REM-MOA: REASONING MEMORY SUSTAINS MIXTURE-OF-AGENTS SCALING

Anonymous authors
Paper under double-blind review

ABSTRACT

Mixture-of-Agents (MoA) architectures improve inference-time scaling by organizing multiple LLM agents into layered reasoning pipelines. However, existing MoA variants struggle to sustain gains as depth increases, exhibiting degradation, early plateauing, or saturation on most benchmarks. We propose ReM-MoA, a memory-augmented MoA framework that sustains scaling through two mechanisms: (1) a Ranked Reasoning Memory that persistently stores and ranks reasoning traces from all layers using a comparative Reviewer Agent, and (2) a Curated Diversified Memory Routing scheme that exposes different agents to distinct combinations of successful and failed traces, preserving exploration diversity while propagating high-quality reasoning. We further introduce an optional multi-domain Reviewer distillation pipeline that improves ranking quality through frontier-model supervision. ReM-MoA consistently outperforms prior MoA variants under both depth and width scaling on five reasoning benchmarks spanning math, formal logic, code, knowledge, and commonsense. As depth grows, ReM-MoA widens the lead and gains nearly three times as much accuracy per additional token as any baseline, establishing structured cross-layer reasoning memory as a key missing mechanism for scalable MoA inference.

1 INTRODUCTION

Inference-time computation has emerged as a complementary scaling axis to parameter scaling for large language models (Snell et al., 2024). Among inference-time strategies, multi-LLM collaboration leverages collective reasoning by aggregating outputs from multiple agents, with paradigms ranging from majority voting (Wang et al., 2022) to multi-agent debate (Du et al., 2024; Liang et al., 2024). A more recent paradigm, **Mixture-of-Agents (MoA)** (Wang et al., 2025), organizes N proposer agents into L stacked layers followed by a final aggregator, enabling iterative refinement across depth. However, existing MoA systems generally fail to sustain gains across benchmarks as architectures deepen, often degrading, plateauing, or saturating after only a few layers.

Existing MoA variants attempt to sustain scaling through three broad strategies. The first strategy, *information consolidation*, curates the inputs of each layer to reduce redundancy, exemplified by RMoA (Xie et al., 2025), which selects a diverse subset of prior outputs and forwards them together with the inter-layer residuals to the next layer. The second, *inter-agent interaction*, exemplified by AttentionMoA (Wen et al., 2026), enables agents to refine each other within a layer through natural-language attention. The third, *quality-guided selection*, exemplified by VeriMoA (Ping et al., 2025), maintains a global cache of the highest-scoring solutions but requires domain-specific verifiers such as hardware description language simulators, limiting its applicability to specialized tasks.

While these methods differ architecturally, none sustains performance across benchmarks as MoA depth increases. Three mechanisms drive this scaling collapse (Xie et al., 2025; Wen et al., 2026). First, raw inter-layer concatenation propagates both useful reasoning and accumulated errors, allowing mistakes to compound across depth. Second, architectures with only local inter-layer communication fail to preserve strong reasoning trajectories generated earlier in the hierarchy. Third, as agents consume increasingly similar accumulated context, exploration diversity collapses and agents converge toward redundant solution paths, undermining the core benefit of multi-agent reasoning (Zhang et al., 2025; Li et al., 2025). More importantly, retaining memory alone is insufficient: exposing all agents to identical accumulated reasoning can itself accelerate convergence toward repetitive trajectories.

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.

Sustained MoA scaling therefore requires simultaneously preserving high-quality reasoning across layers while maintaining diversity among agents within each layer.

To address these scaling failures, we propose **ReM-MoA**, a Mixture-of-Agents framework centered on a *Ranked Reasoning Memory* that stores reasoning traces produced across the layers. Unlike episodic or retrieval-based memory systems that accumulate experience across tasks, ReM-MoA maintains a structured cross-layer reasoning memory within a single problem instance, enabling deeper layers to revisit prior reasoning paths. At each layer, proposer agents generate reasoning traces, and a *Reviewer Agent* (Zheng et al., 2023) comparatively evaluates these traces by assigning scores together with corresponding rationales. The ranked traces and rationale are then stored in memory (Ouyang et al., 2025). Subsequent layers consult this memory through a *Curated Diversified Memory Routing* scheme that exposes different agents to distinct combinations of high-quality and low-quality reasoning trajectories, providing contrastive reasoning signals while preserving exploration diversity.

ReM-MoA therefore treats MoA scaling as a joint quality-diversity preservation problem rather than solely an aggregation or interaction problem. As an optional enhancement, the Reviewer Agent can be distilled from a frontier model on multi-domain data (Hinton et al., 2015), yielding a general-purpose Reviewer that transfers across reasoning tasks. Our contributions are as follows:

- We empirically characterize three scaling failure modes in existing MoA systems: degradation under raw concatenation, early plateauing under local residual propagation, and diversity collapse under intra-layer interaction mechanisms. These observations motivate structured cross-layer reasoning memory as a critical mechanism for sustained MoA scaling.
- We propose **ReM-MoA**, which sustains MoA scaling through a Ranked Reasoning Memory and a Curated Diversified Memory Routing scheme. At each layer, a comparative Reviewer Agent ranks reasoning traces with rationales, and subsequent agents receive distinct combinations of successful and failed traces, jointly preserving quality and diversity. An optional frontier-model distillation pipeline further upgrades the Reviewer Agent through multi-domain training.
- Across five benchmarks (math, formal logic, code, knowledge, and commonsense), ReM-MoA consistently outperforms prior MoA variants across all tested configurations, with the performance gap widening rather than diminishing as the architecture scales. ReM-MoA gains nearly three times as much accuracy per additional token as any baseline under depth scaling, exceeds compute-matched self-consistency by up to 9.3 points, and keeps the lead across multiple model sizes.

2 Related Work

Mixture-of-Agents variants. Building on the original MoA framework (Wang et al., 2025), subsequent variants extend MoA along three paradigms. *Information consolidation* curates the inputs of each layer: RMoA (Xie et al., 2025) forwards a diverse subset of prior outputs together with inter-layer residuals, while Self-MoA (Li et al., 2025) argues that quality dominates diversity and aggregates samples from a single strong model. *Inter-agent interaction* lets agents refine each other within a layer, as in AttentionMoA (Wen et al., 2026). DEI (Zhang et al., 2025) follows the same paradigm and ensembles specialist agents with complementary expertise. *Quality-guided selection*, exemplified by VeriMoA (Ping et al., 2025), keeps a global cache of the highest-scoring solutions scored by a domain-specific verifier. No prior MoA variant maintains a structured cross-layer memory of reasoning traces accessible to subsequent layers.

Memory in LLM agents. Explicit memory has been used for LLM agent self-improvement: Reflexion (Shinn et al., 2023) stores verbal self-reflection from past failures, ReasoningBank (Ouyang et al., 2025) distills reasoning strategies from successful and failed trajectories across tasks, and AgentAuditor (Luo et al., 2026) augments LLM evaluators with retrieval-augmented examples for safety judgment. Our Ranked Reasoning Memory instead operates within a single problem across MoA depth, holding ranked traces from each layer for subsequent layers to consult.

LLM-as-Judge and judge distillation. LLM-as-Judge (Zheng et al., 2023) has emerged as a scalable proxy for human preference, with pairwise comparison generally more reliable than independent scoring. JudgeLM (Zhu et al., 2025) distills frontier-model judgments into open-source judges via knowledge distillation (Hinton et al., 2015) and parameter-efficient fine-tuning (Hu et al., 2022).

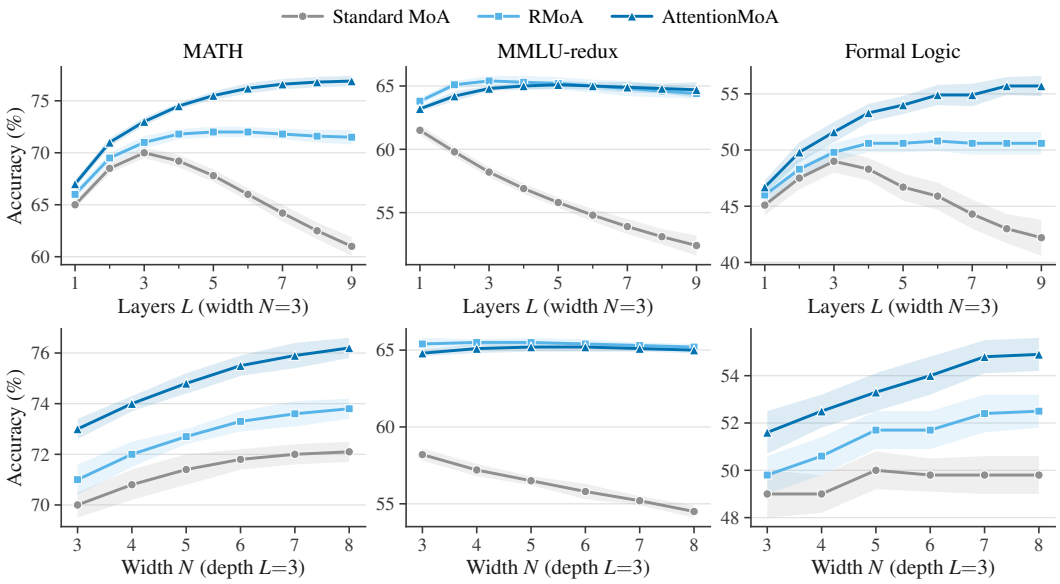

Figure 1: Scaling behavior of three existing MoA variants on three benchmarks (MATH, MMLU-redux, Formal Logic) along two scaling axes. All variants share the same heterogeneous proposer pool of Qwen2.5-7B, Qwen2.5-Coder-7B, and Llama-3.1-8B. **Top row**: layer depth scaling (L from 1 to 9, width fixed at $N = 3$). **Bottom row**: layer width scaling (N from 3 to 8, depth fixed at $L = 3$). Points are five-seed means, and shaded bands show one standard deviation.

Our Reviewer Agent extends this with a layer-wise comparative ranking that feeds a downstream reasoning memory.

3 Preliminaries: MoA Scaling

The standard MoA framework. The original Mixture-of-Agents framework (Wang et al., 2025) defines the *standard* MoA structure: N proposer agents are organized into L sequential layers followed by a final aggregator. Layer 1 receives the original problem x and produces N candidate responses. Each subsequent layer $l \in \{2, \ldots, L\}$ receives the previous layer's outputs together with x and produces N new responses. The aggregator then synthesizes the outputs of layer L into a final answer.

Quality and diversity drive scaling. Li et al. (2025) model MoA accuracy through a linear regression

$$t = \alpha q + \beta d + \gamma, \tag{1}$$

where q is the average proposer accuracy and d is the Vendi Score of proposer outputs. With $\alpha > \beta$ across datasets, MoA performance depends on both factors but is more sensitive to quality. Sustained scaling therefore requires mechanisms that preserve quality and diversity simultaneously, since gains in one cannot offset losses in the other.

Empirical scaling failures. Figure 1 plots the accuracy of three representative MoA variants under depth and width scaling. In the depth-scaling panels (top row), Standard MoA, RMoA, and AttentionMoA each exhibit a distinct failure pattern. Standard MoA (Wang et al., 2025) peaks at shallow depths and then degrades, with the drop reaching nearly 10 percentage points on MMLU-redux. RMoA (Xie et al., 2025) avoids this degradation but plateaus early, because its adjacent-layer residual cannot retrieve high-quality reasoning produced in non-adjacent earlier layers. AttentionMoA (Wen et al., 2026) improves but shows diminishing late-layer gains on most benchmarks (with slight late decline on MMLU-redux), since intra-layer attention promotes convergence among the layer's agents. Width scaling changes accuracy less: Standard MoA still degrades on MMLU-redux, while RMoA and AttentionMoA continue to improve with width on most benchmarks.

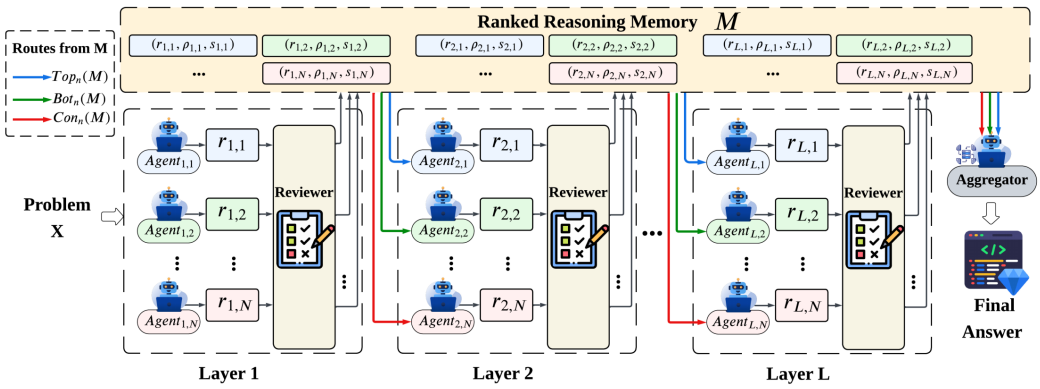

Figure 2: The ReM-MoA framework. At each layer, N proposer agents produce reasoning traces $r_{l,j}$, and the Reviewer Agent compares them and assigns per-trace scores $s_{l,j}$ and rationales $\rho_{l,j}$, committed to the Ranked Reasoning Memory $\mathcal{M}$. From layer 2 onwards, each proposer agent receives a curated reference set from $\mathcal{M}$ drawn from three families: Top_n (highest-scoring traces), Bot_n (lowest-scoring), or Con_n (contrastive, $\text{Top}_a \cup \text{Bot}_b$).

Design requirements. The three patterns above instantiate the failure mechanisms outlined in Section 1 and show that existing variants violate the quality-diversity balance of Equation (1) as MoA scales. Sustaining scaling therefore requires preserving proposer quality and output diversity:

(1) Q (Quality Preservation): A scalable MoA should retain the quality of reasoning produced at each layer so that deeper layers build on it rather than dilute it. Both high-quality successful traces (positive guidance) and low-quality failed traces (counter-examples) enrich the input signal beyond positive examples alone. Preserving quality therefore calls for presenting traces in a quality-ranked form rather than raw concatenation and for keeping traces from all earlier layers accessible.

(2) D (Diversity Preservation): A scalable MoA should also preserve exploration diversity among agents within each layer, so that they do not collapse onto a single solution path as memory accumulates. Accordingly, different agents in the same layer should engage with different selections from the memory rather than sharing identical inputs.

Building on this analysis, we propose ReM-MoA to meet requirements Q and D. In ReM-MoA, a Ranked Reasoning Memory ranks and retains the traces of every layer, and a Curated Diversified Memory Routing scheme delivers distinct quality-aware reference sets to the agents of each layer.

4 ReM-MoA

ReM-MoA addresses the failure patterns observed in Section 3 through three interconnected components: a Ranked Reasoning Memory $\mathcal{M}$, a curated diversified memory routing scheme Π, and an optional reviewer distillation pipeline. Let $\mathcal{A}_l = \{A_{l,1}, \ldots, A_{l,N}\}$ denote the N proposer agents at layer $l \in \{1, \ldots, L\}$, and let $r_{l,j} = A_{l,j}(P_{l,j})$ denote the reasoning trace produced by agent $A_{l,j}$ given prompt $P_{l,j}$. We write $\mathbf{r}_l = (r_{l,1}, \ldots, r_{l,N})$ for the vector of all traces produced at layer l. Figure 2 illustrates the two core components: the memory ranks and retains $\mathbf{r}_l$ at every layer (Section 4.1), and the routing scheme delivers distinct curated reference sets to within-layer agents (Section 4.2). The optional distillation pipeline further improves the Reviewer Agent that ranks the memory entries (Section 4.3). After layer L, the aggregator consults the memory $\mathcal{M}$ through routes drawn from the same three families as the proposer agents, and synthesizes the final answer.

4.1 Ranked Reasoning Memory

Memory subsystem. The Ranked Reasoning Memory $\mathcal{M}$ is a shared subsystem that records ranked traces at every proposer layer. After layer l has been processed, the memory accumulates as the union

of per-layer entries:

$$\mathcal{M}_{\leq l} = \bigcup_{k=1}^{l} \mathcal{M}_k, \tag{2}$$

where $\mathcal{M}_k$ is the entry committed by layer k (defined below). The memory is append-only and supports read access from any subsequent layer: for any $l' > l$, layer l' may consult $\mathcal{M}_{\leq l}$. This subsystem addresses Q (Quality Preservation) because traces are stored in quality-ranked form rather than raw concatenated outputs, and remain accessible across arbitrary layer distances rather than only between adjacent layers.

The Reviewer Agent. The ranking inside the memory is performed by an LLM-based Reviewer Agent $\mathcal{R}$. After the N proposers at layer l produce their traces $\mathbf{r}_l$, the Reviewer is invoked once with all N traces presented together. It compares the traces against each other and emits, for each trace, an integer score together with a corresponding rationale specific to that trace:

$$(\mathbf{s}_l, \boldsymbol{\rho}_l) = \mathcal{R}(x, \mathbf{r}_l), \tag{3}$$

where $\mathbf{s}_l = (s_{l,1}, \ldots, s_{l,N})$ with each $s_{l,j} \in \{1, \ldots, S\}$ being an integer score on a scale with maximum S (higher is better), and $\boldsymbol{\rho}_l = (\rho_{l,1}, \ldots, \rho_{l,N})$ with each $\rho_{l,j}$ being a rationale specific to trace $r_{l,j}$ (rationales are independent across traces). We use *comparative* scoring rather than independent scoring because grounding each trace's score in relative differences across candidates produces more reliable estimates than absolute scoring (Zheng et al., 2023).

Memory entries and access. Each layer commits one entry to the memory, formed by zipping the traces with their Reviewer-assigned scores and rationales:

$$\mathcal{M}_l = \{(r_{l,j}, s_{l,j}, \rho_{l,j})\}_{j=1}^{N}. \tag{4}$$

Each $\mathcal{M}_l$ holds N trace-score-rationale triples, so $\mathcal{M}_{\leq l}$ offers $N \cdot l$ triples for cross-layer retrieval.

4.2 Curated Diversified Memory Routing

Curated reference delivery. At every subsequent layer, each proposer agent receives, in addition to the problem x, a curated set of reference traces drawn from the memory. The selection follows two design principles. First, the reference set must carry quality-aware information for the agent's reasoning: it comprises high-scoring traces as positive examples, low-scoring traces as negative examples, or both. Second, the reference sets must differ across the N agents within the same layer, so the layer as a whole explores the memory's content rather than redundantly consuming the same references. Formally, at layer $l \geq 2$, the curated diversified memory routing scheme Π operates on the *global pool* of all previously committed entries, $\mathcal{P}_l = \mathcal{M}_{\leq l-1}$:

$$\mathcal{S}_{l,j} = \Pi_j(\mathcal{P}_l), \tag{5}$$

where $\mathcal{S}_{l,j} \subset \mathcal{P}_l$ is the curated reference set routed to agent $A_{l,j}$. Operating on $\mathcal{P}_l$ rather than only the most recent layer realizes the cross-layer access promised by the memory subsystem (Section 4.1). The prompt for agent $A_{l,j}$ is then:

$$P_{l,j} = \begin{cases} x & \text{if } l = 1, \\ x \oplus \mathcal{S}_{l,j} & \text{if } l \geq 2, \end{cases} \tag{6}$$

where $\oplus$ denotes concatenation and each entry of $\mathcal{S}_{l,j}$ enters the prompt as a complete trace-score-rationale triple (Equation (4)).

Routing functions. We implement the two design principles above through three families of routing functions operating on $\mathcal{P}_l$. Let $\text{Top}_n(\mathcal{M})$ denote the size-n subset of $\mathcal{M}$ containing the highest-scoring entries, and $\text{Bot}_n(\mathcal{M})$ the analogous subset for the lowest-scoring ones. The three families are: (1) *Contrastive*, of the form $\text{Top}_a \cup \text{Bot}_b$ (denoted Con_n with $n = a + b$), pairing high- and low-scoring entries. (2) *Positive reinforcement*, of the form Top_a, emphasizing successful patterns. (3) *Failure-mode focus*, of the form Bot_a, surfacing common pitfalls. For each layer, the N routing functions $\Pi_1, \ldots, \Pi_N$ are distributed across these three families, with the distribution chosen to balance the three modes of reference. As one concrete instance, for $N = 3$ we use one route per family:

$$\Pi_j(\mathcal{P}_l) = \begin{cases} \text{Top}_1(\mathcal{P}_l) \cup \text{Bot}_1(\mathcal{P}_l) & j = 1, \\ \text{Top}_2(\mathcal{P}_l) & j = 2, \\ \text{Bot}_2(\mathcal{P}_l) & j = 3. \end{cases} \tag{7}$$

Appendix A.7 details the distributions used for larger N in our scaling experiments. This design realizes both Q (Quality Preservation) and D (Diversity Preservation). For Q, the curated reference sets preserve the Reviewer-extracted quality signal as it propagates forward through the layers. For D, the N distinct routes within each layer sustain exploration diversity even as the memory grows.

4.3 Reviewer Distillation (Optional Enhancement)

Distillation pipeline. The Reviewer Agent $\mathcal{R}$ is an off-the-shelf LLM by default and can optionally be enhanced via frontier-model distillation (Hinton et al., 2015). The training data come from *ReM-MoA's own runs*: we execute the framework on a multi-domain set of training problems, collect the proposer traces $\mathbf{r}_l$ of each layer, and query a strong external teacher $\mathcal{T}$ on each (problem, traces) pair for supervision targets:

$$\mathcal{D}_{\text{train}} = \left\{ (x_i, \mathbf{r}_i, \mathbf{s}_i^*, \boldsymbol{\rho}_i^*) \mid (\mathbf{s}_i^*, \boldsymbol{\rho}_i^*) = \mathcal{T}(x_i, \mathbf{r}_i) \right\}. \tag{8}$$

A smaller base Reviewer $\mathcal{R}_\theta$ is then fine-tuned with LoRA (Hu et al., 2022) on $\mathcal{D}_{\text{train}}$ to imitate the teacher under the standard supervised loss. Because the training traces come from ReM-MoA itself, the distilled Reviewer learns to judge the trace distribution that ReM-MoA produces at inference. Multi-domain training lets a single distilled Reviewer improve accuracy on every evaluated benchmark, including three benchmarks absent from the distillation data (Section 5.5). Appendix A lists the full training configuration.

5 Experiments

We organize the evaluation around four research questions (RQs). **RQ1:** Does ReM-MoA continue to improve as depth and width scale, where prior MoA variants degrade, plateau, or saturate? **RQ2:** Is the advantage of ReM-MoA explained by additional compute or by proposer scale rather than by the architecture? **RQ3:** How much do the Ranked Reasoning Memory and the Curated Diversified Memory Routing each contribute to the advantage? **RQ4:** Does multi-domain Reviewer distillation transfer to held-out reasoning domains?

5.1 Experimental Setup

Benchmarks and metrics. We evaluate on five benchmarks with one benchmark per target domain. MATH (Hendrycks et al., 2021) covers mathematics and MMLU-redux (Gema et al., 2025) covers knowledge. Formal Logic, the `formal_logic` subject of MMLU (Hendrycks et al., 2020), covers formal logic. CRUX, short for CRUXEval (Gu et al., 2024), covers code understanding, and HellaSwag (Zellers et al., 2019) covers commonsense reasoning. We report accuracy on every benchmark except the code benchmark CRUX, for which we report pass@1.

Models. Three 7–8B instruction-tuned models form the heterogeneous proposer pool: Qwen2.5-7B-Instruct (Yang et al., 2024), Llama-3.1-8B-Instruct (Grattafiori et al., 2024), and Qwen2.5-Coder-7B-Instruct (Hui et al., 2024). Qwen2.5-7B-Instruct also serves as the aggregator in ReM-MoA, as the off-the-shelf Reviewer. ReM-MoA* uses a LoRA fine-tune of the same model distilled from GPT-5.5 on 500 training problems from five domains (Appendix A). Appendix G lists all prompt templates and the 1–10 scoring rubric of the Reviewer.

Baselines. For RQ1 and RQ3, we compare against Standard MoA (Wang et al., 2025), RMoA (Xie et al., 2025), and AttentionMoA (Wen et al., 2026) from Section 2. RMoA retains a diverse subset of three outputs per layer, which equals N in the depth experiments. VeriMoA (Ping et al., 2025) is not included because VeriMoA scores candidates by simulating generated hardware designs against testbenches, which our benchmarks do not provide. ReM-MoA and the MoA baselines share the same proposer pool, aggregator, and sampling hyperparameters. For RQ2, we add self-consistency (SC) (Wang et al., 2022) over each proposer model as a compute-matched control outside the MoA family. SC samples K independent responses and returns the majority answer, with K chosen to match the token consumption of ReM-MoA at each evaluated depth L (Appendix C.1).

Protocol. We sweep depth $L \in \{1, \ldots, 9\}$ at N=3 and width $N \in \{3, \ldots, 8\}$ at L=3. All results are means over five independent runs with seeds 0–4. We measure compute as the total number of input and output tokens per query, summed over all LLM calls. Sampling temperatures and hardware follow Appendix A.6.

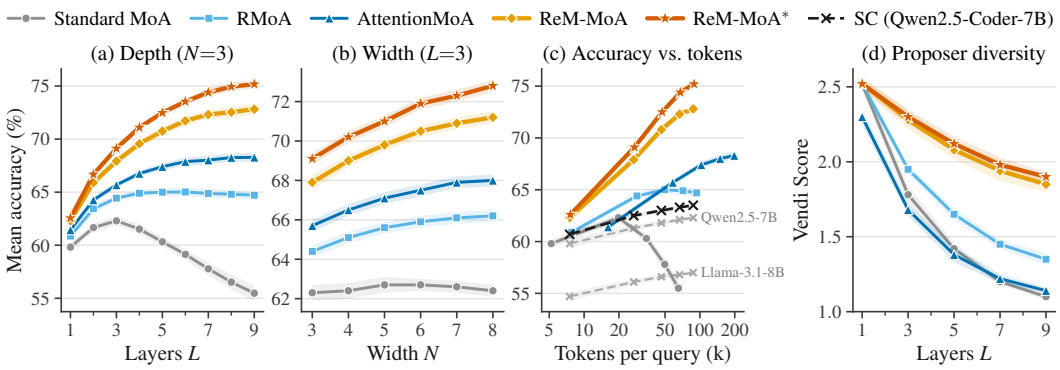

Figure 3: Scaling overview on the five-benchmark mean (accuracy, pass@1 for CRUX). (a) Depth scaling at N=3. (b) Width scaling at L=3. (c) Mean accuracy against tokens per query on a logarithmic axis under depth scaling at $L \in \{1, 3, 5, 7, 9\}$, together with compute-matched self-consistency (SC) over each proposer model. (d) Vendi Score of the N=3 proposer outputs, which ranges from 1 for identical outputs to 3 for orthogonal outputs. The SC budget at each depth matches the tokens of ReM-MoA. Points are five-seed means, and shaded bands show one standard deviation.

5.2 Sustained Scaling and Cross-Benchmark Results (RQ1)

ReM-MoA leads in every configuration. Figures 3(a) and 3(b) plot the five-benchmark mean with one-standard-deviation bands along both scaling axes, and Table 1 reports each benchmark at L=1 and L=9. At every tested depth and width, ReM-MoA and ReM-MoA* outperform all baselines on each of the five benchmarks (full tables in Appendix B). In Table 1, the lead of ReM-MoA over AttentionMoA grows from L=1 to L=9 on every benchmark, most on CRUX ($+1.0 \rightarrow +6.3$) and Formal Logic ($+0.8 \rightarrow +5.4$) and least on HellaSwag ($+0.7 \rightarrow +2.7$). From L=3 onward, the lead over the nearest baseline exceeds one point and both five-seed standard deviations on every benchmark (Table 6).

The gap widens with depth. Figure 3(a) shows that the lead of ReM-MoA over the strongest baseline grows from $+0.86$ at L=1 to $+4.54$ at L=9, a $5.3\times$ expansion. Against Standard MoA, the lead grows from $+2.48$ to $+17.34$ ($7.0\times$). Figure 3(b) shows a smaller expansion under width scaling, with the lead over the strongest baseline rising from $+2.24$ at N=3 to $+3.20$ at N=8.

Distillation extends sustained scaling to every benchmark. ReM-MoA* matches or exceeds ReM-MoA in every configuration of the full tables. In Table 1, ReM-MoA* exceeds ReM-MoA at L=9 by 3.3 points on Formal Logic and 3.0 points on MATH. Distillation also lifts the depth trend on MMLU-redux and HellaSwag from plateau to sustain.

In answer to RQ1, ReM-MoA sustains depth scaling on MATH, Formal Logic, and CRUX while ReM-MoA* sustains on all five benchmarks. Under width scaling, ReM-MoA and ReM-MoA* lead at every width with a margin that widens from N=3 to N=8.

5.3 Compute and Proposer-Scale Controls (RQ2)

The advantage does not come from extra tokens. Figure 3(c) plots mean accuracy against per-query tokens under depth scaling, and Table 8 in Appendix C.1 lists the values. Scaling depth from L=1 to L=9, ReM-MoA gains 1.3 points per 10k additional tokens, nearly three times the rate of the most efficient baseline (RMoA, 0.45). AttentionMoA gains 0.38 points per 10k tokens, and Standard MoA loses accuracy over the same range. At every depth, ReM-MoA also uses fewer tokens than RMoA and AttentionMoA, and AttentionMoA consumes 2.1–2.3$\times$ the tokens of ReM-MoA. The distilled Reviewer of ReM-MoA* adds less than 2% tokens for 2.4 more points than ReM-MoA at L=9.

Compute-matched self-consistency does not close the gap. Figure 3(c) also shows SC over each proposer model, and SC over Qwen2.5-Coder-7B is the strongest at every budget. According to Table 8, the strongest SC gains 1.0 point from 27k to 88k tokens while ReM-MoA gains 4.9 points. At 88k tokens, ReM-MoA leads the best SC by 9.3 points (72.8 vs. 63.5). The lead over the best SC

Table 1: Depth scaling summary (N=3). For each benchmark, the columns give the result at L=1 and L=9 (accuracy, pass@1 for CRUX). The marker after each L=9 result classifies the unrounded gain from L=7 to L=9 as (1) degrade $\downarrow$ below -0.5, (2) plateau $\rightarrow$ up to $+0.2$, (3) saturate $\uparrow\rightarrow$ up to $+0.5$, or (4) sustain $\uparrow\uparrow$ above $+0.5$. The Avg. columns average the five benchmarks. Ablations use the off-the-shelf Reviewer. Entries are five-seed means with standard deviations up to 1.6 (Appendices B and D). **Bold** and underline mark the best and second-best method.

Method	MATH		MMLU-redux		Formal Logic		CRUX		HellaSwag		Avg.	
	L=1	L=9	L=1	L=9	L=1	L=9	L=1	L=9	L=1	L=9	L=1	L=9
Standard MoA	65.0	61.0$\downarrow$	61.5	52.4$\downarrow$	45.1	42.2$\downarrow$	54.0	51.0$\downarrow$	73.5	70.8$\downarrow$	59.8	55.5
RMoA	66.0	71.5$\rightarrow$	63.8	64.4$\rightarrow$	46.0	50.6$\rightarrow$	54.5	59.5$\rightarrow$	74.0	77.6$\rightarrow$	60.9	64.7
AttentionMoA	67.0	76.9$\uparrow\rightarrow$	63.2	64.7$\rightarrow$	46.7	55.7$\uparrow\uparrow$	55.5	63.9$\uparrow\rightarrow$	74.8	80.2$\rightarrow$	61.4	68.3
ReM-MoA	<u>68.0</u>	<u>81.0</u>$\uparrow\uparrow$	<u>64.0</u>	<u>68.9</u>$\rightarrow$	**47.5**	<u>61.1</u>$\uparrow\uparrow$	<u>56.5</u>	<u>70.2</u>$\uparrow\uparrow$	<u>75.5</u>	<u>82.9</u>$\rightarrow$	<u>62.3</u>	<u>72.8</u>
ReM-MoA*	**68.3**	**84.0**$\uparrow\uparrow$	**64.2**	**70.2**$\uparrow\uparrow$	**47.5**	**64.4**$\uparrow\uparrow$	**57.0**	**73.0**$\uparrow\uparrow$	**75.8**	**84.3**$\uparrow\uparrow$	**62.6**	**75.2**
Ablations of ReM-MoA												
− cross-layer access	68.0	76.2$\rightarrow$	64.0	66.8$\rightarrow$	47.5	55.7$\rightarrow$	56.5	64.5$\rightarrow$	75.5	80.3$\rightarrow$	62.3	68.7
− routing diversification	68.0	77.7$\uparrow\rightarrow$	64.0	67.6$\rightarrow$	47.5	57.0$\rightarrow$	56.5	66.9$\uparrow\rightarrow$	75.5	81.6$\rightarrow$	62.3	70.2
− rationale	67.7	79.3$\uparrow\uparrow$	63.9	68.0$\rightarrow$	46.7	58.6$\uparrow\uparrow$	56.1	68.0$\uparrow\uparrow$	75.3	82.2$\rightarrow$	61.9	71.2

widens monotonically from $+1.6$ at 7.5k tokens, so additional sampling alone does not reproduce the depth gain of ReM-MoA.

The advantage persists with stronger proposers. Table 10 and Figure 7 in Appendix C.2 repeat depth scaling with 14B and 32B proposer pools. At both scales, ReM-MoA leads AttentionMoA at L=9 by 4.5 to 5.2 points and gains more from depth than with the 7–8B pool ($+13.0$ at 14B and $+12.7$ at 32B vs. $+10.5$). Standard MoA still degrades and RMoA still plateaus, while AttentionMoA gains about half as much as ReM-MoA from L=7 to L=9 (0.6 and 0.8 vs. 1.2 and 1.3). The advantage of ReM-MoA therefore does not rely on weak proposers.

In answer to RQ2, ReM-MoA gains more accuracy per additional token than every baseline and outperforms compute-matched SC, so the advantage of ReM-MoA does not rely on additional compute. The lead of ReM-MoA also holds from 7–8B to 32B proposers, so this advantage does not rely on proposer scale either.

5.4 Ablation Studies (RQ3)

We ablate three components of ReM-MoA independently. − *cross-layer access* restricts every memory read to the previous-layer entry $\mathcal{M}_{l-1}$. − *routing diversification* gives all N proposers of a layer one shared reference set. − *rationale* removes the Reviewer rationales from memory entries. All variants use the off-the-shelf Reviewer, and the lower block of Table 1 reports the variants at L=1 and L=9.

Cross-layer access and routing drive sustained scaling. In the Avg. columns of Table 1, all three variants start within 0.4 points of ReM-MoA at L=1 but fall behind by L=9. Removing cross-layer access costs the most, lowering the five-benchmark mean at L=9 by 4.12 points (72.8 $\rightarrow$ 68.7) with a peak drop of 5.7 points on CRUX. At L=9, the full ReM-MoA leads AttentionMoA by 4.5 points, whereas the variant without cross-layer access leads by only 0.4 points. Removing routing diversification costs 2.66 points (72.8 $\rightarrow$ 70.2) and turns MATH and CRUX from sustain to saturate and Formal Logic from sustain to plateau. The memory itself is unchanged in this variant, so the 2.66-point drop comes from giving all proposers the same references. Removing rationales costs only 1.60 points (72.8 $\rightarrow$ 71.2) and preserves every trend, so rationales refine accuracy rather than enable scaling. From L=5 onward, the gap between ReM-MoA and each variant exceeds both five-seed standard deviations (Table 11 in Appendix D).

Routing preserves measured diversity. Figure 3(d) plots the mean Vendi Score (Friedman & Dieng, 2023) of the N=3 proposer outputs with one-standard-deviation bands, and Table 12 in Appendix D.2 lists the values. From L=1 to L=9, the Vendi Score of ReM-MoA decreases only from 2.52 to 1.85 (ReM-MoA*: 1.90), whereas every baseline falls to 1.35 or below. The retained diversity is consistent with the routes of Equation (7), which give each proposer a structurally different view of

Table 2: Reviewer distillation (N=3) on the five distillation domains. For each domain, the columns give the result at L=1 and L=9 (accuracy, pass@1 for MBPP) with trend markers as in Table 1. For each domain, ReM-MoA* (held-out) uses a Reviewer distilled only on the other four domains. Recovered is computed at L=9 as the gain of ReM-MoA* (held-out) over ReM-MoA divided by the gain of ReM-MoA* over ReM-MoA, and the Avg. entry divides the mean gains. Entries are five-seed means (Appendix E). **Bold** and underline mark the best and second-best Reviewer.

Method	MATH		ARC-Challenge		MedQA		MBPP		HellaSwag		Avg.	
	L=1	L=9	L=1	L=9	L=1	L=9	L=1	L=9	L=1	L=9	L=1	L=9
ReM-MoA	68.0	81.0↑↑	86.8	92.7→	44.8	60.5↑↑	83.0	88.9↑→	<u>75.5</u>	82.9→	71.6	81.2
ReM-MoA* (held-out)	<u>68.2</u>	<u>83.0</u>↑↑	<u>86.9</u>	<u>93.1</u>→	<u>45.0</u>	<u>61.4</u>↑↑	<u>83.1</u>	<u>89.5</u>↑→	<u>75.5</u>	<u>83.2</u>→	<u>71.7</u>	<u>82.0</u>
ReM-MoA*	**68.3**	**84.0**↑↑	**87.0**	**93.5**→	**45.2**	**62.2**↑↑	**83.8**	**91.6**↑→	**75.8**	**84.3**↑↑	**72.0**	**83.1**
Recovered (%)	67		50		53		22		21		44	

the memory. Appendix D.2 details the embedding protocol and the limits of this measurement, and Appendix F traces two problems through all four methods at L=9.

In answer to RQ3, the Ranked Reasoning Memory contributes the largest share through cross-layer access (4.12 points at L=9) and a smaller share through rationales (1.60 points). The Curated Diversified Memory Routing contributes 2.66 points and retains proposer diversity as depth grows.

5.5 Reviewer Distillation (RQ4)

We compare three Reviewers on the five distillation domains, namely (1) MATH, (2) ARC-Challenge (Clark et al., 2018), (3) MedQA (Jin et al., 2021), (4) MBPP (Austin et al., 2021), and (5) HellaSwag. ReM-MoA uses the off-the-shelf Reviewer, and ReM-MoA* uses the Reviewer distilled on all five domains. ReM-MoA* (held-out) uses, for each evaluated domain, a Reviewer distilled only on the other four domains. Table 2 reports the three Reviewers at L=1 and L=9, and Figure 9 and Table 13 in Appendix E report every depth.

Distillation gains grow with depth. In the Avg. columns of Table 2, the gain of ReM-MoA* over ReM-MoA grows from +0.40 points at L=1 to +1.92 points at L=9. At L=9, ReM-MoA* exceeds ReM-MoA in every domain, most on MATH (+3.0) and least on ARC-Challenge (+0.8). On HellaSwag, distillation also lifts the depth trend from plateau to sustain.

Distillation transfers partially to held-out domains. The Recovered row of Table 2 shows that ReM-MoA* (held-out) recovers 44% of the mean gain of ReM-MoA* over ReM-MoA at L=9 (+0.84 of +1.92 points) without seeing the evaluated domain. Recovery ranges from 67% on MATH to 21% on HellaSwag, and ReM-MoA* (held-out) keeps every depth trend of ReM-MoA. The held-out Reviewer trains on 400 instead of 500 problems, so part of the remaining gap to ReM-MoA* may reflect data volume. The partial transfer is consistent with the gains of ReM-MoA* in Table 1 on MMLU-redux, Formal Logic, and CRUX, none of which is a distillation domain.

In answer to RQ4, multi-domain distillation improves ReM-MoA in every domain with gains that grow with depth. The distilled Reviewer also transfers partially to held-out domains and recovers 44% of the in-domain gain on average.

6 Conclusion

We presented ReM-MoA, a Mixture-of-Agents framework that sustains scaling through a Ranked Reasoning Memory and a Curated Diversified Memory Routing scheme, jointly preserving quality and diversity across layers. Across five reasoning benchmarks, ReM-MoA outperforms all baselines under both depth and width scaling, and the lead widens as depth grows. Under depth scaling, ReM-MoA gains nearly three times as much accuracy per additional token as any baseline, exceeds compute-matched self-consistency by 9.3 points at the largest budget, and keeps the lead across multiple model sizes. An optional multi-domain Reviewer distillation further extends sustained scaling to all benchmarks and transfers partially to held-out domains, establishing structured cross-layer memory as a critical missing mechanism for scalable MoA reasoning.

AI USE STATEMENT

In this work, we used generative AI tools for limited language editing, paraphrasing, and polishing of the manuscript text. We have not used generative AI tools for research ideation, experimental design, implementation, data analysis, or the generation of any technical content, claims, or results. We have reviewed all AI-assisted work: all AI-edited text was checked and verified by the authors, and all technical content, analysis, experimental results, and conclusions were developed and verified 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 does not involve human subjects, personally identifiable information, or newly collected data. All experiments use publicly available benchmarks under their respective licenses. ReM-MoA inherits the biases and potential failure modes of its constituent LLMs, including known judge biases of the LLM-based Reviewer Agent. Our comparative scoring and contrastive routing reduce but do not eliminate these effects, and deployment in high-stakes settings would require additional safeguards. Beyond the added inference cost of multi-agent generation, we do not foresee direct negative societal impacts specific to this work.

REPRODUCIBILITY STATEMENT

Our code is available in an anonymous repository at `https://anonymous.4open.science/r/ReM-MoA-64A8/README.md`. Section 5.1 describes the complete experimental setup, including benchmarks, the proposer pool and aggregator, and baselines. Appendix A details (1) the Reviewer distillation pipeline with trace generation and teacher labeling, (2) LoRA configuration, (3) training hyperparameters, (4) train/test isolation, (5) inference settings, and (6) the routing distributions for all widths (Appendix A.7). Appendix G provides the scoring rubric of the Reviewer and the prompt templates of the proposer agents, the Reviewer Agent, the aggregator, and the teacher model.

REFERENCES

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, and Charles Sutton. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*, 2021.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try ARC, the AI2 reasoning challenge. *arXiv preprint arXiv:1803.05457*, 2018.

DeepSeek-AI. DeepSeek-R1: Incentivizing reasoning capability in LLMs via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

Yilun Du, Shuang Li, Antonio Torralba, Joshua B. Tenenbaum, and Igor Mordatch. Improving factuality and reasoning in language models through multiagent debate. In *Proceedings of the 41st International Conference on Machine Learning (ICML)*, 2024.

Dan Friedman and Adji Bousso Dieng. The Vendi Score: A diversity evaluation metric for machine learning. *Transactions on Machine Learning Research*, 2023.

Aryo Pradipta Gema, Joshua Ong Jun Leang, Giwon Hong, Alessio Devoto, Alberto Carlo Maria Mancino, Rohit Saxena, Xuanli He, Yuhao Zhao, Xiaotang Du, Mohammad Reza Ghasemi Madani, Claire Barale, et al. Are we done with MMLU? 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. 5069–5096, 2025.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, et al. The Llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Alex Gu, Baptiste Rozière, Hugh Leather, Armando Solar-Lezama, Gabriel Synnaeve, and Sida I. Wang. CRUXEval: A benchmark for code reasoning, understanding and execution. *arXiv preprint arXiv:2401.03065*, 2024.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*, 2020.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the MATH dataset. *arXiv preprint arXiv:2103.03874*, 2021.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Liang Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations (ICLR)*, 2022.

Binyuan Hui, Jian Yang, Zeyu Cui, Jiaxi Yang, Dayiheng Liu, Lei Zhang, Tianyu Liu, Jiajun Zhang, Bowen Yu, Keming Lu, Kai Dang, et al. Qwen2.5-Coder technical report. *arXiv preprint arXiv:2409.12186*, 2024.

Di Jin, Eileen Pan, Nassim Oufattole, Wei-Hung Weng, Hanyi Fang, and Peter Szolovits. What disease does this patient have? a large-scale open domain question answering dataset from medical exams. *Applied Sciences*, 11(14):6421, 2021.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with PagedAttention. In *Proceedings of the 29th Symposium on Operating Systems Principles (SOSP)*, pp. 611–626, 2023.

Wenzhe Li, Yong Lin, Mengzhou Xia, and Chi Jin. Rethinking mixture-of-agents: Is mixing different large language models beneficial? *arXiv preprint arXiv:2502.00674*, 2025.

Tian Liang, Zhiwei He, Wenxiang Jiao, Xing Wang, Yan Wang, Rui Wang, Yujiu Yang, Shuming Shi, and Zhaopeng Tu. Encouraging divergent thinking in large language models through multi-agent debate. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 17889–17904, 2024.

Hanjun Luo, Shenyu Dai, Chiming Ni, Xinfeng Li, Guibin Zhang, Kun Wang, Tongliang Liu, and Hanan Salam. AgentAuditor: Human-level safety and security evaluation for LLM agents. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 38, pp. 43241–43298, 2026.

Siru Ouyang, Jun Yan, I. Hsu, Yanfei Chen, Ke Jiang, Zifeng Wang, Rujun Han, Long T. Le, Shaunak Daruki, Xinyang Tang, Vidisha Tirumalashetty, et al. ReasoningBank: Scaling agent self-evolving with reasoning memory. *arXiv preprint arXiv:2509.25140*, 2025.

Heng Ping, Arijit Bhattacharjee, Peiyu Zhang, Shixuan Li, Wei Yang, Anzhe Cheng, Xiaole Zhang, Jesse Thomason, Ali Jannesari, Nesreen Ahmed, and Paul Bogdan. VeriMoA: A mixture-of-agents framework for spec-to-HDL generation. *arXiv preprint arXiv:2510.27617*, 2025.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 36, pp. 8634–8652, 2023.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling LLM test-time compute optimally can be more effective than scaling model parameters. *arXiv preprint arXiv:2408.03314*, 2024.

Junlin Wang, Jue Wang, Ben Athiwaratkun, Ce Zhang, and James Y. Zou. Mixture-of-agents enhances large language model capabilities. In *International Conference on Learning Representations (ICLR)*, pp. 33944–33963, 2025.

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. *arXiv preprint arXiv:2203.11171*, 2022.

Jianyu Wen, Yang Wei, Xiongxi Yu, Changxuan Xiao, and Ke Zeng. Attention-MoA: Enhancing mixture-of-agents via inter-agent semantic attention and deep residual synthesis. *arXiv preprint arXiv:2601.16596*, 2026.

Zhentao Xie, Chengcheng Han, Jinxin Shi, Wenjun Cui, Wayne Xin Zhao, Xingjiao Wu, and Jiabao Zhao. RMoA: Optimizing mixture-of-agents through diversity maximization and residual compensation. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 6575–6602, 2025.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, et al. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2024.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. HellaSwag: Can a machine really finish your sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 4791–4800, 2019.

Kexun Zhang, Weiran Yao, Zuxin Liu, Yihao Feng, Zhiwei Liu, Rithesh Ramapura Narasimha Murthy, Tian Lan, Lei Li, Renze Lou, Jiacheng Xu, Bo Pang, et al. Diversity empowers intelligence: Integrating expertise of software engineering agents. In *International Conference on Learning Representations (ICLR)*, pp. 86691–86714, 2025.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, et al. Judging LLM-as-a-Judge with MT-Bench and chatbot arena. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 36, pp. 46595–46623, 2023.

Lianghui Zhu, Xinggang Wang, and Xinlong Wang. JudgeLM: Fine-tuned large language models are scalable judges. In *International Conference on Learning Representations (ICLR)*, pp. 51257–51296, 2025.

A Experimental Setup Details

This appendix expands the configuration referenced in Section 5.1, covering (1) the Reviewer distillation pipeline, (2) LoRA configuration, (3) training hyperparameters, (4) train/test isolation, (5) inference-time settings, and (6) the routing distributions for larger widths.

A.1 Distillation Variants

We evaluate three Reviewer configurations:

- **ReM-MoA** uses the off-the-shelf Qwen2.5-7B-Instruct Reviewer without fine-tuning.
- **ReM-MoA*** uses a Reviewer LoRA-distilled jointly on five domains (MATH, ARC-Challenge, MedQA, MBPP, HellaSwag, 100 problems each, 500 total).
- **ReM-MoA*** **(held-out)** uses five Reviewers, each LoRA-distilled on four of the five domains (400 problems per Reviewer) and evaluated on the held-out domain (Section 5.5).

A.2 Trace Generation and Teacher Labeling

For each training problem, we run ReM-MoA at $L{=}3$, $N{=}3$ to collect the proposer traces produced at every layer, yielding $\sim$9 traces per problem and $1{,}500$ (problem, layer-traces) tuples over the 500 training problems. Each (problem, traces) tuple is then sent to a frontier-model teacher, GPT-5.5, which receives the same inputs as the Reviewer (the problem and the proposer traces) and is constrained by a structured prompt (Appendix G.5) to emit labels conforming to the Reviewer's output schema (an integer score and an independent rationale for each trace). The teacher receives no ground-truth answer. Traces are filtered by length (50 to 2,000 tokens) before training to remove degenerate generations.

A.3 LoRA Configuration

LoRA (Hu et al., 2022) is applied to the attention projection matrices of Qwen2.5-7B-Instruct.

Table 3: LoRA configuration.

Rank (r)	16
LoRA alpha	32
Dropout	0.05
Target modules	q_proj, k_proj, v_proj, o_proj
Trainable parameters	$\sim$10.1M (0.13% of 7.6B)

A.4 Training Hyperparameters

Table 4: Training hyperparameters for Reviewer distillation.

Optimizer	AdamW
Learning rate	2×10^{-4} (cosine decay)
Warmup ratio	0.05
Effective batch size	8 (per-device 4 $\times$ grad-accum 2)
Max sequence length	8,192
Epochs	3
Precision	bf16
Weight decay	0.01
Loss	Causal LM cross-entropy

Each distillation run completes in approximately 2 hours on a single NVIDIA A100 (80GB) GPU. The full distillation pipeline (the ReM-MoA* Reviewer plus five held-out Reviewers, Appendix A.1) requires approximately 12 A100-hours of training compute in total.

A.5 Train/Test Isolation

Fine-tuning data is drawn exclusively from each dataset's publicly available training split, while evaluation uses the held-out test split. For HellaSwag, we evaluate on the validation split since test labels are not public, following standard practice. Because train and test problems are disjoint by construction, label leakage from the GPT-5.5 teacher into evaluation is not possible at the problem level. The structured evaluation prompt (Appendix G.5) further constrains the teacher's output to fixed dimensions (score and rationale), focusing the student on the meta-skill of judging reasoning quality rather than memorizing canonical solutions. The held-out results in Section 5.5 further show that the distilled Reviewer transfers partially to domains excluded from the distillation data.

A.6 Inference and Hardware

All experiments run on NVIDIA A100 (80GB) GPUs via vLLM (Kwon et al., 2023). Proposer agents sample at T=0.8 (top-p 0.95), the Reviewer at T=0.3 (top-p 0.9), and the aggregator at T=0.5 (top-p 0.9). Maximum new tokens per call: 2,048 for proposers and aggregator, 8,192 for the Reviewer (which must emit N score-rationale pairs in a single response).

A.7 Routing Distributions for Larger Widths

Section 4.2 introduces the routing scheme and gives one concrete instance for N=3 in Equation 7. Table 5 extends this assignment to $N \in \{4, \ldots, 8\}$, the range used in our width-scaling experiments (Table 7). The design principle is to keep the three route families (contrastive, positive reinforcement, failure-mode focus) represented at every layer while scaling the size of each route with N, so that the reference sets cover more of the memory pool while remaining mutually distinct.

Table 5: Routing function assignments $\Pi_1, \ldots, \Pi_N$ for each width N. Con_n denotes $\text{Top}_a \cup \text{Bot}_b$ with $a + b = n$, $a = \lceil n/2 \rceil$, $b = \lfloor n/2 \rfloor$.

N	Route assignment
3	$\text{Con}_2, \text{Top}_2, \text{Bot}_2$
4	$\text{Con}_2, \text{Con}_3, \text{Top}_2, \text{Bot}_2$
5	$\text{Con}_2, \text{Con}_3, \text{Top}_2, \text{Top}_3, \text{Bot}_2$
6	$\text{Con}_2, \text{Con}_3, \text{Top}_2, \text{Top}_3, \text{Bot}_2, \text{Bot}_3$
7	$\text{Con}_2, \text{Con}_3, \text{Con}_4, \text{Top}_2, \text{Top}_3, \text{Bot}_2, \text{Bot}_3$
8	$\text{Con}_2, \text{Con}_3, \text{Con}_4, \text{Top}_2, \text{Top}_3, \text{Top}_4, \text{Bot}_2, \text{Bot}_3$

Handling sparse early layers. At layer l, the global pool $\mathcal{P}_l = \mathcal{M}_{\leq l-1}$ contains $N \cdot (l-1)$ entries. Since every route size n satisfies $n \leq N$ in all configurations above, the pool already has at least N entries at l=2, so no route is ever truncated in practice.

B Full Scaling Results

Tables 6 and 7 report per-benchmark depth and width scaling, which Table 1 and Figure 3 summarize in Section 5.2. Figures 4 and 5 plot the same data. The Formal Logic test split contains 126 questions, so one question corresponds to 0.79 accuracy points on Formal Logic. Differences that the main text reports with two decimals come from unrounded means and can therefore differ slightly from differences of the rounded entries in Appendices B to E.

Table 6: Per-benchmark depth scaling (N=3). Entries are mean±std over five seeds (accuracy in %, pass@1 for CRUX). Trend markers follow Table 1 on the late-layer gain from L=7 to L=9. Our methods are in **bold**.

Benchmark	Method	L=1	L=2	L=3	L=4	L=5	L=6	L=7	L=8	L=9	Trend
MATH	Standard MoA	65.0±0.4	68.5±0.5	70.0±0.5	69.2±0.6	67.8±0.6	66.0±0.6	64.2±0.8	62.5±0.8	61.0±0.9	↓
	RMoA	66.0±0.4	69.5±0.5	71.0±0.6	71.8±0.5	72.0±0.5	72.0±0.5	71.8±0.5	71.6±0.6	71.5±0.7	→
	AttentionMoA	67.0±0.4	71.0±0.4	73.0±0.4	74.5±0.3	75.5±0.4	76.2±0.5	76.6±0.5	76.8±0.5	76.9±0.5	↑→
	ReM-MoA	**68.0±0.4**	**72.5±0.5**	**75.0±0.4**	**77.0±0.6**	**78.5±0.4**	**79.5±0.4**	**80.2±0.6**	**80.7±0.6**	**81.0±0.8**	↑↑
	ReM-MoA*	**68.3±0.4**	**73.5±0.3**	**76.8±0.4**	**79.2±0.3**	**81.0±0.4**	**82.2±0.4**	**83.0±0.5**	**83.7±0.6**	**84.0±0.6**	↑↑
MMLU-redux	Standard MoA	61.5±0.4	59.8±0.4	58.2±0.4	56.9±0.4	55.8±0.4	54.8±0.5	53.9±0.6	53.1±0.6	52.4±0.8	↓
	RMoA	63.8±0.3	65.1±0.3	65.4±0.4	65.3±0.5	65.2±0.5	65.0±0.5	64.8±0.6	64.6±0.5	64.4±0.4	→
	AttentionMoA	63.2±0.2	64.2±0.4	64.8±0.2	65.0±0.3	65.1±0.3	65.0±0.3	64.9±0.4	64.8±0.4	64.7±0.6	→
	ReM-MoA	**64.0±0.4**	**66.0±0.3**	**67.0±0.3**	**67.8±0.3**	**68.3±0.5**	**68.6±0.4**	**68.8±0.5**	**68.9±0.5**	**68.9±0.5**	→
	ReM-MoA*	**64.2±0.2**	**66.3±0.2**	**67.4±0.2**	**68.3±0.2**	**68.9±0.3**	**69.3±0.4**	**69.6±0.3**	**69.9±0.4**	**70.2±0.4**	↑↑
Formal Logic	Standard MoA	45.1±0.9	47.5±1.0	49.0±1.0	48.3±1.0	46.7±1.2	45.9±1.2	44.3±1.3	43.0±1.3	42.2±1.6	↓
	RMoA	46.0±0.8	48.3±1.0	49.8±0.8	50.6±0.8	50.6±0.8	50.8±0.9	50.6±1.0	50.6±1.0	50.6±1.0	→
	AttentionMoA	46.7±0.6	49.8±0.9	51.6±0.9	53.3±0.8	54.0±0.8	54.9±0.9	54.9±1.0	55.7±0.8	55.7±0.9	↑↑
	ReM-MoA	**47.5±0.9**	**51.7±0.8**	**54.1±0.9**	**56.3±0.9**	**57.8±0.9**	**59.4±1.0**	**60.3±0.9**	**60.2±1.1**	**61.1±1.1**	↑↑
	ReM-MoA*	**47.5±1.0**	**52.5±0.8**	**55.6±0.9**	**58.6±0.7**	**60.2±0.8**	**61.7±0.9**	**63.5±1.0**	**64.3±0.7**	**64.4±1.0**	↑↑
CRUX	Standard MoA	54.0±0.5	56.5±0.6	57.5±0.6	57.0±0.6	56.0±0.6	54.8±0.6	53.5±0.8	52.2±0.8	51.0±1.0	↓
	RMoA	54.5±0.4	57.5±0.5	58.5±0.3	59.0±0.5	59.2±0.5	59.4±0.6	59.4±0.7	59.5±0.6	59.5±0.5	→
	AttentionMoA	55.5±0.4	58.8±0.4	60.5±0.4	61.8±0.4	62.7±0.5	63.3±0.5	63.6±0.5	63.8±0.6	63.9±0.5	↑→
	ReM-MoA	**56.5±0.5**	**60.5±0.6**	**63.0±0.5**	**65.2±0.5**	**67.0±0.6**	**68.5±0.7**	**69.5±0.6**	**70.0±0.7**	**70.2±0.8**	↑↑
	ReM-MoA*	**57.0±0.3**	**61.6±0.2**	**64.6±0.4**	**67.2±0.4**	**69.3±0.4**	**71.1±0.5**	**72.2±0.6**	**72.8±0.5**	**73.0±0.6**	↑↑
HellaSwag	Standard MoA	73.5±0.3	76.0±0.2	76.8±0.4	76.2±0.4	75.3±0.3	74.2±0.5	73.0±0.5	71.8±0.4	70.8±0.5	↓
	RMoA	74.0±0.4	76.8±0.3	77.5±0.3	77.8±0.3	78.0±0.5	77.9±0.2	77.8±0.4	77.7±0.3	77.6±0.4	→
	AttentionMoA	74.8±0.4	77.5±0.3	78.5±0.5	79.2±0.3	79.7±0.3	80.0±0.3	80.1±0.2	80.2±0.3	80.2±0.3	→
	ReM-MoA	**75.5±0.3**	**78.8±0.2**	**80.5±0.5**	**81.5±0.3**	**82.2±0.5**	**82.6±0.3**	**82.8±0.4**	**82.9±0.6**	**82.9±0.3**	→
	ReM-MoA*	**75.8±0.2**	**79.5±0.2**	**81.2±0.3**	**82.2±0.3**	**83.0±0.4**	**83.4±0.3**	**83.7±0.2**	**84.0±0.3**	**84.3±0.4**	↑↑

C Compute and Proposer-Scale Controls

C.1 Token Cost and Compute-Matched Self-Consistency

We count compute as all input and output tokens over the LLM calls of a query and average the count over the five benchmarks. For self-consistency (SC), each proposer model samples K responses independently at the proposer temperature of Appendix A.6 and returns the majority answer. We set K so that the token consumption of SC matches the budget of ReM-MoA at each $L \in \{1, 3, 5, 7, 9\}$. Table 8 reports depth scaling together with the three SC controls, and Table 9 reports width scaling. Figure 6 plots accuracy against tokens along both axes, and Section 5.3 discusses the results.

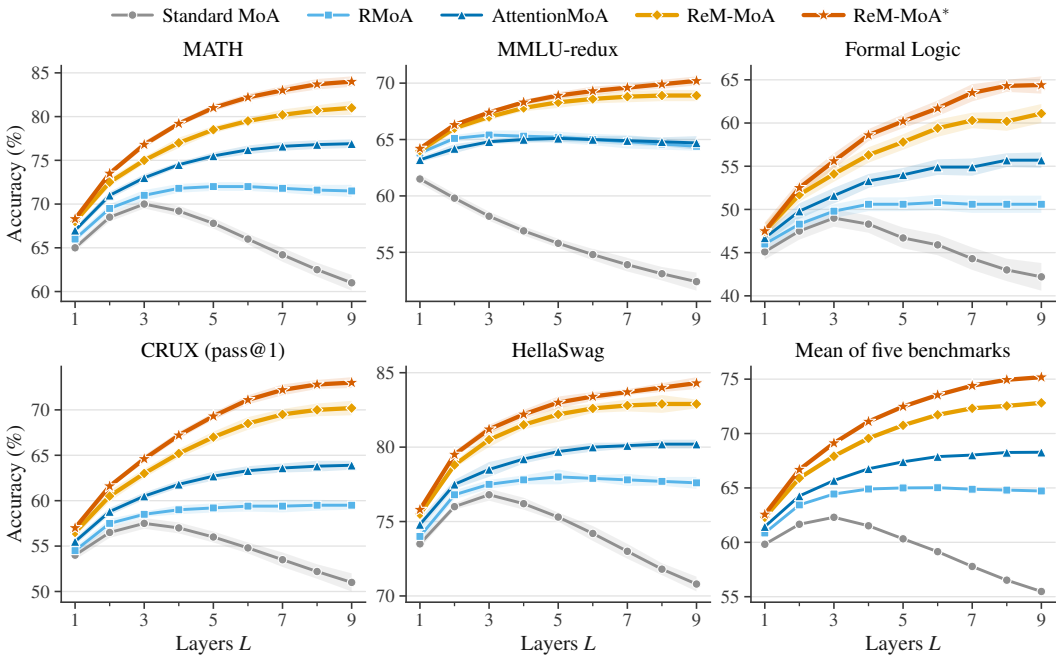

Figure 4: Per-benchmark depth scaling (N=3) with the values of Table 6. Shaded bands show one standard deviation over five seeds, and the last panel averages the five benchmarks.

Table 7: Per-benchmark width scaling (L=3). Entries are mean±std over five seeds (accuracy in %, pass@1 for CRUX). Trend markers use the thresholds of Table 1 on the gain from N=6 to N=8. Our methods are in **bold**.

Benchmark	Method	N=3	N=4	N=5	N=6	N=7	N=8	Trend
MATH	Standard MoA	70.0±0.5	70.8±0.6	71.4±0.6	71.8±0.4	72.0±0.4	72.1±0.4	↑→
	RMoA	71.0±0.6	72.0±0.5	72.7±0.3	73.3±0.4	73.6±0.5	73.8±0.4	↑→
	AttentionMoA	73.0±0.4	74.0±0.3	74.8±0.4	75.5±0.4	75.9±0.5	76.2±0.4	↑↑
	ReM-MoA	**75.0±0.4**	**76.2±0.5**	**77.2±0.4**	**78.0±0.4**	**78.5±0.5**	**78.9±0.4**	↑↑
	ReM-MoA*	**76.8±0.4**	**78.0±0.4**	**79.1±0.3**	**79.9±0.4**	**80.5±0.4**	**80.9±0.3**	↑↑
MMLU-redux	Standard MoA	58.2±0.4	57.2±0.4	56.5±0.3	55.8±0.5	55.2±0.3	54.5±0.4	↓
	RMoA	65.4±0.4	65.5±0.2	65.5±0.3	65.4±0.3	65.3±0.2	65.2±0.2	→
	AttentionMoA	64.8±0.2	65.1±0.3	65.2±0.3	65.2±0.3	65.1±0.2	65.0±0.2	→
	ReM-MoA	**67.0±0.3**	**67.8±0.3**	**68.3±0.3**	**68.8±0.3**	**69.0±0.3**	**69.2±0.3**	↑→
	ReM-MoA*	**67.4±0.2**	**68.2±0.3**	**68.7±0.3**	**69.3±0.3**	**69.5±0.3**	**69.8±0.2**	↑↑
Formal Logic	Standard MoA	49.0±1.0	49.0±0.8	50.0±0.8	49.8±0.7	49.8±0.8	49.8±0.8	→
	RMoA	49.8±0.8	50.6±0.8	51.7±0.8	51.7±0.8	52.4±0.8	52.5±0.7	↑↑
	AttentionMoA	51.6±0.9	52.5±0.7	53.3±0.8	54.0±0.8	54.8±0.7	54.9±0.7	↑↑
	ReM-MoA	**54.1±0.9**	**55.7±0.9**	**56.2±0.8**	**57.0±0.9**	**57.8±0.7**	**57.8±0.7**	↑↑
	ReM-MoA*	**55.6±0.9**	**57.0±0.8**	**57.8±0.8**	**59.4±0.8**	**59.4±0.7**	**60.2±0.7**	↑↑
CRUX	Standard MoA	57.5±0.6	58.0±0.6	58.4±0.5	58.6±0.5	58.5±0.4	58.3±0.4	→
	RMoA	58.5±0.3	59.3±0.5	59.8±0.4	60.3±0.4	60.6±0.3	60.7±0.4	↑→
	AttentionMoA	60.5±0.4	61.5±0.3	62.3±0.4	62.9±0.3	63.3±0.4	63.6±0.4	↑↑
	ReM-MoA	**63.0±0.5**	**64.3±0.4**	**65.3±0.4**	**66.2±0.4**	**66.8±0.3**	**67.3±0.4**	↑↑
	ReM-MoA*	**64.6±0.4**	**66.0±0.3**	**67.0±0.3**	**68.0±0.3**	**68.7±0.3**	**69.3±0.4**	↑↑
HellaSwag	Standard MoA	76.8±0.4	77.2±0.3	77.4±0.3	77.5±0.3	77.4±0.4	77.2±0.3	→
	RMoA	77.5±0.3	78.0±0.2	78.4±0.2	78.7±0.3	78.8±0.2	78.9±0.2	→
	AttentionMoA	78.5±0.5	79.2±0.2	79.7±0.3	80.1±0.2	80.3±0.2	80.4±0.2	↑→
	ReM-MoA	**80.5±0.5**	**81.2±0.3**	**81.8±0.3**	**82.3±0.3**	**82.6±0.2**	**82.9±0.3**	↑↑
	ReM-MoA*	**81.2±0.3**	**82.0±0.2**	**82.6±0.2**	**83.1±0.3**	**83.4±0.2**	**83.7±0.2**	↑↑

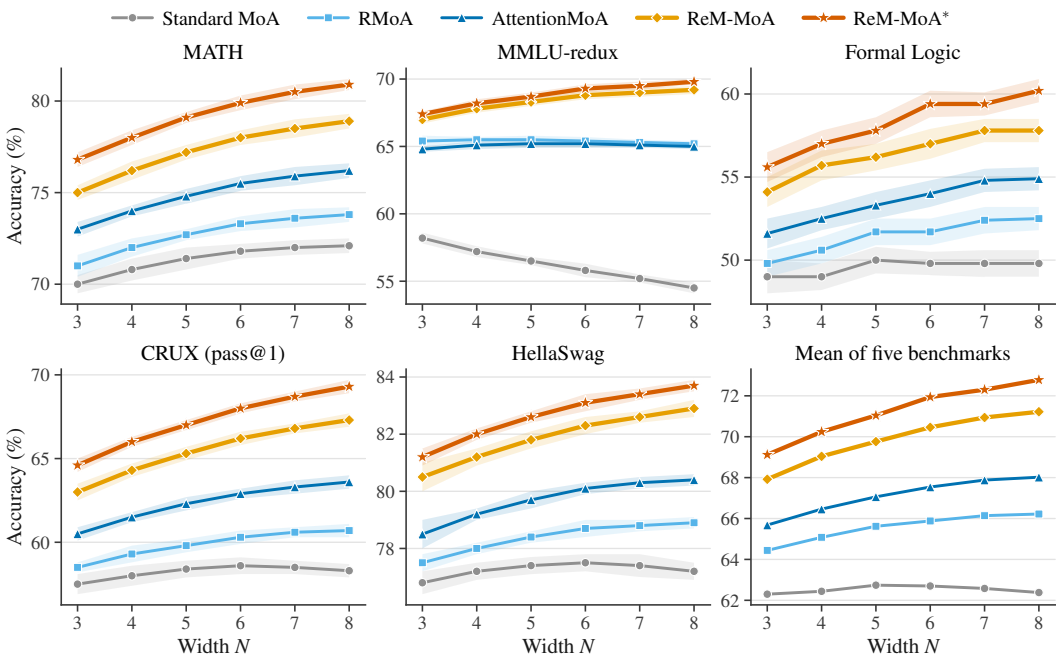

Figure 5: Per-benchmark width scaling (L=3) with the values of Table 7. Shaded bands show one standard deviation over five seeds, and the last panel averages the five benchmarks.

Table 8: Token cost and compute-matched self-consistency under depth scaling (N=3). Accuracy is the five-benchmark mean as mean±std over five seeds. Tokens count all input and output tokens per query in thousands. In each column, the budget of every SC control matches the tokens of ReM-MoA. Our methods are in **bold**.

Method	L=1	L=3	L=5	L=7	L=9
Mean accuracy (%)					
Standard MoA	59.8±0.4	62.3±0.4	60.3±0.4	57.8±0.6	55.5±0.7
RMoA	60.9±0.3	64.4±0.3	65.0±0.4	64.9±0.4	64.7±0.4
AttentionMoA	61.4±0.3	65.7±0.3	67.4±0.3	68.0±0.4	68.3±0.4
ReM-MoA	**62.3±0.4**	**67.9±0.4**	**70.8±0.4**	**72.3±0.4**	**72.8±0.5**
ReM-MoA*	**62.6±0.3**	**69.1±0.3**	**72.5±0.3**	**74.4±0.4**	**75.2±0.4**
SC (Llama-3.1-8B)	54.7±0.4	56.1±0.3	56.6±0.2	56.8±0.2	57.0±0.2
SC (Qwen2.5-7B)	59.8±0.5	61.3±0.3	61.8±0.2	62.1±0.2	62.3±0.2
SC (Qwen2.5-Coder-7B)	60.7±0.3	62.5±0.2	63.0±0.2	63.3±0.2	63.5±0.2
ReM-MoA − best SC	+1.6	+5.4	+7.8	+9.0	+9.3
Tokens per query (k)					
Standard MoA	5.2	19.8	34.5	49.7	65.2
RMoA	7.8	28.6	49.8	71.5	93.7
AttentionMoA	16.1	58.3	102.8	149.5	198.2
ReM-MoA	7.5	26.8	46.5	66.8	87.6
ReM-MoA*	7.6	27.2	47.2	67.8	89.0
Number of SC samples K					
SC (Llama-3.1-8B)	9	34	60	85	112
SC (Qwen2.5-7B)	9	32	56	80	106
SC (Qwen2.5-Coder-7B)	8	30	52	74	97

Table 9: Token cost under width scaling (L=3). Accuracy is the five-benchmark mean as mean±std over five seeds, and tokens count all input and output tokens per query in thousands. Our methods are in **bold**.

Method	N=3	N=4	N=5	N=6	N=7	N=8
Mean accuracy (%)						
Standard MoA	62.3±0.4	62.4±0.4	62.7±0.4	62.7±0.3	62.6±0.3	62.4±0.3
RMoA	64.4±0.3	65.1±0.3	65.6±0.3	65.9±0.3	66.1±0.3	66.2±0.3
AttentionMoA	65.7±0.3	66.5±0.3	67.1±0.3	67.5±0.3	67.9±0.3	68.0±0.3
ReM-MoA	**67.9±0.4**	**69.0±0.3**	**69.8±0.3**	**70.5±0.3**	**70.9±0.3**	**71.2±0.3**
ReM-MoA*	**69.1±0.3**	**70.2±0.3**	**71.0±0.3**	**71.9±0.3**	**72.3±0.3**	**72.8±0.3**
Tokens per query (k)						
Standard MoA	19.8	30.2	42.8	57.5	74.3	93.3
RMoA	28.6	35.2	42.0	49.1	56.4	63.9
AttentionMoA	58.3	86.2	118.7	155.6	196.8	242.4
ReM-MoA	26.8	36.5	47.2	58.8	71.4	84.8
ReM-MoA*	27.2	37.1	47.9	59.7	72.4	86.1

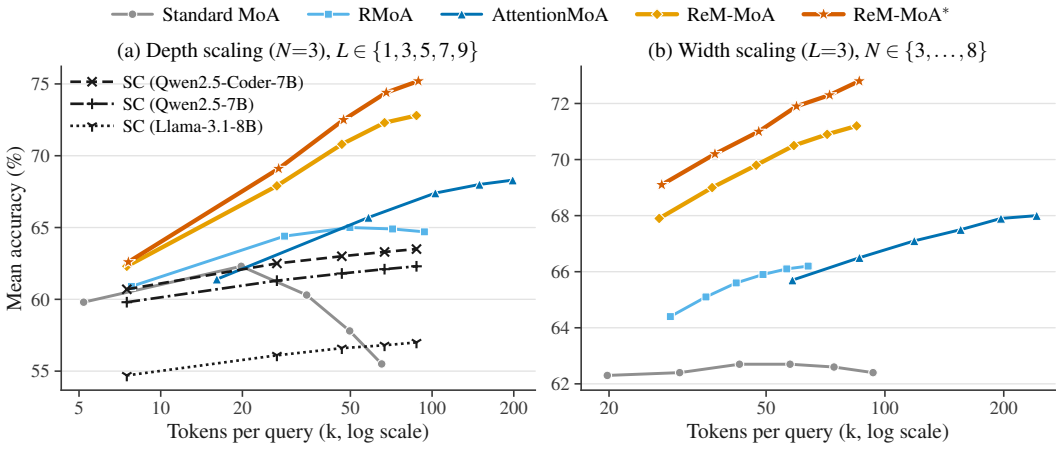

Figure 6: Five-benchmark mean accuracy against tokens per query on a logarithmic axis. (a) Depth scaling at N=3 with the three compute-matched SC controls, whose budgets follow ReM-MoA. (b) Width scaling at L=3. Values follow Tables 8 and 9.

C.2 Larger Proposer Pools

The 14B pool consists of Qwen2.5-14B-Instruct, Qwen2.5-Coder-14B-Instruct, and DeepSeek-R1-Distill-Qwen-14B (DeepSeek-AI, 2025). The 32B pool consists of Qwen2.5-32B-Instruct, Qwen2.5-Coder-32B-Instruct, and DeepSeek-R1-Distill-Qwen-32B. In each pool, the Qwen2.5-Instruct model of matching size serves as the aggregator and as the off-the-shelf Reviewer. All other settings follow Section 5.1. Table 10 and Figure 7 report depth scaling at N=3. Under the thresholds of Table 1, AttentionMoA sustains at both scales. The late-layer gain of AttentionMoA (0.6 at 14B and 0.8 at 32B) is still about half the gain of ReM-MoA (1.2 and 1.3).

Table 10: Depth scaling (N=3) with 14B and 32B proposer pools. Entries are five-benchmark mean accuracy (%) as mean±std over five seeds. Trend markers follow Table 1 on the gain from L=7 to L=9.

Pool	Method	L=1	L=3	L=5	L=7	L=9	Trend
14B	Standard MoA	66.4±0.2	69.3±0.3	67.8±0.4	65.8±0.4	63.8±0.6	↓
	RMoA	67.2±0.2	71.2±0.2	72.3±0.3	72.5±0.5	72.4±0.3	→
	AttentionMoA	68.0±0.4	73.0±0.3	75.0±0.2	76.2±0.2	76.8±0.3	↑↑
	ReM-MoA	**69.0±0.2**	**75.0±0.5**	**78.5±0.3**	**80.8±0.4**	**82.0±0.5**	↑↑
32B	Standard MoA	72.4±0.2	75.0±0.2	73.8±0.3	72.2±0.4	70.8±0.5	↓
	RMoA	73.2±0.2	76.8±0.2	78.0±0.3	78.5±0.2	78.5±0.2	→
	AttentionMoA	74.0±0.2	78.5±0.2	80.8±0.3	82.2±0.4	83.0±0.4	↑↑
	ReM-MoA	**74.8±0.2**	**80.5±0.2**	**84.0±0.3**	**86.2±0.4**	**87.5±0.3**	↑↑

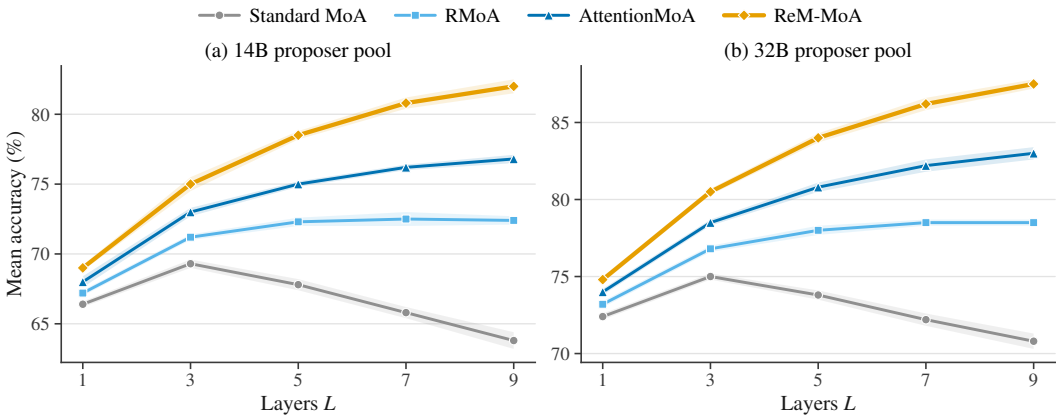

Figure 7: Depth scaling (N=3) of the five-benchmark mean with (a) the 14B pool and (b) the 32B pool. Shaded bands show one standard deviation over five seeds, and Table 10 lists the values.

D Ablation and Diversity

D.1 Per-Benchmark Ablation

Table 11 and Figure 8 report the three ablations of Section 5.4 at every depth. All variants use the off-the-shelf Reviewer.

D.2 Diversity Measurement Protocol

For each query and each layer, we embed the N=3 proposer outputs with all-MiniLM-L6-v2 and form their cosine similarity matrix $\mathbf{G} \in \mathbb{R}^{N\times N}$. The Vendi Score (Friedman & Dieng, 2023) is $\exp\big(-\sum_{i=1}^{N} \lambda_i \log \lambda_i\big)$, where λ_i are the eigenvalues of $\mathbf{G}/N$. The score equals 1 for identical outputs and N for orthogonal outputs, so the score reads as the effective number of distinct outputs. We average the score over the queries of each benchmark and then over the five benchmarks. The

Table 11: Per-benchmark ablation under depth scaling (N=3) with the off-the-shelf Reviewer. Entries are mean±std over five seeds (accuracy in %, pass@1 for CRUX). The **ReM-MoA (full)** rows repeat Table 6, and trend markers follow Table 1.

Benchmark	Method	L=1	L=2	L=3	L=4	L=5	L=6	L=7	L=8	L=9	Trend
MATH	**ReM-MoA (full)**	**68.0±0.4**	**72.5±0.5**	**75.0±0.4**	**77.0±0.6**	**78.5±0.4**	**79.5±0.4**	**80.2±0.6**	**80.7±0.6**	**81.0±0.8**	↑↑
	− cross-layer access	68.0±0.4	72.5±0.5	74.1±0.5	75.5±0.5	76.0±0.5	76.4±0.6	76.5±0.6	76.3±0.7	76.2±0.6	→
	− routing diversification	68.0±0.4	72.0±0.5	74.0±0.6	75.2±0.5	76.2±0.6	77.0±0.6	77.4±0.6	77.6±0.6	77.7±0.8	↑→
	− rationale	67.7±0.5	71.9±0.5	74.2±0.7	75.9±0.5	77.2±0.6	78.0±0.6	78.7±0.6	79.1±0.8	79.3±0.8	↑↑
MMLU-redux	**ReM-MoA (full)**	**64.0±0.4**	**66.0±0.3**	**67.0±0.3**	**67.8±0.3**	**68.3±0.5**	**68.6±0.4**	**68.8±0.5**	**68.9±0.5**	**68.9±0.5**	→
	− cross-layer access	64.0±0.4	66.0±0.3	66.4±0.3	66.7±0.2	66.9±0.5	67.0±0.5	66.9±0.6	66.7±0.6	66.8±0.5	→
	− routing diversification	64.0±0.4	65.8±0.3	66.5±0.3	66.9±0.4	67.2±0.4	67.4±0.3	67.5±0.5	67.6±0.5	67.6±0.5	→
	− rationale	63.9±0.3	65.8±0.4	66.5±0.5	67.2±0.4	67.5±0.4	67.8±0.4	68.0±0.4	68.0±0.4	68.0±0.5	→
Formal Logic	**ReM-MoA (full)**	**47.5±0.9**	**51.7±0.8**	**54.1±0.9**	**56.3±0.9**	**57.8±0.9**	**59.4±1.0**	**60.3±0.9**	**60.2±1.1**	**61.1±1.1**	↑↑
	− cross-layer access	47.5±0.9	51.7±0.8	53.3±0.9	54.9±0.7	54.9±0.9	55.7±0.9	55.7±0.9	55.7±1.0	55.7±1.0	→
	− routing diversification	47.5±0.9	50.6±0.8	53.2±0.9	54.1±1.0	55.6±1.1	56.2±1.1	57.0±0.8	57.0±1.1	57.0±0.9	→
	− rationale	46.7±0.9	50.6±0.9	53.3±0.8	54.9±1.1	56.2±1.0	57.0±0.8	57.8±1.0	58.6±1.0	58.6±1.0	↑↑
CRUX	**ReM-MoA (full)**	**56.5±0.5**	**60.5±0.6**	**63.0±0.5**	**65.2±0.5**	**67.0±0.6**	**68.5±0.7**	**69.5±0.6**	**70.0±0.7**	**70.2±0.8**	↑↑
	− cross-layer access	56.5±0.5	60.5±0.6	62.1±0.4	63.5±0.7	64.2±0.6	64.7±0.6	64.5±0.6	64.4±0.6	64.5±0.8	→
	− routing diversification	56.5±0.5	60.0±0.5	62.0±0.5	63.5±0.6	64.8±0.5	65.8±0.7	66.5±0.6	66.8±0.7	66.9±0.7	↑→
	− rationale	56.1±0.4	59.8±0.4	62.1±0.4	63.9±0.5	65.4±0.5	66.5±0.7	67.3±0.6	67.8±0.8	68.0±0.9	↑↑
HellaSwag	**ReM-MoA (full)**	**75.5±0.3**	**78.8±0.2**	**80.5±0.5**	**81.5±0.3**	**82.2±0.5**	**82.6±0.3**	**82.8±0.4**	**82.9±0.6**	**82.9±0.3**	→
	− cross-layer access	75.5±0.3	78.8±0.2	79.7±0.3	80.3±0.3	80.6±0.3	80.5±0.5	80.4±0.3	80.2±0.5	80.3±0.5	→
	− routing diversification	75.5±0.3	78.5±0.4	79.7±0.3	80.4±0.4	81.0±0.3	81.3±0.3	81.5±0.5	81.6±0.5	81.6±0.4	→
	− rationale	75.3±0.2	78.5±0.3	79.8±0.3	80.8±0.3	81.5±0.3	81.8±0.4	82.0±0.2	82.2±0.4	82.2±0.4	→

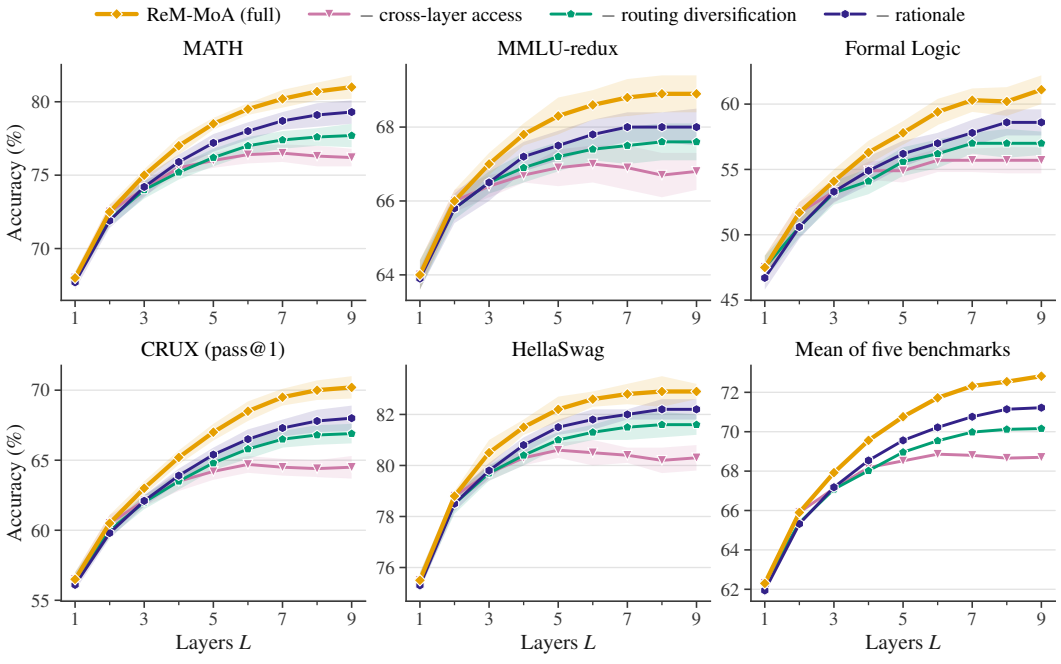

Figure 8: Per-benchmark ablations under depth scaling (N=3) with the values of Table 11. All variants use the off-the-shelf Reviewer. Shaded bands in the per-benchmark panels show one standard deviation over five seeds, and the last panel averages the five benchmarks.

protocol follows Li et al. (2025), who do not specify an embedding model, so we use all-MiniLM-L6-v2 as a widely adopted sentence encoder. The encoder truncates inputs beyond 256 word pieces, so the score reflects similarity among the opening portions of each trace. The three route families also produce different prompt preambles, so part of the measured diversity may reflect prompt wording rather than distinct reasoning paths. Table 12 reports the odd layers from L=1 to L=9.

Table 12: Vendi Score of the N=3 proposer outputs under depth scaling, averaged over the five benchmarks. The score ranges from 1 for identical outputs to 3 for orthogonal outputs. Entries are mean±std over five seeds, and our methods are in **bold**.

Method	L=1	L=3	L=5	L=7	L=9
Standard MoA	2.52±0.04	1.78±0.06	1.42±0.04	1.20±0.02	1.10±0.02
RMoA	2.52±0.04	1.95±0.03	1.65±0.04	1.45±0.03	1.35±0.04
AttentionMoA	2.30±0.05	1.68±0.04	1.38±0.05	1.22±0.03	1.14±0.02
ReM-MoA	**2.52±0.04**	**2.28±0.06**	**2.08±0.06**	**1.94±0.07**	**1.85±0.05**
ReM-MoA*	**2.52±0.04**	**2.30±0.05**	**2.12±0.05**	**1.98±0.05**	**1.90±0.03**

E Reviewer Distillation Results

Table 13 and Figure 9 report the three Reviewers of Section 5.5 at every depth.

Table 13: Reviewer distillation under depth scaling (N=3) on the five distillation domains. Entries are mean±std over five seeds (accuracy in %, pass@1 for MBPP). ReM-MoA uses the off-the-shelf Reviewer, and ReM-MoA* uses the Reviewer distilled on all five domains. For each domain, ReM-MoA* (held-out) uses a Reviewer distilled on the other four domains. Trend markers follow Table 1.

Domain	Reviewer	L=1	L=2	L=3	L=4	L=5	L=6	L=7	L=8	L=9	Trend
MATH	ReM-MoA	68.0±0.4	72.5±0.5	75.0±0.4	77.0±0.6	78.5±0.4	79.5±0.4	80.2±0.6	80.7±0.6	81.0±0.8	↑↑
	ReM-MoA* (held-out)	68.2±0.4	72.9±0.3	75.7±0.5	78.0±0.4	79.7±0.4	80.9±0.5	81.8±0.6	82.5±0.7	83.0±0.6	↑↑
	ReM-MoA*	**68.3±0.4**	**73.5±0.3**	**76.8±0.4**	**79.2±0.3**	**81.0±0.4**	**82.2±0.4**	**83.0±0.5**	**83.7±0.6**	**84.0±0.6**	↑↑
ARC-Challenge	ReM-MoA	86.8±0.2	90.2±0.2	91.2±0.3	91.8±0.4	92.2±0.4	92.5±0.3	92.6±0.4	92.7±0.3	92.7±0.4	→
	ReM-MoA* (held-out)	86.9±0.3	90.4±0.2	91.5±0.2	92.1±0.4	92.5±0.2	92.8±0.3	93.0±0.3	93.1±0.2	93.1±0.4	→
	ReM-MoA*	**87.0±0.2**	**90.6±0.4**	**91.8±0.2**	**92.5±0.2**	**92.8±0.2**	**93.2±0.3**	**93.4±0.2**	**93.5±0.3**	**93.5±0.2**	→
MedQA	ReM-MoA	44.8±0.6	51.0±0.4	54.0±0.4	56.2±0.6	57.8±0.5	59.0±0.6	59.8±0.7	60.2±0.7	60.5±0.7	↑↑
	ReM-MoA* (held-out)	45.0±0.3	51.4±0.5	54.5±0.6	56.8±0.5	58.5±0.6	59.7±0.6	60.5±0.6	61.0±0.6	61.4±0.6	↑↑
	ReM-MoA*	**45.2±0.4**	**52.0±0.3**	**55.5±0.5**	**57.8±0.4**	**59.5±0.5**	**60.8±0.4**	**61.5±0.5**	**62.0±0.5**	**62.2±0.5**	↑↑
MBPP	ReM-MoA	83.0±0.4	85.0±0.3	86.5±0.3	87.5±0.3	88.0±0.2	88.4±0.4	88.6±0.4	88.8±0.5	88.9±0.4	↑→
	ReM-MoA* (held-out)	83.1±0.3	85.2±0.4	87.0±0.4	88.1±0.3	88.5±0.3	88.9±0.3	89.1±0.2	89.4±0.4	89.5±0.5	↑→
	ReM-MoA*	**83.8±0.3**	**86.5±0.3**	**88.5±0.2**	**89.8±0.2**	**90.5±0.2**	**91.0±0.2**	**91.4±0.4**	**91.5±0.3**	**91.6±0.4**	↑→
HellaSwag	ReM-MoA	75.5±0.3	78.8±0.2	80.5±0.5	81.5±0.3	82.2±0.5	82.6±0.3	82.8±0.4	82.9±0.6	82.9±0.3	→
	ReM-MoA* (held-out)	75.5±0.4	78.9±0.4	80.6±0.3	81.6±0.2	82.4±0.3	82.8±0.4	83.0±0.3	83.1±0.4	83.2±0.5	→
	ReM-MoA*	**75.8±0.2**	**79.5±0.2**	**81.2±0.3**	**82.2±0.3**	**83.0±0.4**	**83.4±0.3**	**83.7±0.2**	**84.0±0.3**	**84.3±0.4**	↑↑

F Case Studies

We trace two problems through the four methods at L=9 and N=3. Tables 14 and 15 give the final answer of each method and the behavior of its agents across layers.

Case 1 (MATH, Algebra, Level 5). Find the sum of all values of x such that $2^{x^2-3x-2} = 4^{x-4}$. The answer is 5. The key step is $4^{x-4} = 2^{2x-8}$, which yields $x^2 - 5x + 6 = 0$ with roots 2 and 3.

Case 2 (MMLU Formal Logic). Determine whether the argument with premises $P \supset Q$ and $Q \cdot R$ and conclusion $\neg P \cdot R$ is valid, and if it is invalid, choose a counterexample. The answer is (B): with $P = Q = R = \mathrm{T}$ both premises hold and the conclusion is false.

Standard MoA fails both cases through unchecked error propagation. RMoA succeeds through information preservation in Case 1 but cannot correct an incomplete verification that all agents share in Case 2. AttentionMoA corrects errors through interaction in Case 2, but its consensus mechanism suppresses the correct minority in Case 1. ReM-MoA succeeds on both by combining quality-ranked memory with diversified routing, so that information preservation and error correction occur together.

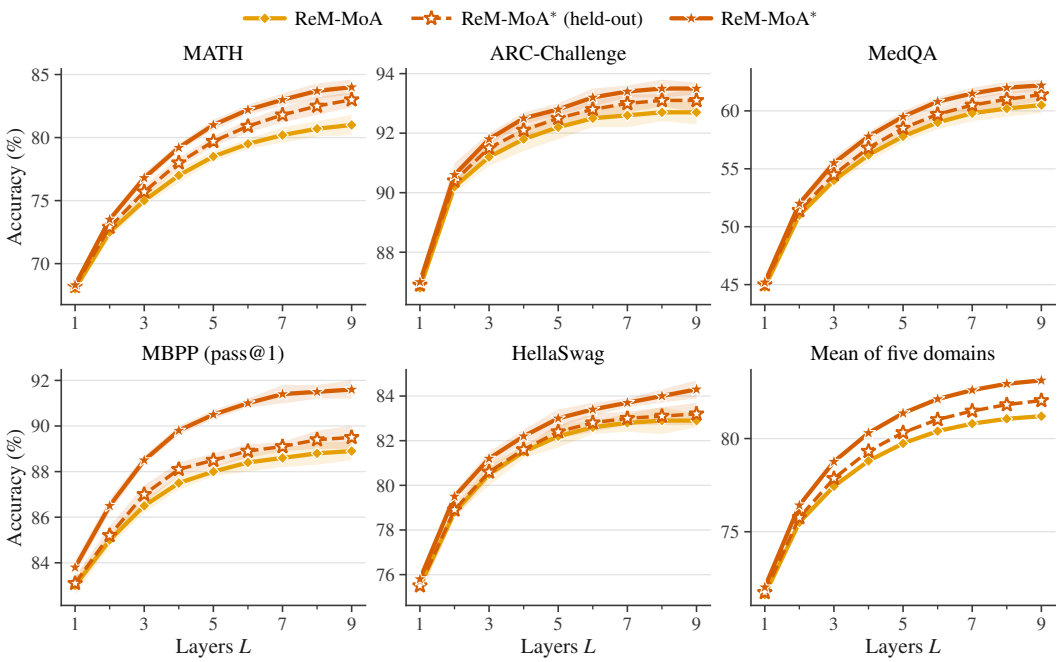

Figure 9: Reviewer distillation under depth scaling (N=3) with the values of Table 13. Shaded bands in the per-domain panels show one standard deviation over five seeds, and the last panel averages the five domains.

Table 14: Case 1. Final answers at L=9 and agent behavior across layers.

Method	**Answer**	**Behavior**
Standard MoA	4 (×)	Agents at L=7 simplify 4^{x-4} to 2^{x-4}, which leads to $x^2 - 4x + 2 = 0$ with root sum 4. All agents see the same flawed references and the error propagates uncorrected.
RMoA	5 (✓)	The residual preserves the correct base conversion from L=3, but all agents reproduce the same derivation verbatim with no change beyond L=4.
AttentionMoA	4 (×)	One agent converts the base correctly at L=5, but attention at L=6 pushes it toward the majority simplification 2^{x-4} and all agents converge on the wrong equation.
ReM-MoA	5 (✓)	The Top route retrieves a high-scored L=3 trace with the correct conversion. The Con route receives a low-scored L=7 trace whose rationale flags the incorrect base conversion, and the agent explicitly avoids it.

G Prompt Templates and Scoring Rubric

Prompts are reproduced verbatim (including internal punctuation). Ellipses marked [. . .] indicate where repeated blocks are omitted for space. This appendix provides the prompt templates used by the proposer agents, the Reviewer Agent, the aggregator, and the frontier-model teacher (GPT-5.5) during distillation, together with the scoring rubric followed by the Reviewer.

G.1 Scoring Scale and Rubric

The Reviewer assigns each trace an integer score on a 1-10 scale (i.e., S=10 in Equation 3). Following common LLM-as-judge practice, scores are organized into five tiers:

- **9–10 (Excellent)**: reaches the correct answer through a complete, coherent chain of reasoning with no logical errors.

Table 15: Case 2. Final answers at L=9 and agent behavior across layers.

Method	Answer	Behavior
Standard MoA	(A) (×)	Two of three agents select "valid" at L=3 on surface inspection, and by L=9 the majority error has propagated to all agents.
RMoA	(C) (×)	Agents find an assignment with a false conclusion ($P = \mathrm{T}$, $R = \mathrm{F}$) but fail to notice that the second premise $Q \cdot R$ is false under it, so (C) is not a valid counterexample, and the residual preserves this incomplete analysis.
AttentionMoA	(B) (✓)	Attention at L=4 lets the agent that tested $P = Q = R = \mathrm{T}$ correct the others, and all agents agree on (B) with identical traces.
ReM-MoA	(B) (✓)	The Bot route receives the low-scored trace that selected (C) together with the rationale that $Q \cdot R$ is false under its assignment. The agent then tests $P = \mathrm{T}$ with $R = \mathrm{T}$ and identifies (B).

- **7–8 (Good)**: reaches the correct answer with minor reasoning slips, redundancies, or under-justified steps that do not affect the final result.
- **5–6 (Partial)**: demonstrates valid reasoning on parts of the problem but reaches an incomplete or partially incorrect final answer.
- **3–4 (Weak)**: contains substantive reasoning errors propagating to an incorrect final answer, though some intermediate steps may still be useful as counter-examples.
- **1–2 (Poor)**: final answer is wrong and the reasoning chain contains major flaws throughout.

Within each tier, the higher score indicates a stronger instance (e.g., a 10 is preferred to a 9). The accompanying rationale (one per trace) cites concrete strengths and weaknesses of the specific trace, anchored in its reasoning steps and conclusion. Rationales are independent across traces (Section 4.1).

G.2 Proposer Prompt

At layer 1, the proposer receives only the problem. At layer $l \geq 2$, the proposer additionally receives the curated reference set $\mathcal{S}_{l,j}$. Each entry in the reference set carries a Reviewer score and rationale.

Proposer Prompt

```
You are solving the following problem.

Problem:
{problem}

## Reference Reasoning Traces (with quality ratings)

Below are reasoning traces from previous attempts on this
problem, each rated by an expert reviewer on a 1-10 scale
(higher is better):

### [Score: {s_1}/10] Reference Trace 1:
{trace_1}

Reviewer's Assessment: {rationale_1}

### [Score: {s_2}/10] Reference Trace 2:
{trace_2}

Reviewer's Assessment: {rationale_2}

[...] (additional traces follow the same format)

## Your Task

Drawing on what the high-scoring traces did well and what
```

```
the low-scoring traces got wrong, produce your own reasoning
trace and final answer. Do not simply copy any reference
trace; contribute distinct reasoning.
```

At layer 1, the "Reference Reasoning Traces" block is omitted and the prompt contains only the problem and the task instruction.

G.3 Reviewer Prompt

The Reviewer is invoked once per layer with all N proposer traces of that layer. It emits scores and rationales for all N traces in a single response, parsed into the memory entry of Equation 4.

Reviewer Prompt

```
You are an expert reviewer evaluating reasoning traces.

Problem:
{problem}

You are given {N} candidate reasoning traces. Evaluate each
trace by comparing it against the others, then assign:
  (1) an integer score from 1-10 (rubric below);
  (2) a brief rationale (1-3 sentences) explaining the
      score, citing specific reasoning steps.

Scoring rubric (1-10):
  9-10: Excellent (correct and well-reasoned).
  7-8 : Good (correct with minor flaws).
  5-6 : Partial (incomplete or partly wrong).
  3-4 : Weak (mostly incorrect; useful as counter-example).
  1-2 : Poor (entirely wrong).

### Trace 1
{trace_1}

### Trace 2
{trace_2}

[...] (Traces 3 through N follow the same format)

Respond in the following JSON format:

{
  "trace_1": {"score": <int>, "rationale": "<text>"},
  "trace_2": {"score": <int>, "rationale": "<text>"},
  [...]
  "trace_N": {"score": <int>, "rationale": "<text>"}
}
```

G.4 Aggregator Prompt

After layer L, the aggregator receives the problem and three curated reference sets drawn from the memory through the same three families used by proposer agents (Section 4), and produces the final answer.

Aggregator Prompt

```
You are producing the final answer to the following problem
by synthesizing the strongest reasoning across multiple
```

```
attempts.

Problem:
{problem}

## Top-scoring traces (positive reinforcement)
{top_traces_with_scores_and_rationales}

## Bottom-scoring traces (common mistakes to avoid)
{bot_traces_with_scores_and_rationales}

## Contrastive selection (top paired with bottom)
{con_traces_with_scores_and_rationales}

Synthesize the most reliable reasoning from the top-scoring
traces while avoiding the errors exhibited in the
bottom-scoring traces. Output your final answer with a
concise justification.
```

G.5 Teacher Prompt for Distillation

The frontier-model teacher (GPT-5.5) receives the same inputs as the Reviewer (the problem and the proposer traces) and is constrained to emit labels in the same JSON schema as the Reviewer. Because the teacher sees no ground-truth answer, the student learns to imitate judgments made from the same information that the Reviewer has at inference.

Teacher Prompt (Distillation Labeling)

```
You are an expert evaluator generating high-quality reviewer
labels for distillation training.

Problem:
{problem}

Below are {N} candidate reasoning traces. Assign each
trace a score (1-10) and a brief rationale, exactly as a
strong reviewer would. Use the same rubric:

  9-10: Excellent (correct and well-reasoned).
  7-8 : Good (correct with minor flaws).
  5-6 : Partial (incomplete or partly wrong).
  3-4 : Weak (mostly incorrect; useful as counter-example).
  1-2 : Poor (entirely wrong).

### Trace 1
{trace_1}

[...] (Traces 2 through N follow the same format)

Respond in JSON:
{
  "trace_1": {"score": <int>, "rationale": "<text>"},
  [...]
  "trace_N": {"score": <int>, "rationale": "<text>"}
}
```