correct

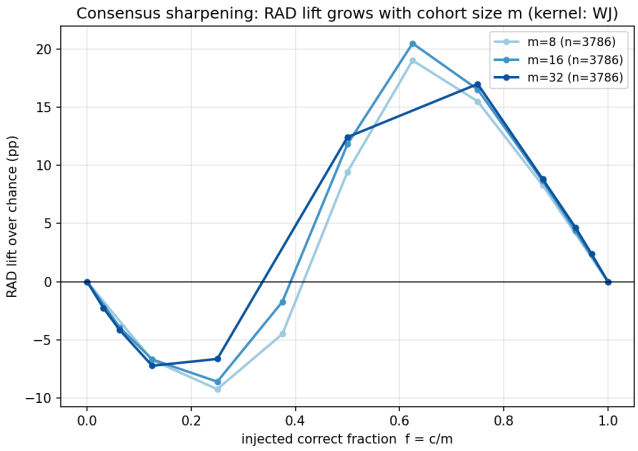


Figure 17: **The consensus S-curve sharpens with cohort size.** RAD lift over chance vs. f for m=8/16/32 (feasible set, n=3786, boundary anchor): the S-curve amplitude grows with m, the mechanism behind RAD's scaling in Table 26.

rollout is by definition outside the consensus basin. The operational corollary is that low-pass-rate / single-needle regimes should prefer DeepConf to RAD. The DeepConf needle lift is family-skewed (Qwen-heavy); we report it pooled and do not break it out per family.

Table 29: **Single-needle regime (c=1): lift over chance (pp).** Exactly one correct rollout (f=1/m). DeepConf is above chance at every m; RAD is below chance, since a lone correct rollout is outside the consensus basin.

m	RAD	DC	RAD+DC	n
Boundary anchor				
8	−6.75	+7.23	−4.05	1440
16	−3.93	+5.23	−2.94	1140
32	−2.26	+2.92	−1.72	738
Delimiter anchor				
8	−7.21	+5.01	−4.80	1418
16	−3.96	+3.27	−3.10	1091
32	−2.07	+2.03	−1.69	699

Domain dependence. RAD's lift over chance at m=64 orders mathematics $\gg$ science, code (Table 30): +6.9 pp on math vs. +1.6 (science) / +2.0 (code) at the boundary anchor, with all three CIs excluding 0 and a Holm correction over the three domains leaving the calls unchanged. DeepConf mirrors RAD on math but is null on code at both anchors (and on science at the delimiter anchor). This is the same math > science > code ordering reported for selector intelligence Δ_{sel} in Table 12, here measured as lift over chance rather than over the cohort baseline; the value of routing-space consensus is bounded by how routing-coherent correct reasoning is: high on math and marginal elsewhere.

DeepConf on the natural distribution. The composition control above injects a uniform sweep of f; on the *natural* pool, by contrast, most cohorts are already converged (correct answers a clear majority), the regime where DeepConf's complementary lift is smallest. This is why adding DeepConf does not significantly change RAD on the natural distribution (paired exact McNemar n.s. in every well-posed cell after Holm correction, Table 18) even though the composition control and the per-rollout correlation (Table 31) show the two signals are genuinely near-orthogonal: that near-orthogonality is realised mainly in the rare-correct tail, which is under-represented at the natural cohort. Under weighted Jaccard no cell is significantly harmful.

Table 30: **Domain dependence: RAD/DC/RAD+DC lift over chance (pp) at m=64,** with 95% cluster-bootstrap CIs. Mathematics ≫ science, code; a Holm correction over the three domains leaves the RAD calls unchanged. (n per domain: math 605/592, science 1355/1223, code 872/821 for boundary / delimiter.)

Domain	RAD	DC	RAD+DC
Boundary anchor			
math	+6.88 [+5.20, +8.54]	+3.25 [+1.13, +5.42]	+7.38 [+5.62, +9.16]
science	+1.61 [+0.41, +2.85]	+1.68 [+0.19, +3.19]	+3.75 [+2.62, +4.94]
code	+1.99 [+0.35, +3.63]	−0.65 [−2.42, +1.10]	+2.79 [+1.05, +4.43]
Delimiter anchor			
math	+6.19 [+4.75, +7.77]	+3.32 [+1.24, +5.46]	+6.70 [+5.02, +8.31]
science	+2.20 [+1.06, +3.23]	−0.67 [−2.14, +0.77]	+2.36 [+1.26, +3.55]
code	+2.64 [+0.90, +4.38]	−0.65 [−2.41, +1.13]	+3.12 [+1.58, +4.75]

Table 31: **DeepConf confidence and RAD routing density are near-orthogonal axes.** Mean within-problem Pearson correlation between the per-rollout DeepConf cBW scalar and the K-NN routing density (K=10) over 3,765 pools (sd = 0.26); near zero under both kernels and in every split, so RAD+DC fuses two non-redundant axes rather than re-weighting one.

Kernel	All	Well-posed	Code	% pools $\lvert r\rvert < 0.2$
Weighted Jaccard (deployed)	0.03	0.02	0.05	61
Binary Jaccard	0.03	0.02	0.06	60

Cross-anchor robustness. Every finding above replicates at both anchors (Figure 18): RAD scales (slope +2.90 boundary / +3.26 delimiter), DeepConf does not, the composition S-curve preserves its shape and sign flip, the single-needle dichotomy holds, and RAD+DC ties Majority at m=64 (−0.39 / +0.11 pp). The boundary anchor (read at the end-of-thinking marker) and the delimiter anchor (read at the boxed-answer / code-fence delimiter) thus expose the same mechanism; the deployed selector uses the delimiter anchor (§4).

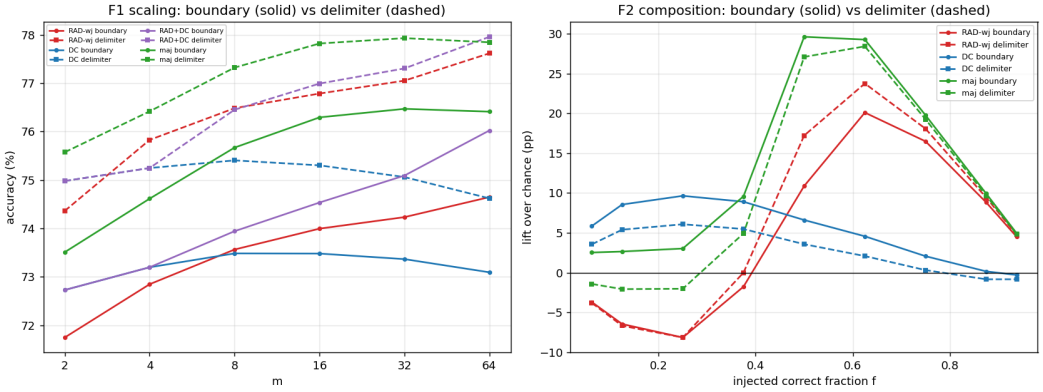


Figure 18: **Cross-anchor robustness: boundary (solid) vs. delimiter (dashed).** *Left:* accuracy vs. m. *Right:* composition lift vs. f. The scaling, saturation, S-curve, and Majority-tie all replicate at both anchors (weighted Jaccard).

A.16 Failure case studies

Table 32 summarises two qualitative regimes that route geometry exposes; we then anchor each regime to the data via a structural signature.

We identify the two regimes using deterministic structural predicates over the rollout pool:

Table 32: **Qualitative regimes for routing-based selection.** Two classes of cases that recur across the benchmark and motivate RAD's selection rule.

Regime	What route geometry shows	What RAD does	Lesson
Code-generation case	Exact-string voting degenerates; one route basin is denser	RAD selects the basin centre and returns a direct pass@1 trajectory	Routing density operates where lexical voting cannot
False consensus	The largest route basin is wrong	RAD follows the dense wrong basin	Basin agreement is not truth

- *Tie-breaking success.* Problems on which textual Majority's tie set has size ≥ 3 but RAD's route density has a clear single argmax that matches the correct basin. These are over-represented on AIME25, BRUMO25, and (for OSS Thinking models) LiveCodeBench.
- *False consensus.* Problems on which the route basin selected by RAD has ≥ 32 rollouts and is wrong. These are concentrated on the hardest difficulty bucket of GPQA (where many rollouts converge to the same plausible-but-wrong answer choice) and a small subset of HMMT25.

Quantitative, mutually exclusive breakdown. The false-consensus predicate above covers 84.2% of math failures, 90.2% of GPQA failures, and 88.3% overall, but overlaps pools in which all 64 candidates are wrong. Table 33 attributes every RAD failure on the well-posed pool to exactly one cause, conditioning route false consensus on a correct candidate being present: 66.9% of failures occur although a correct rollout exists because RAD follows a wrong majority route basin, 26.5% are unrecoverable from the fixed pool, and 6.6% come from a smaller wrong basin or an unavailable readout. Routing density estimates shared commitment, not truth.

Table 33: **Mutually exclusive attribution of RAD failures** (well-posed pool of Table 1; counts, row %).

Scope	RAD failures	No correct candidate	False consensus, correct exists	Wrong basin < 32	No valid route basin
Math (4 sets)	260	102 (39.2%)	140 (53.8%)	10 (3.8%)	8 (3.1%)
GPQA	571	118 (20.7%)	416 (72.9%)	34 (6.0%)	3 (0.5%)
All	831	220 (26.5%)	556 (66.9%)	44 (5.3%)	11 (1.3%)

A.17 Why routing agreement can work: an informal account

Routing need not encode correctness; it only needs to preserve solution-basin similarity. At a phase-aligned commitment position, trajectories following the same latent strategy induce more similar context-conditioned hidden states, and when router margins are stable, nearby hidden states keep overlapping top-k expert assignments, so routing similarity is a sparse proxy for shared computation. RAD selects the locally densest such basin; standard self-consistency then links basin prevalence, not routing itself, to a higher probability of correctness. This predicts the observed failures: a coherent misconception that dominates the pool (App. A.16) or correct solutions fragmented across strategies, as on code where functionally equivalent programs disperse across route neighborhoods while common flawed templates form compact basins (App. A.4); and it predicts the successes on compact correct minorities that textual voting outvotes (App. A.3).