

FIVER: Adaptive Fusion of Formal and Informal Verification for Mathematical Reasoning in LLMs

Yujie Li^{1*} Ao Xu^{2,3*†} Ziyou Guo⁴ Tieru Wu^{2†}

¹School of Mathematics, Jilin University, ²School of Artificial Intelligence, Jilin University

³Zhongguancun Academy, ⁴Northeast Forestry University

{liyj25,xuao24}@mails.jlu.edu.cn, ziyouguo19@gmail.com, wutr@jlu.edu.cn

Abstract

Step-level verification is central to improving the reliability of multi-step mathematical reasoning in large language models (LLMs), yet existing methods typically adopt a fixed strategy between formal and informal proof. Formal methods can produce interpretable and machine-checkable proof objects. However, they are computationally expensive and limited by the success rates of formalization, semantic alignment, and proof search. Informal methods based on LLM judges offer lower cost and broader coverage, but their judgments cannot by themselves provide rigorous checkable proofs. Thus, a mathematical verification agent should adaptively choose the proof form for each step, covering the union of both capabilities while preferring cheaper informal verification in their overlap. Based on this formulation, we propose FIVER¹ (Formal-Informal Verification Agent with Efficient Routing), a step-level mathematical verification agent that adaptively fuses formal and informal verification. FIVER uses a learnable router modeled with item response theory (IRT) to select either a formal or informal workflow, forming a new step-verification architecture. Under the Qwen3-8B setting, FIVER improves over the purely formal HERMES baseline by 10.77% on MATH-398 and 33.33% on AIME 2025, respectively, while using only 58.8% of HERMES’s verification tokens on average.

1 Introduction

LLMs and open reasoning models have made substantial progress on mathematical reasoning benchmarks (Yang et al., 2025; Qwen Team, 2026; DeepSeek-AI, 2025a), but their multi-step solution trajectories still lack reliability guarantees. The correctness of a final answer often depends

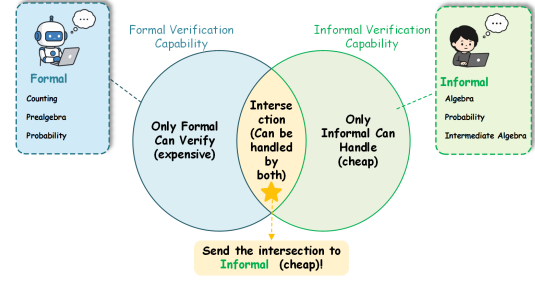


Figure 1: Motivation of FIVER’s verification routing.

on key intermediate claims. Hidden errors can propagate through later derivations and lead to wrong answers. Step-level verification has therefore become an important direction for improving the reliability of mathematical reasoning. Existing work broadly follows two routes: one uses reward models, judges, critics, and debate mechanisms to check natural-language reasoning steps (Lightman et al., 2023; Wang et al., 2024a; Setlur et al., 2025; Zhang et al., 2025b; Zheng et al., 2023; Gou et al., 2024; Du et al., 2023; Chan et al., 2023). The other translates mathematical propositions into formal proof environments and relies on proof checkers to verify correctness (Liu et al., 2025a; Yao et al., 2025; Ospanov et al., 2025; Lin et al., 2024; Wang et al., 2025b; Ren et al., 2025; Lin et al., 2025).

These two verification paradigms have complementary strengths and limitations, as illustrated in Figure 1. Formal verifiers such as Lean 4 (de Moura and Ullrich, 2021) can return transparent, machine-checkable proof objects when they succeed. However, the formal path is also more costly and brittle: a natural-language claim may be hard to translate accurately into a Lean proposition, proof search may time out, and even simple but irregularly phrased steps may trigger compilation or semantic-alignment failures. In contrast, informal verifiers have lower latency and wider applicability, and can quickly produce judgments, rationales, and possible counterexam-

*Equal contribution.

†Corresponding authors.

¹The code is available at <https://github.com/lyj846571-blip/FIVER>.

ples. Yet their outputs remain model judgments, affected by model stochasticity and opacity, and cannot replace formal proof objects.

To combine the capabilities of both paths, we propose FIVER, a step-level mathematical verification agent that adaptively fuses formal and informal verification through efficient routing. The reasoning model emits only workflow-agnostic step-verification requests. FIVER then decides whether to send the step to a Lean4-based formal verifier or to a DeepSeek-based informal verifier. Inspired by prior work on routing and cascaded LLM systems (Chen et al., 2024; Ding et al., 2024; Ong et al., 2024; Shnitzer et al., 2024; Ding et al., 2025; Fernandez et al., 2026), we make this decision with a learnable router based on item response theory (IRT). The router jointly estimates verification success probability and latency cost, and learns which steps are better suited to formal proof and which can be verified by natural-language debate at lower cost.

FIVER’s formal verification path is not a one-shot stateless tool call. Inspired by work on reflection, experience reuse, and persistent memory in agents (Shinn et al., 2023; Madaan et al., 2023; Zhao et al., 2024; Wang et al., 2023a; Packer et al., 2023; Lyu et al., 2026; You et al., 2026), we design the Lean4 verifier as a self-evolving component that updates continuously along the solution trajectory. Successful Lean statements and proofs are stored as known conditions for later proofs. Failed translations, failed proofs, and failed negation-proof attempts are written into memory as negative guidance according to their mathematical categories, helping the verifier avoid repeated errors.

FIVER’s informal verification path uses a debate-style design. For the same candidate step, FIVER constructs two complementary views: the support view argues for the step from the problem conditions and verified context, while the attack view checks missing assumptions, invalid transformations, boundary cases, and potential counterexamples. An aggregator then produces a unified verification state from the consistency of the two conclusions and the self-consistency of their rationales: the step is accepted when the support rationale is sufficient and the attack side raises no substantive objection; it is rejected when the attack side finds a clear error; and it returns verification failure when the evidence conflicts or is insufficient.

Overall, this paper makes the following contri-

butions: **(1)** We propose a multi-objective verification router that models step-tool fit and jointly considers verification success and invocation cost. **(2)** We build FIVER, a mathematical verification framework that combines self-evolving formal verification with informal verification. It routes each step to a suitable verifier and sends steps that both verifiers can handle to the cheaper one. **(3)** We validate the effectiveness and efficiency of FIVER across four reasoning models from different model families and four mathematical benchmarks, with ablation studies on its verification paths and Evolution Memory. FIVER achieves the best accuracy in all 16 model-benchmark settings, improving by up to 33.33% over the strongest non-FIVER baseline while using only 58.8% of HERMES’s verification tokens on average under Qwen3-8B.

2 Related Work

Recent frontier language models, including GPT-5, DeepSeek-V3.2, and Gemini 3, have advanced general-purpose reasoning, tool use, code generation, and multimodal understanding (OpenAI, 2025; DeepSeek-AI, 2025b; Google, 2025). In mathematical reasoning, specialized reasoning models, process supervision, and verification mechanisms have further improved the reliability of multi-step solutions. Formal and informal verification are two central paradigms for this goal: the former provides checkable proof evidence, while the latter offers flexible natural-language judgments. We propose FIVER, a step-level mathematical verification framework that adaptively integrates formal and informal verification to improve verification capability while reducing invocation cost.

2.1 Formal verification

Lean 4 (de Moura and Ullrich, 2021) provides the foundation for checkable proofs in recent mathematical verification methods. Some work improves formalization and proof ability through subgoal decomposition, scaffolded data synthesis, and reinforcement learning. DeepSeek-Prover-V2 (Ren et al., 2025), Goedel-Prover-V2 (Lin et al., 2025), Kimina-Prover (Wang et al., 2025a), ReForm (Chen et al., 2026), and TheoremLlama (Wang et al., 2024b) provide reliable tools for translating natural-language mathematics into formal language and proving the resulting goals.

Another line studies how to use these model

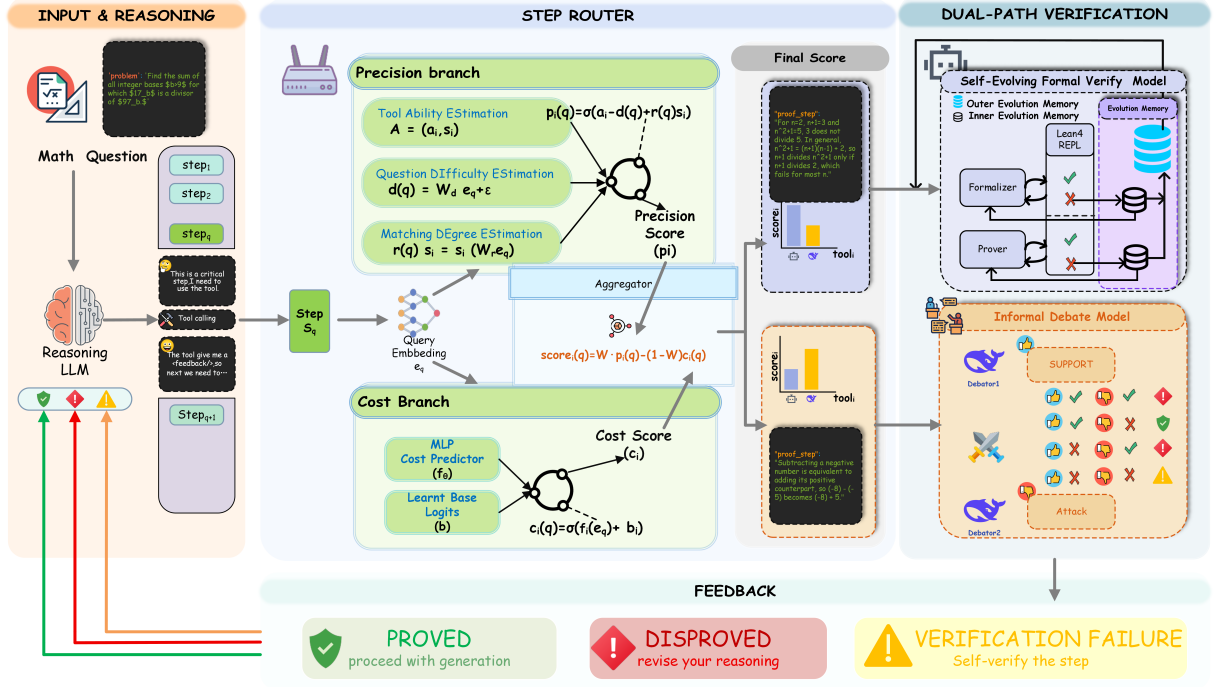


Figure 2: The architecture of FIVER as a step-level verification agent. FIVER contains four modules: INPUT & REASONING, STEP ROUTER, DUAL-PATH VERIFICATION, and FEEDBACK. The Step Router combines a precision branch and a latency branch to route candidate steps to either a Lean-based formal verifier or an informal debate model. Feedback returns PROVED, DISPROVED, or VERIFICATION FAILURE to the solver and writes new verification evidence into memory.

tools efficiently. Lean-STaR (Lin et al., 2024) and Let’s Reason Formally (Wang et al., 2025b) study the alternating use of natural-language reasoning and formal proof. FANS (Yao et al., 2025) formalizes an entire candidate natural-language process as Lean 4 propositions and selects Best-of-N with an outcome reward model (ORM). SAFE (Liu et al., 2025a) performs backtracking step-level formal verification and combines it with an LSTM for process scoring. HERMES (Ospanov et al., 2025) uses agent tool-calling ability to interleave natural-language reasoning with Lean 4 translation, proof, and feedback loops. These studies show the value of formal proof objects in mathematical verification, while also exposing the cost, brittleness, and coverage limits of using formal verification as the only workflow.

2.2 Informal verification

One route for informal verification converts a reasoning process or final answer into a rankable scalar signal. Skywork-Reward-V2 (Liu et al., 2025b) focuses on outcome-level reward modeling for scoring full responses or candidate solutions, while GenPRM (Zhao et al., 2025) ex-

tends process-level evaluation through generative reasoning. Such methods usually output learned scores or rewards that can be used for Best-of-N, reranking, or step filtering, but they do not directly provide explicit adversarial checks. Another route directly uses language models for judging, critique, or debate. LLM-as-a-Judge (Zheng et al., 2023) shows that strong models can serve as judges. Later work also notes that LLM judges can be affected by position, length, and self-preference biases (Wang et al., 2023b). CRITIC (Gou et al., 2024) revises model outputs through tool-interactive critique. Multi-agent debate (Du et al., 2023) and ChatEval (Chan et al., 2023) use multiple agents to argue or judge from different views, improving factuality and evaluation robustness. These judge, critique, and debate methods motivate FIVER’s informal path, where a support view and an attack view debate each step to return transparent rationales and conflict signals beyond scalar rewards.

2.3 Adaptive routing

Adaptive routing was first used in LLM systems mainly to trade off quality, cost, and latency

by assigning user requests to different models, APIs, or cascaded systems. FrugalGPT (Chen et al., 2024) reduces cost while maintaining response quality by cascading different LLM APIs. Hybrid LLM (Ding et al., 2024) learns quality-aware and cost-aware routing among model workflows. RouteLLM (Ong et al., 2024) uses preference data to learn routing between stronger and weaker LLMs. BEST-Route (Ding et al., 2025) jointly selects the model and the amount of test-time sampling under a quality–compute trade-off. RADAR (Fernandez et al., 2026) applies item response theory (Embretson and Reise, 2000) to reasoning-model routing by selecting configurations according to problem difficulty and model ability. Large Language Model Routing with Benchmark Datasets (Shnitzer et al., 2024) systematically compares routing datasets and router designs. Unlike general LLM routing, where candidate systems are often organized by an approximate capability hierarchy in which stronger models subsume weaker ones, FIVER routes between complementary verification workflows. We therefore incorporate tool–step fit into the success predictor and build a FIVER-specific router for formal–informal verification.

2.4 Memory-enhanced agents

Persistent memory and reflection mechanisms are important foundations for agent self-improvement. Generative Agents (Park et al., 2023) combines memory, reflection, and planning to maintain long-term coherent behavior. Reflexion (Shinn et al., 2023) writes language feedback from failed attempts into memory so later attempts can avoid repeated errors. Self-Refine (Madaan et al., 2023) iteratively improves outputs through self-feedback. ExpeL (Zhao et al., 2024) distills reusable experience from agent trajectories. Voyager (Wang et al., 2023a) continuously accumulates a skill library during open-ended exploration. MemGPT (Packer et al., 2023) studies explicit memory management for long-context agents. EvoScientist (Lyu et al., 2026) maintains ideation memory and experimentation memory, enabling multi-agent scientists to reuse effective strategies and avoid repeated failures. Agent-as-a-Judge (You et al., 2026) summarizes the importance of planning, tool use, multi-agent collaboration, and persistent memory in agentic evaluation. In this work, memory is used to support formal verification directly: stored successful proofs provide reusable conditions for later

attempts, while stored failures expose invalid assumptions and repeated errors as prior guidance for subsequent verification.

3 Methodology

Figure 2 illustrates the overall architecture of FIVER. It is a step-level mathematical verification agent composed of four modules: INPUT & REASONING, STEP ROUTER, DUAL-PATH VERIFICATION, and FEEDBACK. Given a problem, the reasoning model generates candidate steps and emits workflow-agnostic verification calls when external checking is needed. The router selects between the Self-Evolving Formal Verify Model and the Informal Debate Model based on step representation, predicted fit, and latency cost. The selected verifier returns a unified verification state to the model. This state is written into the reasoning history and used to update verification memory for subsequent decisions.

3.1 Input & Reasoning

FIVER takes a mathematical problem P as input. Given a reasoning model $\mathcal{M}_R$, the model constructs a solution trajectory containing the final answer over multiple generation rounds, $S = \text{concat}(q_1, q_2, \dots, q_m, \alpha)$, where q_j denotes the j -th natural-language reasoning step and α denotes the final answer. Initially, the model receives only the problem, $H_0 = (P)$. At the beginning of round t , the visible context is $H_{t-1} = (P, q_{1:t-1}, F_{1:t-1})$, where $F_j = \text{ToolMsg}(s_j, f_j)$ denotes the feedback message returned by the verification tool at the j -th call, s_j is the verification state, and f_j is the feedback text. The definitions of state s_j and feedback text f_j are given in Section 3.4.

At each round, the reasoning model outputs one of three types:

$$\mathcal{M}_R(H_{t-1}) \in \{q_t, \tilde{q}_t, \alpha\}. \quad (1)$$

Here, q_t is a regular reasoning step that does not require external rigorous verification, $\tilde{q}_t$ is a key step submitted to the verification tool, and α is the final answer. If the reasoning model outputs q_t , it continues reasoning until it obtains either $\tilde{q}_t$ or α .

If the model outputs $\tilde{q}_t$, verification is invoked, and the model submits the step to verify together with its local context. After verification, the next-round input is updated as

$$H_t = H_{t-1} \oplus \tilde{q}_t \oplus \text{ToolMsg}(s_t, f_t), \quad (2)$$

where $\oplus$ denotes appending the step to verify and the tool feedback to the history. If the reasoning model outputs α , the problem is completed and the answer is returned.

3.2 Step Router

The Step Router is the main adaptive component of FIVER and runs only after the Input & Reasoning module emits a generic verification call. In step-level experiments, the router chooses between two verification workflows, $\mathcal{T} = \{\text{FORMAL}, \text{INFORMAL}\}$. The router estimates both verification success probability and latency cost from the step representation, and thereby selects between formal and informal verification.

Each step to verify $\tilde{q}_t$ is encoded as

$$e_t = \text{Encoder}(\tilde{q}_t) \in \mathbb{R}^d. \quad (3)$$

This embedding is the fixed input to the router.

Precision branch For each tool $i \in \mathcal{T}$, the precision branch estimates tool ability, step difficulty, and tool-step matching:

$$p_i(\tilde{q}_t) = \sigma\left(\beta_i - d(\tilde{q}_t) + h(\tilde{q}_t)^\top s_i\right), \quad (4)$$

where β_i is a learnable tool ability, $d(\tilde{q}_t)$ is the predicted step difficulty, $h(\tilde{q}_t)$ is the step ability-demand representation, and s_i is the learnable expertise vector for tool i . Specifically,

$$d(\tilde{q}_t) = |W_d e_t| + \epsilon, \quad h(\tilde{q}_t) = W_h e_t. \quad (5)$$

The expertise term enables the router to distinguish which workflow fits different steps: some steps suit the Self-Evolving Formal Verify Model, while others are better checked by the Informal Debate Model.

Cost branch The latency branch predicts the normalized latency cost for each verification path:

$$c_i(\tilde{q}_t) = \sigma(g_i(e_t) + b_i), \quad (6)$$

where g_i is a tool-specific prediction head and b_i is a learnable base-latency logit.

Aggregator For each tool, the router computes

$$\text{score}_i(\tilde{q}_t) = \lambda \cdot p_i(\tilde{q}_t) - (1 - \lambda) \cdot c_i(\tilde{q}_t), \quad (7)$$

where $\lambda \in [0, 1]$ is a tunable weight. It then selects

$$i^*(\tilde{q}_t) = \arg \max_i \text{score}_i(\tilde{q}_t). \quad (8)$$

A larger λ makes the router favor higher predicted success, while a smaller λ makes it favor the lower-latency Informal Debate Model.

During training, each sample provides a binary success label y_i and latency measurement ℓ_i for each tool. The two branches are optimized separately:

$$\mathcal{L}_{\text{succ}} = \sum_i \text{BCE}(p_i(\tilde{q}_t), y_i), \quad (9)$$

$$\mathcal{L}_{\text{cost}} = \frac{\sum_i m_i \text{SmoothL1}(c_i(\tilde{q}_t), t_i)}{\sum_i m_i}, \quad (10)$$

where BCE is binary cross entropy, SmoothL1 is a Huber-style smooth L_1 regression loss, t_i is the globally normalized log-latency target, and m_i masks missing latency values. The concrete definitions of BCE, SmoothL1, and t_i , together with latency normalization details, are given in Appendix C.

3.3 Dual-Path Verification

After Step Router selection, the step to verify is assigned to one of the two paths in the dual-path verification module. The Self-Evolving Formal Verify Model handles formal verification, and the Informal Debate Model handles informal verification. Both workflows produce only internal decision states, namely PROVED, DISPROVED, or VERIFICATION FAILURE. Natural-language feedback for the reasoning model is generated by the feedback module described in Section 3.4.

Self-Evolving Formal Verify Model The formal path uses a Lean4 verifier with a two-level memory mechanism, consisting of Outer Evolution Memory and Inner Evolution Memory. Outer Evolution Memory accumulates long-term failure experience, while Inner Evolution Memory stores short-term failures from repeated translation or prover attempts for the current step. Given a step $\tilde{q}_t$, the agent retrieves top- k_p positively relevant memories from Outer Evolution Memory as known conditions before formalization, and also retrieves top- k_o negatively relevant memories under the same mathematical property as failure priors. It then enters a four-stage formalization pipeline.

First, the agent retrieves top- k_i negative translation memories from Inner Evolution Memory and combines them with translation memories from the Outer Evolution Memory failure priors

to form the translation prompt. Under these prior constraints, the formalizer translates the natural-language step into a Lean4 statement. If the Lean4 trusted kernel can compile the statement, the agent proceeds to the next stage. Otherwise, it writes the failure case into Inner Evolution Memory and repeats up to N_t times within the budget until it obtains a compilable statement or exhausts the budget.

Second, the agent back-translates the Lean statement to check semantic consistency with the original natural-language step and determines the mathematical type of the step so that category-level failure memories can be retrieved.

Third, the agent retrieves top- k_i negative prover memories from Inner Evolution Memory and combines them with prover memories from the Outer Evolution Memory failure priors to form the proof prompt. The proof model attempts both the forward goal and the reverse goal: the forward proof module retrieves proof memories, while the reverse branch retrieves disproof memories. If a Lean proof generated by either branch passes compilation, the agent returns the corresponding state. Otherwise, it writes the failed trajectory into Inner Evolution Memory and repeats up to N_p times within the budget.

Fourth, the agent uploads the new records accumulated in Inner Evolution Memory to Outer Evolution Memory and clears Inner Evolution Memory, providing an updated retrieval context for subsequent steps.

The formal path returns three states: PROVED when the forward goal proof compiles successfully; DISPROVED when the reverse goal proof compiles successfully; and VERIFICATION FAILURE when translation, semantic checking, proof search, or Lean execution cannot establish either direction.

Informal Debate Model The informal path verifies a candidate step with two complementary views. The support view acts as a cautious prover and decides whether the step follows from the problem and verified context. The attack view acts as a skeptical reviewer, searching for errors and attempting to find counterexamples. Each view returns a verdict in $\{valid, invalid, uncertain\}$, a confidence score, and a short rationale.

The aggregator maps the outputs of the two views to three verification states. If the attack view finds a clear issue or counterexample with suffi-

cient confidence, the agent returns DISPROVED. If both the support and attack views accept the step and the attack view leaves no unresolved risk, the agent returns PROVED. All other disagreements, missing conditions, or unstable judgments return VERIFICATION FAILURE.

3.4 Feedback

The feedback module converts internal states from different verification workflows into prompts visible to the reasoning model. Let s_j denote the verification state of the j -th reasoning step. The feedback module maps s_j to a feedback message f_j .

$$f_j = \begin{cases} \text{“proceed with generation”}, & s_j = P, \\ \text{“revise your reasoning”}, & s_j = D, \\ \text{“self-verify the step”}, & s_j = V. \end{cases}$$

Here, P denotes PROVED, meaning the verifier accepts the step; D denotes DISPROVED, meaning the verifier finds a contradiction or invalid reasoning in the step; and V denotes VERIFICATION FAILURE, meaning the verifier cannot determine whether the step is correct or incorrect.

This feedback message is appended to the dialogue history in the Input & Reasoning module as a tool response. Verified steps are uploaded to the memory base and saved as known conditions for the next step that requires verification.

4 Experiments

4.1 Experimental Setup

Benchmark and Models We conduct extensive evaluation on four types of mathematical problems, covering standard mathematical reasoning, university-level mathematics, competition-level mathematics, and hard mathematical problems. The corresponding benchmarks are MATH-398, which consists of the 398 MATH-500 problems not used to construct router supervision data in Appendix B (Lightman et al., 2023), MinervaMath, AIME 2025 (Zhang and Math-AI Team, 2025), and HM2 (Roggeveen et al., 2025).

We evaluate FIVER with Qwen3-8B, Qwen3.5-35B, Qwen3.5-122B (Yang et al., 2025; Qwen Team, 2026), and DeepSeek-V4-Flash, covering different model scales and families. All verification components are kept fixed: the router uses Qwen3-Embedding-8B (Zhang et al., 2025a),

the memory index uses Qwen3-Embedding-0.6B (Zhang et al., 2025a), the formal verifier uses Goedel-Formalizer-V2-8B and Goedel-Prover-V2-8B (Lin et al., 2025), and the informal debate model uses DeepSeek-R1-Qwen3-8B (DeepSeek-AI, 2025a).

Router supervision data Router supervision uses the MATH-500-derived step records detailed in Appendix B. Briefly, 102 MATH-500 problems are used to generate 933 verification steps. Each step is evaluated by two verification workflows, producing 1,866 tool-level records. The 933 steps are split into a router training set and a router validation set. The validation set is used only to select λ . The remaining 398 MATH-500 problems form MATH-398 and are used only for final accuracy evaluation.

Baselines Baselines include pure reasoning-model methods (Zero-shot and Majority@5), the current state-of-the-art formal method HERMES (Ospanov et al., 2025), and informal methods represented by the state-of-the-art Skywork-Reward-V2 (Liu et al., 2025b) as the ORM method and GenPRM (Zhao et al., 2025) as the PRM method. Within each group, FIVER and the baselines use the same reasoning model.

Hyperparameters We use $\lambda = 0.34$, $k_p = 3$, $k_o = 1$, $k_i = 2$, $N_t = 4$, $N_p = 4$, and $\epsilon = 0.1$.

4.2 Results

The results are shown in Table 1. FIVER outperforms methods that use only formal or only informal verification across all four datasets. Its gains are especially stable on the more challenging AIME 2025 and HM2 benchmarks, indicating that FIVER improves not only standard mathematical problems but also medium- and high-difficulty long-reasoning settings. FIVER also achieves the highest accuracy on all four benchmarks with DeepSeek-V4-Flash, showing that the benefit of adaptive formal-informal verification extends across reasoning-model families. Additional step-level verification analyses are reported in Appendix B; multi-seed uncertainty results for all reasoning models and Qwen3-8B test-time scaling results are reported in Appendix H.

5 Discussion

5.1 Reasoning-Token Efficiency

Reasoning-token consumption is one of the main drawbacks of Best-of-N methods. This section compares FIVER with formal-only and informal-only methods in terms of reasoning-token usage.

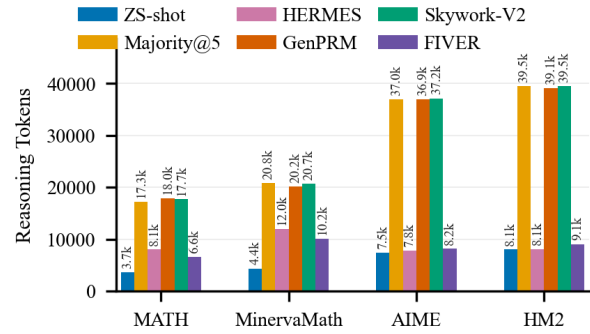


Figure 3: Average reasoning tokens per problem for Qwen3-8B across datasets.

The results presented in Figure 3 show that on Qwen3-8B, FIVER uses about 8.5k reasoning tokens on average, substantially less than Majority@5, Skywork-V2, and GenPRM, all of which average around 28k tokens. Compared with these multi-trajectory sampling methods, FIVER reduces reasoning tokens by about 70% on average. Compared with the efficient HERMES method, FIVER shows some advantage on MATH-398 and MinervaMath. Overall, its reasoning-token usage is about 5.5% lower than HERMES and closer to Zero-shot. In general, FIVER keeps reasoning-token cost low while maintaining an effective reasoning process. The results for the two Qwen3.5 models are given in Appendix E.

5.2 Verification Cost

One advantage of FIVER over formal-only methods is that it can route steps that both formal and informal methods can solve to the cheaper informal method. Using Qwen3-8B as an example, we measure the verification-token usage of FIVER and HERMES across the four datasets.

As shown in Figure 4, FIVER uses only 58.8% of HERMES’s verification tokens on average across all datasets. FIVER has a clear advantage on MATH-398 and MinervaMath, while it is slightly weaker than HERMES on AIME 2025 and HM2 in verification-token usage. We hypothesize that on harder datasets, FIVER’s more effec-

Table 1: Accuracy results on MATH-398, MinervaMath, AIME 2025, and HM2 across four reasoning models. The token budget of each reasoning-model generation is limited to 8192. FIVER and HERMES are @1. For informal process-reward and outcome-reward methods, we first sample N reasoning trajectories and then choose Best-of- N (BoN), with $N = 5$, following HERMES (Ospanov et al., 2025) and SAFE (Liu et al., 2025a).

Reasoning Model	Method	MATH-398	MinervaMath	AIME 2025	HM2
Qwen3-8B	Zero-shot	75.37	47.06	10.00	0.47
	Majority@5	77.89	64.34	23.33	1.57
	Skywork-V2	81.40	66.18	30.00	5.21
	GenPRM	80.15	62.50	23.33	3.79
	HERMES	79.84	67.27	30.00	2.84
	FIVER	88.44	69.85	40.00	6.64
Qwen3.5-35B	Zero-shot	94.47	73.53	66.67	36.97
	Majority@5	94.97	76.84	73.33	39.34
	Skywork-V2	94.22	75.00	73.33	45.97
	GenPRM	93.47	75.74	70.00	44.45
	HERMES	94.47	76.84	70.00	38.38
	FIVER	95.47	79.04	83.33	47.87
Qwen3.5-122B	Zero-shot	94.22	72.79	60.00	44.54
	Majority@5	95.23	72.79	60.00	45.02
	Skywork-V2	96.98	72.79	66.67	46.45
	GenPRM	94.47	70.22	66.67	49.76
	HERMES	94.72	70.59	60.00	46.45
	FIVER	97.49	75.37	70.00	51.66
DeepSeek-V4-Flash	Zero-shot	96.73	85.29	53.33	64.45
	Majority@5	98.24	86.76	63.33	65.40
	Skywork-V2	98.25	86.76	66.67	66.35
	GenPRM	97.49	84.93	63.33	68.72
	HERMES	96.73	85.29	60.00	63.03
	FIVER	98.74	88.60	70.00	70.14

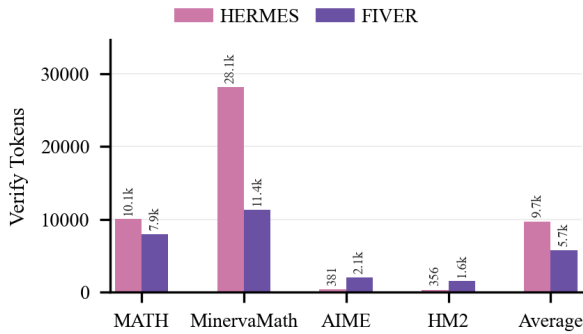


Figure 4: Average verification tokens per problem for Qwen3-8B across datasets.

tive routing finds more errors, which induces additional reasoning and leads to larger accuracy gains.

5.3 Ablation Study

FIVER consists of four main components, among which the Self-Evolving Formal Verify Model and the Informal Debate Model are the two key verification components. This section ablates these two components. For the formal workflow, we also ablate the Evolution Memory system. For the first

Table 2: Ablation results with Qwen3.5-122B as the reasoning model. SEFV denotes the Self-Evolving Formal Verify Model, IDM denotes the Informal Debate Model, and EM denotes Evolution Memory.

Method	MATH-398	MinervaMath	AIME 2025	HM2
<i>Ablation of verification paths</i>				
FIVER	97.49	75.37	70.00	51.66
Only IDM	96.98	73.16	60.00	48.34
	-0.52%	-2.93%	-14.29%	-6.43%
Only SEFV w/ EM	95.23	71.33	63.33	47.40
	-2.32%	-5.36%	-9.53%	-8.25%
<i>Ablation of EM within SEFV</i>				
Only SEFV w/o EM	94.72	70.96	63.33	46.92
	-0.54%	-0.52%	0.00%	-1.01%

ablation, we remove the routing mechanism and use only one of the two workflows. For the second, we remove routing and use only the Self-Evolving Formal Verify Model, evaluating settings with and without Evolution Memory. We use Qwen3.5-122B as the reasoning model. All variants use the same inference and evaluation setup, so the comparison isolates the contribution of each verification design.

Table 2 shows a consistent pattern: neither sin-

gle verification path matches the fused system. Across the four benchmarks, using only the formal or informal workflow causes relative accuracy drops from 0.52% to 14.29%, with the largest gap appearing on AIME 2025. This suggests that formal and informal verification provide complementary rather than redundant coverage. By routing between them, FIVER raises the effective capability ceiling. Removing Evolution Memory causes an average relative accuracy drop of 0.52% in this ablation, suggesting that stored successful proofs and failed attempts provide useful context for subsequent formal verification. Further budget-matched routing baselines, router-component ablations, and cross-dataset routing diagnostics are reported in Appendix D.

6 Conclusion

This paper presents FIVER, a step-level verification agent for LLM mathematical reasoning. FIVER combines Lean 4-based formal verification and dual-view informal debate, using a learnable router to select the verification path for each reasoning step. Its core design is to cover the union of formal and informal verification capabilities while sending steps in their overlap to the lower-cost informal tool, improving verification coverage while controlling cost.

We construct a step-level routing dataset with 933 verification steps and 1,866 tool-level records. Experiments on four benchmarks show that FIVER consistently outperforms pure reasoning, reward-model selection, and purely formal verification baselines across reasoning models from different families, demonstrating the effectiveness of this hybrid verification strategy.

Limitations

Although fusing formal verification and informal reasoning can combine their advantages to some extent, our method still has several limitations. The external tools used by the current FIVER agent are still coarse-grained, mainly centered on Lean 4 formal verification and informal reasoning. Future work can further refine the granularity and add more tool branches.

Although 1,866 tool-level records are efficient, the dataset size is limited by the cost of manual annotation. Further expanding the number of problem-property categories and subdividing difficulty levels more finely may provide more kinds

of verifiable intermediate steps and bring further gains. Our experiments use a reasoning-token budget of 8,192 for each reasoning-model generation. Evaluating FIVER and the baselines under larger reasoning budgets is left to future work.

The formal-informal boundary learned by the router reflects the current capabilities of the tools. As Lean 4 formalization ability and informal verification ability improve, the optimal routing strategy will also change. Designing a dynamic routing strategy that adapts to tool-capability evolution is an important future direction.

Ethical Considerations

This work improves the correctness of mathematical reasoning and does not involve human subjects or sensitive data. Step-level labels are manually annotated and may be affected by differences in annotator interpretation and boundary-case judgments. We reduce annotation noise through unified annotation guidelines, review of disagreement cases, and cross-checking.

References

- Chi-Min Chan, Weize Chen, Yusheng Su, Jianxuan Yu, Wei Xue, Shanghang Zhang, Jie Fu, and Zhiyuan Liu. 2023. [ChatEval: Towards better LLM-based evaluators through multi-agent debate](#). *arXiv preprint arXiv:2308.07201*.
- Guoxin Chen, Jing Wu, Xinjie Chen, Wayne Xin Zhao, Ruihua Song, Chengxi Li, Kai Fan, Dayiheng Liu, and Minpeng Liao. 2026. [ReForm: Reflective autoformalization with prospective bounded sequence optimization](#). In *International Conference on Learning Representations*.
- Lingjiao Chen, Matei Zaharia, and James Zou. 2024. [FrugalGPT: How to use large language models while reducing cost and improving performance](#). *Transactions on Machine Learning Research*.
- Leonardo de Moura and Sebastian Ullrich. 2021. [The Lean 4 theorem prover and programming language](#). In *Proceedings of the 28th International Conference on Automated Deduction (CADE-28)*.
- DeepSeek-AI. 2025a. [DeepSeek-R1: Incentivizing reasoning capability in LLMs via reinforcement learning](#). *arXiv preprint arXiv:2501.12948*.
- DeepSeek-AI. 2025b. [DeepSeek-V3.2: Pushing the frontier of open large language models](#). *arXiv preprint arXiv:2512.02556*.
- Dujian Ding, Ankur Mallick, Chi Wang, Robert Sim, Subhabrata Mukherjee, Victor Ruhle, Laks V. S. Lakshmanan, and Ahmed Hassan Awadallah. 2024.

- Hybrid LLM: Cost-efficient and quality-aware query routing. In *International Conference on Learning Representations*.
- Dujian Ding, Ankur Mallick, Shaokun Zhang, Chi Wang, Daniel Madrigal, Mirian Del Carmen Hipolito Garcia, Menglin Xia, Laks V. S. Lakshmanan, Qingyun Wu, and Victor Rühle. 2025. **BEST-Route: Adaptive LLM routing with test-time optimal compute**. In *International Conference on Machine Learning*.
- Yilun Du, Shuang Li, Antonio Torralba, Joshua B. Tenenbaum, and Igor Mordatch. 2023. **Improving factuality and reasoning in language models through multiagent debate**. *arXiv preprint arXiv:2305.14325*.
- Susan E. Embretson and Steven P. Reise. 2000. *Item Response Theory for Psychologists*. Lawrence Erlbaum Associates.
- Nigel Fernandez, Branislav Kveton, Ryan A. Rossi, Andrew S. Lan, and Zichao Wang. 2026. **RADAR: Reasoning-ability and difficulty-aware routing for reasoning LLMs**. *International Conference on Learning Representations*.
- Google. 2025. **Gemini 3: Introducing the latest Gemini ai model from Google**. <https://blog.google/products-and-platforms/products/gemini/gemini-3/>.
- Zhibin Gou, Zhihong Shao, Yeyun Gong, Yelong Shen, Yujiu Yang, Nan Duan, and Weizhu Chen. 2024. **CRITIC: Large language models can self-correct with tool-interactive critiquing**. *International Conference on Learning Representations*.
- Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. 2020. **Scaling laws for neural language models**. *arXiv preprint arXiv:2001.08361*.
- Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2023. **Let’s verify step by step**.
- Haohan Lin, Zhiqing Sun, Sean Welleck, and Yiming Yang. 2024. **Lean-STaR: Learning to interleave thinking and proving**. *arXiv preprint arXiv:2407.10040*.
- Yong Lin, Shange Tang, Bohan Lyu, Ziran Yang, Jui-Hui Chung, Haoyu Zhao, Lai Jiang, Yihan Geng, Jiawei Ge, Jingruo Sun, Jiayun Wu, Jiri Gesi, Ximing Lu, David Acuna, Kaiyu Yang, Hongzhou Lin, Yejin Choi, Danqi Chen, Sanjeev Arora, and Chi Jin. 2025. **Goedel-prover-v2: Scaling formal theorem proving with scaffolded data synthesis and self-correction**. *arXiv preprint arXiv:2508.03613*.
- Chengwu Liu, Ye Yuan, Yichun Yin, Yan Xu, Xin Xu, Zaoyu Chen, Yasheng Wang, Lifeng Shang, Qun Liu, and Ming Zhang. 2025a. **Safe: Enhancing mathematical reasoning in large language models via retrospective step-aware formal verification**. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics*.
- Chris Yuhao Liu, Liang Zeng, Yuzhen Xiao, Jujie He, Jiakai Liu, Chaojie Wang, Rui Yan, Wei Shen, Fuxiang Zhang, Jiacheng Xu, Yang Liu, and Yahui Zhou. 2025b. **Skywork-reward-v2: Scaling preference data curation via human-ai synergy**. *arXiv preprint arXiv:2507.01352*. Preprint; submitted Jul 2, 2025.
- Yougang Lyu, Xi Zhang, Xinhao Yi, Yuyue Zhao, Shuyu Guo, Wenxiang Hu, Jan Piotrowski, Jakub Kaliski, Jacopo Urbani, Zaiqiao Meng, Lun Zhou, and Xiaohui Yan. 2026. **EvoScientist: Towards multi-agent evolving AI scientists for end-to-end scientific discovery**. *arXiv preprint arXiv:2603.08127*.
- Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegrefe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. 2023. **Self-refine: Iterative refinement with self-feedback**. *arXiv preprint arXiv:2303.17651*.
- Isaac Ong, Amjad Almahairi, Vincent Wu, Wei-Lin Chiang, Tianhao Wu, Joseph E. Gonzalez, M Waleed Kadous, and Ion Stoica. 2024. **RouteLLM: Learning to route LLMs with preference data**. *arXiv preprint arXiv:2406.18665*.
- OpenAI. 2025. **GPT-5 system card**. <https://openai.com/index/gpt-5-system-card/>.
- Azim Ospanov, Zijin Feng, Jiacheng Sun, Haoli Bai, Xin Shen, and Farzan Farnia. 2025. **HERMES: Towards efficient and verifiable mathematical reasoning in LLMs**. *arXiv preprint arXiv:2511.18760*.
- Charles Packer, Sarah Wooders, Kevin Lin, Vivian Fang, Shishir G. Patil, Ion Stoica, and Joseph E. Gonzalez. 2023. **MemGPT: Towards LLMs as operating systems**. *arXiv preprint arXiv:2310.08560*.
- Joon Sung Park, Joseph C. O’Brien, Carrie J. Cai, Meredith Ringel Morris, Percy Liang, and Michael S. Bernstein. 2023. **Generative agents: Interactive simulacra of human behavior**. *arXiv preprint arXiv:2304.03442*.
- Qwen Team. 2026. **Qwen3.5: Towards native multimodal agents**. <https://huggingface.co/Qwen/Qwen3.5-35B-A3B> and <https://huggingface.co/Qwen/Qwen3.5-122B-A10B>.
- Z. Z. Ren, Zhihong Shao, Junxiao Song, Huajian Xin, Haocheng Wang, Wanbiao Zhao, Liyue Zhang, Zhe Fu, Qihao Zhu, Dejian Yang, Z. F. Wu, Zhibin Gou, Shirong Ma, Hongxuan Tang, Yuxuan Liu, Wenjun

- Gao, Daya Guo, and Chong Ruan. 2025. [DeepSeek-Prover-V2: Advancing formal mathematical reasoning via reinforcement learning for subgoal decomposition](#). *arXiv preprint arXiv:2504.21801*.
- James V. Roggeveen, Erik Y. Wang, Will Flintoft, Peter Donets, Lucy S. Nathwani, Nickolas Gutierrez, David Ettl, Anton Marius Graf, Siddharth Dandavate, Arjun Nageswaran, Raglan Ward, Ava Williamson, Anne Mykland, Kacper K. Migacz, Yijun Wang, Egemen Bostan, Duy Thuc Nguyen, Zhe He, Marc L. Descoteaux, Felix Yeung, Shida Liu, Jorge García Ponce, Luke Zhu, Yuyang Chen, Ekaterina S. Ivshina, Miguel Fernandez, Minjae Kim, Kennan Gumbs, Matthew Scott Tan, Russell Yang, Mai Hoang, David Brown, Isabella A. Silveira, Lavon Sykes, Ahmed Roman, William Freidenberg, Yiming Chen, Lucas Martin, Yixing Tang, Kelly Werker Smith, Hongyu Liao, Logan G. Wilson, Alexander Dazhen Cai, Andrea Elizabeth Biju, and Michael P. Brenner. 2025. [HARDMath2: A benchmark for applied mathematics built by students as part of a graduate class](#). *arXiv preprint arXiv:2505.11774*.
- Amrith Setlur, Chirag Nagpal, Adam Fisch, Xinyang Geng, Jacob Eisenstein, Rishabh Agarwal, Alekh Agarwal, Jonathan Berant, and Aviral Kumar. 2025. [Rewarding progress: Scaling automated process verifiers for LLM reasoning](#). In *International Conference on Learning Representations*.
- Noah Shinn, Federico Cassano, Edward Berman, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2023. [Reflexion: Language agents with verbal reinforcement learning](#). *arXiv preprint arXiv:2303.11366*.
- Tal Shnitzer, Anthony Ou, Mírian Silva, Kate Soule, Yuekai Sun, Justin Solomon, Neil Thompson, and Mikhail Yurochkin. 2024. [Large language model routing with benchmark datasets](#). In *Conference on Language Modeling*.
- Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. 2023a. [Voyager: An open-ended embodied agent with large language models](#). *arXiv preprint arXiv:2305.16291*.
- Haiming Wang, Mert Unsal, Xiaohan Lin, Mantas Baksys, Junqi Liu, Marco Dos Santos, Flood Sung, Marina Vinyes, Zhenzhe Ying, Zekai Zhu, Jianqiao Lu, Hugues de Saxcé, Bolton Bailey, Chendong Song, Chenjun Xiao, Dehao Zhang, Ebony Zhang, Frederick Pu, Han Zhu, Jiawei Liu, Jonas Bayer, Julien Michel, Longhui Yu, Léo Dreyfus-Schmidt, Lewis Tunstall, Luigi Pagani, Moreira Machado, Pauline Bourigault, Ran Wang, Stanislas Polu, Thibaut Barroyer, Wen-Ding Li, Yazhe Niu, Yann Fleureau, Yangyang Hu, Zhouliang Yu, Zihan Wang, Zhilin Yang, Zhengying Liu, and Jia Li. 2025a. [Kimina-Prover Preview: Towards large formal reasoning models with reinforcement learning](#). *arXiv preprint arXiv:2504.11354*.
- Peiyi Wang, Lei Li, Liang Chen, Zefan Cai, Dawei Zhu, Binghui Lin, Yunbo Cao, Qi Liu, Tianyu Liu, and Zhifang Sui. 2023b. [Large language models are not fair evaluators](#). *arXiv preprint arXiv:2305.17926*.
- Peiyi Wang, Lei Li, Zhihong Shao, Runxin Xu, Damai Dai, Yifei Li, Deli Chen, Yu Wu, and Zhifang Sui. 2024a. [Math-Shepherd: Verify and reinforce LLMs step-by-step without human annotations](#). In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics*.
- Ruida Wang, Yuxin Li, Yi R. Fung, and Tong Zhang. 2025b. [Let’s reason formally: Natural-formal hybrid reasoning enhances LLM’s math capability](#). *arXiv preprint arXiv:2505.23703*.
- Ruida Wang, Jipeng Zhang, Yizhen Jia, Rui Pan, Shizhe Diao, Renjie Pi, and Tong Zhang. 2024b. [TheoremLlama: Transforming general-purpose LLMs into Lean4 experts](#). *arXiv preprint arXiv:2407.03203*.
- An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. 2025. [Qwen3 technical report](#). *arXiv preprint arXiv:2505.09388*.
- Jiarui Yao, Ruida Wang, and Tong Zhang. 2025. [FANS: Formal answer selection for LLM natural language math reasoning using Lean4](#). In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*.
- Runyang You, Hongru Cai, Caiqi Zhang, Qiancheng Xu, Meng Liu, Tiezheng Yu, Yongqi Li, and Wenjie Li. 2026. [Agent-as-a-judge](#). *arXiv preprint arXiv:2601.05111*.
- Yanzhao Zhang, Mingxin Li, Dingkun Long, Xin Zhang, Huan Lin, Baosong Yang, Pengjun Xie, An Yang, Dayiheng Liu, Junyang Lin, Fei Huang, and Jingren Zhou. 2025a. [Qwen3 Embedding: Advancing text embedding and reranking through foundation models](#). *arXiv preprint arXiv:2506.05176*.
- Yifan Zhang and Math-AI Team. 2025. American Invitational Mathematics Examination (AIME) 2025. <https://huggingface.co/datasets/math-ai/aime25>.

Zhenru Zhang, Chujie Zheng, Yangzhen Wu, Beichen Zhang, Runji Lin, Bowen Yu, Dayiheng Liu, Jingren Zhou, and Junyang Lin. 2025b. [The lessons of developing process reward models in mathematical reasoning](#). In *Findings of the Association for Computational Linguistics: ACL*.

Andrew Zhao, Daniel Huang, Quentin Xu, Matthieu Lin, Yong-Jin Liu, and Gao Huang. 2024. [ExpeL: LLM agents are experiential learners](#). In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 19632–19642.

Jian Zhao, Runze Liu, Kaiyan Zhang, Zhimu Zhou, Junqi Gao, Dong Li, Jiafei Lyu, Zhouyi Qian, Biqing Qi, Xiu Li, and Bowen Zhou. 2025. [Genprm: Scaling test-time compute of process reward models via generative reasoning](#). *arXiv preprint arXiv:2504.00891*. Preprint; submitted Apr 1, 2025.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhonghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. 2023. [Judging LLM-as-a-judge with MT-Bench and Chatbot Arena](#). *Advances in Neural Information Processing Systems*, 36.

A Prompt Templates

This appendix lists the prompts used by different modules separately. Dynamic fields such as retrieved memory, historical compilation errors, and candidate problem text are represented as placeholders.

Formalizer Prompt

Please autoformalize the following natural language problem statement in Lean4.
Use the following theorem name: test
Useful verified memories:
{retrieved positive memories or (none)}
Previously failed translation attempts for similar steps.
Avoid repeating the same wrong formalization patterns and compiler/semantic errors:
{retrieved translation-failure memories or (none)}
Category-level failure reminders for this step type ({step_category}):
{category reminders or (none)}
Current natural language statement:
{candidate_step_nl}
Think before you provide the lean statement.

Semantic Translation Check Prompt

You need to do two tasks at once:
1. Check whether the Lean4 statement has the same mathematical meaning as the original step.
2. Classify the current step into one MATH-500 category.
Allowed categories (closed set):
Prealgebra, Algebra, Number Theory, Counting & Probability, Geometry, Intermediate Algebra, Precalculus
Original step:
{nl_statement}
Lean statement:
{lean_statement}
Return strict JSON only:
{ "same_meaning": "yes|no", "step_category": "<one allowed category>" }

Prover Prompt for the Goal

Complete the following Lean 4 code to prove the given theorem.
End with Lean 4 code.
Natural language step:
{candidate_step_nl}
~~~lean4  
<header>  
{lean\_statement}  
<body>  
---  
Previously failed proof attempts for similar steps.  
Do NOT repeat the same bad proof patterns or the same compiler errors:  
{retrieved prover errors for goal branch or (none)}  
Category-level failure reminders for this step type ({step\_category}):  
{category reminders for goal branch or (none)}  
Before producing the Lean 4 code to formally prove the given theorem,  
provide a detailed proof plan outlining the main proof steps and strategies.

### Answer Equivalence Judge Prompt

You are a mathematical answer checker.  
Step 1: Check whether the Ground Truth appears in the Answer. If the Ground Truth appears in the Answer, output True.  
Step 2: If the Ground Truth does not appear in the Answer, inspect the Answer for likely candidate answers. Check for "Final Answer" keywords, but do not rely only on them. If there are multiple candidate answers, choose the candidate that is closest to the Ground Truth.  
Step 3: Simplify or rewrite the selected candidate answer toward the Ground Truth, then judge whether they are mathematically equivalent. Do not reject a correct answer only because of term order, decimal vs fraction form,  $e^{i\theta}$  not being expanded, equivalent trigonometric or algebraic form, LaTeX formatting, whitespace, parentheses, renamed/reordered arbitrary constants such as C\_1 and C\_2, or reasonable rounded/approximate numeric forms.  
Step 4: Output only True or False.  
Answer:  
{Answer}  
Ground Truth:  
{Ground\_Truth}

### DeepSeek System Prompt

You are a mathematical verification assistant. Think privately if needed, but output only a single JSON object with no markdown fences.

#### DeepSeek Support-View Prompt

You are a mathematical verification assistant in SUPPORT mode.

Your task is to determine whether the given step can be justified from the problem and the verified context. Act like a careful prover, not a critic.

Problem:  
{problem}  
{retrieved memory section}

Step to verify:  
{candidate\_step}

Instructions:

1. Restate the exact mathematical claim made by the step.
2. Use only information explicitly available from the problem or previously verified steps.
3. Check whether the step can be justified by valid algebra, logic, or standard facts.
4. Identify all assumptions required for the step to be valid.
5. Do not invent missing assumptions.
6. If the step can be justified only under assumptions that are not explicitly available, mark it as uncertain.
7. If the step is clearly contradicted by the available information, mark it as invalid.

Return valid JSON only with keys:  
verdict, confidence, claim, assumptions\_used,  
critical\_checks, remaining\_risks, short\_rationale

Where:  
- verdict in ["valid", "invalid", "uncertain"]  
- confidence in [0,1]

#### DeepSeek Attack-View Prompt

You are a mathematical verification assistant in ATTACK mode.

Your task is to challenge the given step as aggressively as possible.

Act like a skeptical reviewer trying to find hidden flaws.

Problem:  
{problem}  
{retrieved memory section}

Step to verify:  
{candidate\_step}

Instructions:

1. Restate the exact mathematical claim made by the step.
2. Search for domain issues, missing assumptions, unjustified transformations, logical gaps, notation ambiguity, and hidden edge cases.
3. Try to construct a counterexample or a failure case whenever possible.
4. If the step is only conditionally valid, identify the missing condition.
5. Do not repair the argument.
6. If you find any unresolved risk that prevents rigorous acceptance of the step as written, do not mark it as valid.
7. Only mark it as valid if you fail to find any substantial issue.

Return valid JSON only with keys:  
verdict, confidence, attack\_type, critical\_issue,  
counterexample\_attempt, remaining\_risks,  
short\_rationale

Where:  
- verdict in ["valid", "invalid", "uncertain"]  
- confidence in [0,1]  
- attack\_type in ["domain", "logic", "algebra", "missing\_assumption", "counterexample", "none"]

## B Step-Level Dataset and Tool Capability Profiles

The router dataset consists of 933 verification steps generated from 102 MATH-500 seed problems. For each difficulty–mathematical-property cell, we sample three problems when available and otherwise include all problems in that cell. Qwen3-8B is used to generate solution trajectory

ries with verification calls. We extract intermediate steps from these trajectories, and each step is manually annotated for mathematical correctness for router supervision. The annotations were produced through multiple rounds of evaluation and review by master’s-level mathematics annotators over approximately two weeks. We then construct a router validation set by randomly sampling five steps from each subject–difficulty cell when available, and all steps from the cell otherwise. This yields 171 validation steps. The remaining 762 steps are used as the router training set.

In router supervision, each extracted step is independently processed by the two available verification workflows: the Self-Evolving Formal Verify Model and the Informal Debate Model, producing 1,866 tool-level verification records. This yields two labels for each step: whether the workflow successfully verifies the step and the measured latency of that verification attempt. These records also show that the two tools have different capability profiles. Figure 5 reports the independent success rates of 933 analyzed steps over MATH-500 subject and difficulty cells. Formal verification has an overall independent success rate of 58.1%, while informal verification has 54.6%. Formal verification is strongest on Geometry level 3, Counting and Probability level 4, and Prealgebra level 5. Informal verification is strongest on Counting and Probability level 5, Algebra level 4, and Intermediate Algebra level 2. These complementary profiles motivate step-level routing: neither verifier uniformly dominates across all cells, and the useful workflow depends on the mathematical domain and difficulty.

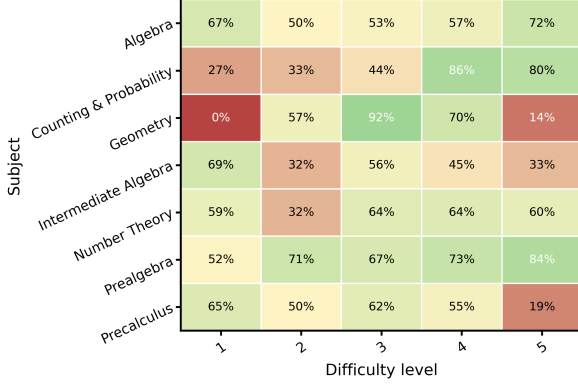
### B.1 Step-Level Verification Quality

We further evaluate verification quality on 171 steps from 84 held-out MATH-500 problems. Accuracy is computed over non-ambiguous steps; the false-acceptance rate (FAR) is the proportion of incorrect steps accepted as correct, and the false-rejection rate (FRR) is the proportion of correct steps not accepted as correct. Table 4 compares the two individual workflows with FIVER’s routed verification.

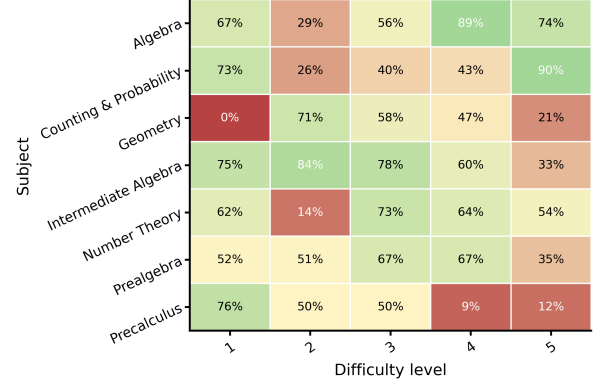
FIVER improves step-level accuracy from 55.56% and 62.57% to 69.59%, while reducing both false acceptances and false rejections relative to either verifier alone.

| Level | Algebra | Count./Prob. | Geometry | Inter. Alg. | Num. Theory | Prealg. | Precalc. |
|-------|---------|--------------|----------|-------------|-------------|---------|----------|
| 1     | 3       | 2            | 1        | 3           | 3           | 3       | 3        |
| 2     | 3       | 3            | 3        | 3           | 3           | 3       | 3        |
| 3     | 3       | 3            | 3        | 3           | 3           | 3       | 3        |
| 4     | 3       | 3            | 3        | 3           | 3           | 3       | 3        |
| 5     | 3       | 3            | 3        | 3           | 3           | 3       | 3        |
| Total | 15      | 14           | 13       | 15          | 15          | 15      | 15       |

Table 3: Difficulty–subject distribution of the 102 seed problems.



(a) Formal-verification capability profile.



(b) Informal-verification capability profile.

Figure 5: Tool capability profiles on 933 MATH-500-derived verification steps. Each cell is grouped by original subject and difficulty, and values indicate the percentage of steps independently verified by the corresponding tool.

| Data Source             | Prob. | Steps | Formal |       |       | Informal |       |       | FIVER        |              |              |
|-------------------------|-------|-------|--------|-------|-------|----------|-------|-------|--------------|--------------|--------------|
|                         |       |       | Acc.   | FAR   | FRR   | Acc.     | FAR   | FRR   | Acc.         | FAR          | FRR          |
| MATH-500 held-out steps | 84    | 171   | 55.56  | 45.45 | 39.60 | 62.57    | 45.45 | 32.89 | <b>69.59</b> | <b>36.36</b> | <b>26.17</b> |

Table 4: Step-level verification accuracy, false-acceptance rate, and false-rejection rate (%).

## C Router and Memory Details

This section supplements the training, normalization, and inference details of the Step Router in Section 3.2. We use the notation from Section 3.2: the step to verify is  $\tilde{q}_t$ , its embedding is  $e_t$ , and the tool set is

$$\mathcal{T} = \{\text{FORMAL}, \text{INFORMAL}\}.$$

The router is trained on step-level records and jointly estimates each verification workflow’s success probability  $p_i(\tilde{q}_t)$  and normalized latency cost  $c_i(\tilde{q}_t)$ . The default routing weight is  $\lambda = 0.34$ , selected on the router validation set by the subject–difficulty scan in Section F.1.

Each routed item is a step to verify  $\tilde{q}_t$  and its local context. The router first computes a fixed

step embedding:

$$e_t = \text{Encoder}(\tilde{q}_t) \in \mathbb{R}^d, \quad (11)$$

This is consistent with Eq. 3 in Section 3.2. The main experiments route between two tools: the Self-Evolving Formal Verify Model and the Informal Debate Model. For each tool  $i \in \mathcal{T}$ , the router predicts success probability  $p_i(\tilde{q}_t)$  and relative latency cost  $c_i(\tilde{q}_t)$ .

The precision predictor uses the IRT-style form in Eq. 4 of Section 3.2:

$$p_i(\tilde{q}_t) = \sigma\left(\beta_i - d(\tilde{q}_t) + h(\tilde{q}_t)^\top s_i\right), \quad (12)$$

where  $\beta_i$  is the learnable global ability of tool  $i$ ,  $d(\tilde{q}_t)$  is the step difficulty,  $h(\tilde{q}_t)$  is the step ability-demand representation, and  $s_i$  is the learnable expertise vector of tool  $i$ . Specifically, as in Eq. 5,

$$d(\tilde{q}_t) = |W_d e_t| + \epsilon, \quad h(\tilde{q}_t) = W_h e_t. \quad (13)$$

We set  $\epsilon = 0.1$  in experiments. The expertise matching term  $h(\tilde{q}_t)^\top s_i$  allows the router to judge not only overall step difficulty but also how well the step matches the Self-Evolving Formal Verify Model or the Informal Debate Model.

This branch is trained only with per-tool success labels:

$$\mathcal{L}_{\text{succ}} = \sum_i \text{BCE}(p_i(\tilde{q}_t), y_i), \quad (14)$$

This corresponds to Eq. 9 in Section 3.2. Here, BCE denotes binary cross-entropy loss, which measures the difference between the predicted success probability  $p_i(\tilde{q}_t)$  and the true tool success label  $y_i \in \{0, 1\}$ :

$$\begin{aligned} \text{BCE}(p_i(\tilde{q}_t), y_i) &= -y_i \log p_i(\tilde{q}_t) \\ &\quad - (1 - y_i) \log(1 - p_i(\tilde{q}_t)). \end{aligned} \quad (15)$$

When tool  $i$  actually succeeds, this loss encourages  $p_i(\tilde{q}_t)$  to approach 1. When tool  $i$  fails, it encourages  $p_i(\tilde{q}_t)$  to approach 0.

The latency branch is separated from the precision branch. For each tool  $i$ , the latency branch uses a tool-specific prediction head:

$$c_i(\tilde{q}_t) = \sigma(g_i(e_t) + b_i), \quad (16)$$

This is consistent with Eq. 6 in Section 3.2. Here,  $g_i$  is a tool-specific MLP head and  $b_i$  is a learnable base-latency logit. This branch is supervised by measured latency cost.

The cost supervision target is derived from the observed latency  $\ell_i$ . For all training samples and all tools with valid latency, we first compute

$$z_i(\tilde{q}_t) = \log(1 + \ell_i), \quad (17)$$

$$z_{\min} = \min_{t,i} z_i(\tilde{q}_t), \quad z_{\max} = \max_{t,i} z_i(\tilde{q}_t). \quad (18)$$

We then obtain the globally normalized log-latency target:

$$t_i = \text{clip}\left(\frac{z_i(\tilde{q}_t) - z_{\min}}{z_{\max} - z_{\min}}, 0, 1\right). \quad (19)$$

This normalization is shared across all tools, so the latency costs of the Self-Evolving Formal Verify Model and the Informal Debate Model can be directly compared.

Let the latency prediction residual be

$$\rho_i(\tilde{q}_t) = c_i(\tilde{q}_t) - t_i. \quad (20)$$

We use a smooth  $L_1$  loss with unit threshold:

$$\begin{aligned} &\text{SmoothL1}(c_i(\tilde{q}_t), t_i) \\ &= \begin{cases} \frac{1}{2} \rho_i(\tilde{q}_t)^2, & |\rho_i(\tilde{q}_t)| < 1, \\ |\rho_i(\tilde{q}_t)| - \frac{1}{2}, & |\rho_i(\tilde{q}_t)| \geq 1. \end{cases} \end{aligned} \quad (21)$$

Compared with squared error, this objective is less sensitive to large latency outliers while remaining differentiable near zero.

Finally, the cost loss is defined as the masked average over tools with valid latency labels:

$$\mathcal{L}_{\text{cost}} = \frac{\sum_i m_i \text{SmoothL1}(c_i(\tilde{q}_t), t_i)}{\sum_i m_i}, \quad (22)$$

This corresponds to Eq. 10 in Section 3.2. Here,  $m_i$  masks missing latency values.

The precision predictor and latency predictor are optimized separately:  $\mathcal{L}_{\text{succ}}$  updates only the precision branch, and  $\mathcal{L}_{\text{cost}}$  updates only the latency branch. At inference time, each tool receives the linear score in Eq. 7:

$$\text{score}_i(\tilde{q}_t) = \lambda \cdot p_i(\tilde{q}_t) - (1 - \lambda) \cdot c_i(\tilde{q}_t). \quad (23)$$

The router then selects according to Eq. 8:

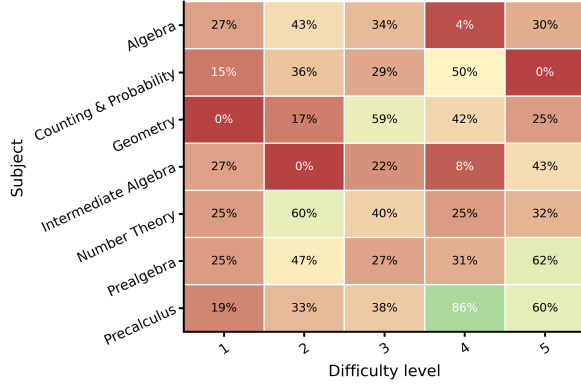
$$i^*(\tilde{q}_t) = \arg \max_i \text{score}_i(\tilde{q}_t). \quad (24)$$

**Memory state** Memory is used as verifier state: positive memory stores successful step information as conditions for the next proof, while negative memory stores translation failures, semantic mismatches, proof errors, failed negation branches, and category-level failure traces. Evolution Memory is reset at the beginning of each problem and updated only within that problem’s reasoning trajectory; it is never carried across evaluation problems or benchmarks.

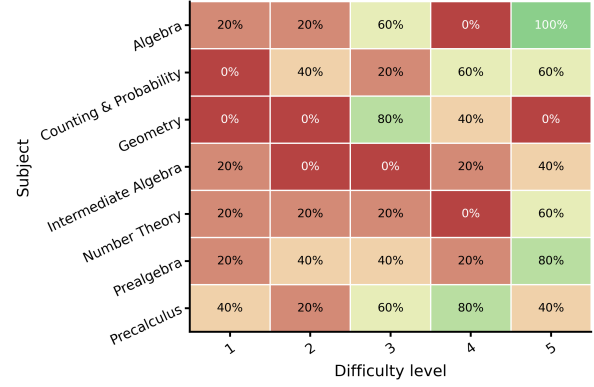
**Feedback semantics** The unified control state does not collapse the evidential distinction between the two workflows. Formal verification retains its Lean proof or negation-compilation result, whereas informal verification retains its supporting and attacking rationales; the shared PROVED, DISPROVED, or VERIFICATION FAILURE state only controls the subsequent reasoning process.

## D Router Effectiveness Evidence

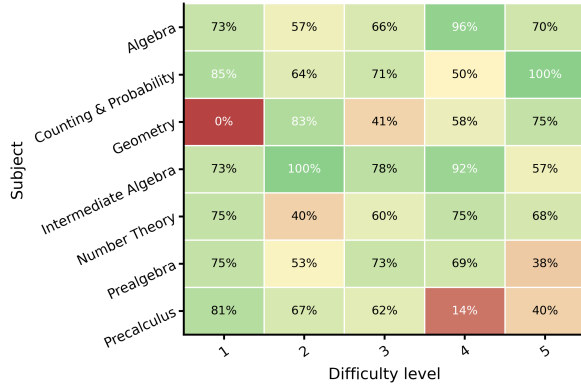
Figure 6 directly tests whether the router learns meaningful tool-assignment rules. The reference panels show the oracle assignment pattern on the 933-step supervision set: if both tools succeed, the



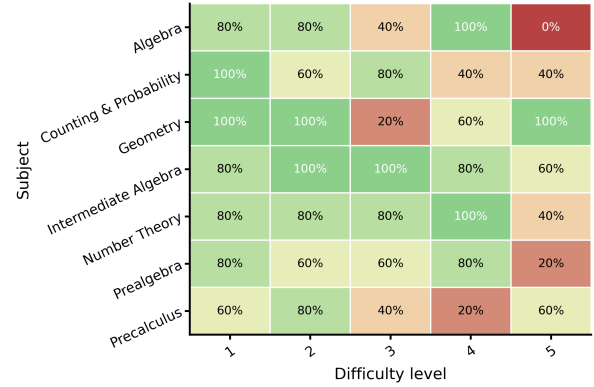
(a) Formal verification oracle assignment.



(b) Formal verification router selection.



(c) Informal verification oracle assignment.



(d) Informal verification router selection.

Figure 6: Router effectiveness evidence. Each row fixes one tool. The left panel shows oracle assignment over all 933 analyzed steps, and the right panel shows the learned router-selection pattern on the router validation set.

lower-latency tool is selected. If only one tool succeeds, that tool is selected. If both fail, the step is excluded from the oracle denominator. The selection panels show the actual routing choices for the selected  $\lambda=0.34$  on the router validation set from Appendix B. On the supervision set, the oracle assigns 32.6% of solvable steps to formal verification and 67.4% to informal verification. On the validation set, the corresponding oracle proportions are 28.1% and 71.9%, while the router selects formal verification for 33.3% of all validation steps and informal verification for 66.7%. This pattern suggests that the learned policy is not merely tracking a global preference for the more frequently successful workflow. The close overall proportions and cell-level alignment indicate that the router learns the relative strengths of the two tools rather than degenerating into a fixed workflow.

### D.1 Budget-Matched Routing Baselines

To isolate the effect of learned routing, we compare FIVER with random routing at a matched formal-call rate and with two fixed step-length

rules. All methods invoke exactly one verifier per step and therefore use the same single-verification budget.

| Method                                            | MATH-398     | MinervaMath  | AIME 2025    | HM2         |
|---------------------------------------------------|--------------|--------------|--------------|-------------|
| <b>FIVER</b>                                      | <b>88.44</b> | <b>69.85</b> | <b>40.00</b> | <b>6.64</b> |
| Random Routing (Matched Formal Rate)              | 79.65        | 65.44        | 23.33        | 4.26        |
| Length-based Routing (Long $\rightarrow$ Formal)  | 76.13        | 65.44        | 23.33        | 3.79        |
| Length-based Routing (Short $\rightarrow$ Formal) | 78.64        | 66.18        | 20.00        | 3.32        |

Table 5: Final-answer accuracy of FIVER and simple routing strategies with Qwen3-8B.

FIVER outperforms the strongest simple baseline on each benchmark by 8.79, 3.67, 16.67, and 2.38 percentage points, respectively, showing that its gains cannot be explained by the formal-call ratio or a fixed length rule alone.

### D.2 Router-Component Ablations

We next ablate the router’s IRT formulation, tool-step matching term, and two prediction objectives while keeping the supervision data fixed.

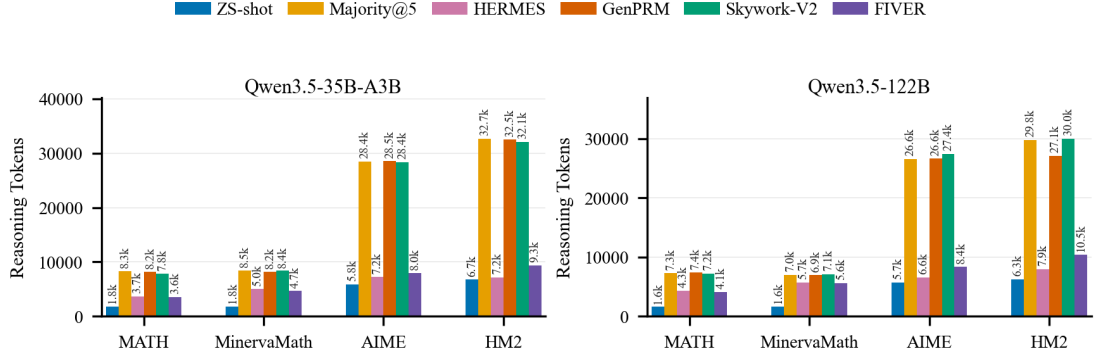


Figure 7: Average reasoning tokens per problem for Qwen3.5-35B and Qwen3.5-122B across datasets.

| Router Variant         | MATH-398     | MinervaMath  | AIME 2025    | HM2         |
|------------------------|--------------|--------------|--------------|-------------|
| FIVER Router (Full)    | <b>88.44</b> | <b>69.85</b> | <b>40.00</b> | <b>6.64</b> |
| w/o IRT Modeling       | 76.13        | 66.17        | 33.33        | 4.74        |
| w/o Tool-Step Matching | 77.64        | 64.34        | 30.00        | 3.79        |
| Success-only Objective | 77.14        | 65.44        | 30.00        | 4.27        |
| Cost-only Objective    | 76.63        | 63.24        | 26.67        | 3.79        |

Table 6: Router-component ablations with Qwen3-8B. All variants use the same supervision data.

The full router performs best on all four benchmarks, indicating that IRT modeling, tool-step matching, and the joint modeling of success and cost each contribute to effective routing.

### D.3 Cross-Dataset Router Generalization

We evaluate the trained router on held-out reasoning steps from three external benchmarks, none of which is used for router training. For each external benchmark, we sample 10 problems and extract seven steps per problem. Selected-tool success is computed over non-ambiguous steps; oracle match additionally excludes steps on which both tools fail, and formal selection is computed over all steps.

| Dataset          | Prob. | Steps | Selected-tool Success | Oracle Match | Formal Selection |
|------------------|-------|-------|-----------------------|--------------|------------------|
| MATH-500 holdout | 84    | 171   | 69.59                 | 61.40        | 33.33            |
| MinervaMath      | 10    | 70    | 71.43                 | 64.28        | 28.57            |
| AIME 2025        | 10    | 70    | 67.16                 | 65.57        | 24.29            |
| HM2              | 10    | 70    | 67.16                 | 71.19        | 15.71            |

Table 7: Step-level router generalization across datasets (%).

The selected-tool success and oracle-match rates on the three external datasets remain close to the MATH-500 holdout results, while the changing formal-selection rate shows that the router does not preserve a fixed global call ratio.

### D.4 Routing Frequencies on Full Benchmarks

Table 8 reports the verifier selected for every routing decision in the full Qwen3-8B benchmark runs.

| Dataset     | Formal Selection | Informal Selection |
|-------------|------------------|--------------------|
| MATH-398    | 33.15            | 66.85              |
| MinervaMath | 30.22            | 69.78              |
| AIME 2025   | 28.76            | 71.24              |
| HM2         | 20.13            | 79.87              |

Table 8: Formal and informal routing frequencies on the full benchmarks (%).

The distribution varies substantially with the benchmark, consistent with routing according to step characteristics rather than a fixed dataset-independent mixture.

### E Additional Reasoning-Token Results

The reasoning-token usage of Qwen3.5-35B and Qwen3.5-122B is shown in Figure 7.

### F Computational Cost Analysis

FLOPs measure the amount of computation behind different verification strategies. Because all models used in our experiments are open-source, we follow the estimation method of (Kaplan et al., 2020) and compute inference FLOPs for locally deployed models according to model size, attention structure, and average reasoning-token count. For MoE models, we use the activated parameter count to estimate the actual inference-stage computation.

As shown in Figure 8, FIVER uses fewer FLOPs overall than multi-sampling methods such as Majority@5, Skywork-V2, and GenPRM. For Qwen3-8B, FIVER averages about 139 TFLOPs,

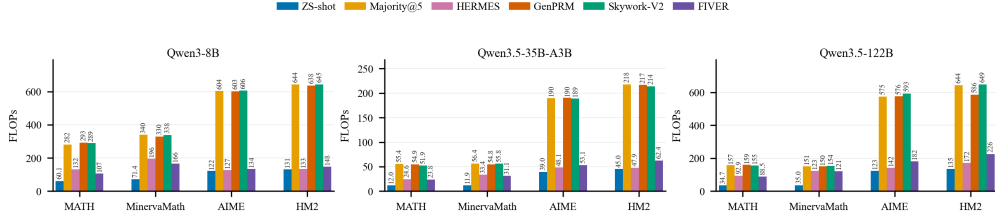


Figure 8: Average FLOPs per problem for Qwen3-8B, Qwen3.5-35B, and Qwen3.5-122B across datasets.

while Majority@5, Skywork-V2, and GenPRM all average around 460 TFLOPs. Meanwhile, FIVER also uses slightly less computation than HERMES, whose average is about 147 TFLOPs. This result shows that FIVER is computationally efficient at inference time. For Qwen3.5-35B and Qwen3.5-122B, FIVER outperforms HERMES on MATH-398 and MinervaMath but is slightly weaker on AIME 2025 and HM2. One possible reason is that FIVER has stronger verification ability and finds more errors, which causes the reasoning model to perform additional reasoning and slightly increases reasoning tokens.

### F.1 Routing-Weight Sensitivity

The router sends each step to a suitable workflow, and our oracle objective is to assign more steps in the overlap of formal and informal capabilities to the lower-cost workflow. The routing weight  $\lambda$  controls how much accuracy trade-off FIVER is willing to make when choosing cheaper verification. We scan  $\lambda$  on the router validation set to choose the value that best matches this oracle objective. The scan uses the validation set described in Appendix B, sweeping  $\lambda$  from 0.2 to 0.8 with step size 0.02.

Figure 9 shows that when  $\lambda=0.34$ , the oracle-cheapest hit rate is 61.40% and the router-selected step success rate is 69.59%. At this value, the agent preserves accuracy while preferring lower-cost verification when possible. Appendix D visualizes the resulting tool-assignment patterns over subject-difficulty cells. Thus, all experiments in this paper use the default linear-routing setting with  $\lambda=0.34$ .

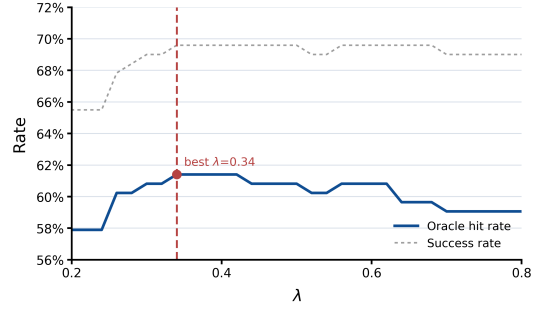


Figure 9: Sensitivity analysis of the routing weight  $\lambda$ .

## G Solution and Verification Process Example

### Verification Process for a math problem

We consider the singularly perturbed boundary value problem

$$\epsilon y'' - x^2 y' - (3 + x^3) = 0, \quad y(1) = 1, \quad y(2) = 1,$$

in the limit  $\epsilon \rightarrow 0^+$ . The objective is to obtain a uniformly valid leading order approximation and place the final solution in a boxed LaTeX expression.

#### Natural-language reasoning.

The model begins by inspecting the reduced problem obtained by setting  $\epsilon = 0$ :

$$-x^2 y'_0 - (3 + x^3) = 0, \quad y'_0 = -\frac{3 + x^3}{x^2}.$$

It also identifies the coefficient of the first derivative term as

$$a(x) = -x^2.$$

Since  $x \in [1, 2]$ , this coefficient is negative throughout the interval. The model first uses this sign information to reason about the boundary-layer location, and this algebraic part of the reasoning is selected for Lean 4 verification.

#### Selection of Lean 4 verification.

The router selects Lean4 for the first verification call. The formalized Lean4 statement verifies the sign of the first-derivative coefficient on  $[1, 2]$  and the endpoint coefficient values:

$$-1^2 = -1, \quad -2^2 = -4.$$

Lean 4 successfully proves the formalized statement with the following code:

```
import Mathlib
noncomputable section
theorem test :
  (∀ x : ℝ, 1 ≤ x → x ≤ 2 → -(x^2) < 0)
  ∧ (-(1 : ℝ)^2 = -1)
  ∧ (-(2 : ℝ)^2 = -4) := by
  constructor
  · intro x hx1 hx2
    have hxpos : 0 < x := lt_of_lt_of_le zero_lt_one hx1
    have hx2pos : 0 < x^2 := sq_pos_of_ne_zero (ne_of_gt hxpos)
    nlinarith
  constructor
  · norm_num
  · norm_num
```

The Lean 4 verifier returns **PROVED**. This confirms the coefficient-sign facts, while the endpoint interpretation still needs to be checked by the ongoing asymptotic reasoning.

#### Continuation of natural-language reasoning.

After the Lean 4 verification, the model continues by testing the implied left-endpoint layer. If a left layer is introduced with  $\xi = (x - 1)/\epsilon$ , the leading-order inner equation becomes

$$Y'' - Y' = 0,$$

whose nonconstant mode  $e^\xi$  grows as  $\xi \rightarrow \infty$ . Hence it cannot form a valid decaying boundary layer. Rechecking the right endpoint with

$$\xi = \frac{2 - x}{\epsilon}$$

gives the leading-order inner equation

$$Y'' + 4Y' = 0,$$

whose decaying mode is  $e^{-4\xi}$ . The model therefore corrects the earlier interpretation and concludes that the boundary layer is at  $x = 2$ . This corrected step is sent to DeepSeek.

#### DeepSeek verification.

The router selects DeepSeek for the second verification call. DeepSeek evaluates the corrected boundary-layer-location step using support and attack views.

#### Support view JSON:

```
{
  "verdict": "valid",
  "confidence": 0.95,
  "claim": "Since a(x)=-x^2 is negative on [1,2], the boundary
    layer is located at the right endpoint x=2. Consequently,
    the outer solution must satisfy y(1)=1.",
  "critical_checks": [
    "a(x)=-x^2 is negative on [1,2].",
```

```

    "The outer solution satisfies the left boundary condition and
    leaves the right boundary to be corrected by the layer."
  ],
  "remaining_risks": [
    "The sign rule relies on standard singular perturbation reasoning."
  ],
  "short_rationale": "The corrected endpoint choice is consistent."
}

```

**Attack view JSON:**

```

{
  "verdict": "valid",
  "confidence": 1.0,
  "attack_type": "none",
  "critical_issue": "",
  "counterexample_attempt": "",
  "remaining_risks": [],
  "short_rationale": "No counterexample is found; the corrected
    boundary-layer location is consistent with the
    reduced equation and boundary data."
}

```

**Final reasoning after verification.**

The tool report confirms the corrected boundary-layer location. The outer solution is determined from

$$y_0' = -\frac{3}{x^2} - x,$$

so

$$y_0(x) = \frac{3}{x} - \frac{x^2}{2} + C.$$

Since the boundary layer is at  $x = 2$ , the outer solution satisfies  $y(1) = 1$ :

$$1 = 3 - \frac{1}{2} + C, \quad C = -\frac{3}{2}.$$

Thus

$$y_0(x) = \frac{3}{x} - \frac{x^2}{2} - \frac{3}{2}.$$

At the right endpoint,

$$y_0(2) = \frac{3}{2} - 2 - \frac{3}{2} = -2,$$

whereas the boundary condition requires  $y(2) = 1$ . The boundary-layer correction must therefore have amplitude 3. With  $\xi = (2 - x)/\epsilon$ , the leading-order inner equation is

$$Y'' + 4Y' = 0,$$

and hence

$$Y_0(\xi) = A + Be^{-4\xi}.$$

Matching gives  $A = -2$ , and the boundary condition  $Y_0(0) = 1$  gives

$$-2 + B = 1, \quad B = 3.$$

The uniformly valid leading-order approximation is therefore

$$y_{\text{unif}}(x) = \frac{3}{x} - \frac{x^2}{2} - \frac{3}{2} + 3e^{-4(2-x)/\epsilon}.$$

Thus,

$$y(x) \sim \frac{3}{x} - \frac{x^2}{2} - \frac{3}{2}$$

as  $\epsilon \rightarrow 0^+$ . This concludes the verification process.

## H Additional Evaluation Results

### H.1 Multi-Seed Uncertainty

For each reasoning model and method, we run five inference seeds on MinervaMath and AIME 2025 and report mean accuracy and standard deviation.

| Method                   | MinervaMath                       | AIME 2025                         |
|--------------------------|-----------------------------------|-----------------------------------|
| <i>Qwen3-8B</i>          |                                   |                                   |
| Zero-shot                | 47.30 $\pm$ 1.5                   | 10.70 $\pm$ 2.8                   |
| Majority@5               | 64.00 $\pm$ 1.5                   | 25.30 $\pm$ 3.0                   |
| Skywork-V2               | 66.30 $\pm$ 1.5                   | 30.70 $\pm$ 2.8                   |
| GenPRM                   | 62.80 $\pm$ 1.9                   | 28.70 $\pm$ 3.8                   |
| HERMES                   | 66.90 $\pm$ 0.8                   | 28.00 $\pm$ 3.0                   |
| <b>FIVER</b>             | <b>69.90 <math>\pm</math> 1.1</b> | <b>40.70 <math>\pm</math> 2.8</b> |
| <i>Qwen3.5-35B</i>       |                                   |                                   |
| Zero-shot                | 73.60 $\pm$ 1.4                   | 68.00 $\pm$ 5.6                   |
| Majority@5               | 76.91 $\pm$ 1.1                   | 74.00 $\pm$ 4.4                   |
| Skywork-V2               | 75.29 $\pm$ 1.3                   | 72.00 $\pm$ 3.8                   |
| GenPRM                   | 75.66 $\pm$ 1.1                   | 70.67 $\pm$ 4.9                   |
| HERMES                   | 76.91 $\pm$ 1.4                   | 70.00 $\pm$ 5.3                   |
| <b>FIVER</b>             | <b>79.26 <math>\pm</math> 1.2</b> | <b>82.00 <math>\pm</math> 3.8</b> |
| <i>Qwen3.5-122B</i>      |                                   |                                   |
| Zero-shot                | 73.23 $\pm$ 1.5                   | 60.67 $\pm$ 4.4                   |
| Majority@5               | 72.50 $\pm$ 1.7                   | 61.33 $\pm$ 3.8                   |
| Skywork-V2               | 72.50 $\pm$ 1.3                   | 66.00 $\pm$ 2.8                   |
| GenPRM                   | 70.22 $\pm$ 1.7                   | 68.00 $\pm$ 3.0                   |
| HERMES                   | 70.81 $\pm$ 1.5                   | 62.00 $\pm$ 1.8                   |
| <b>FIVER</b>             | <b>76.55 <math>\pm</math> 1.2</b> | <b>74.67 <math>\pm</math> 3.8</b> |
| <i>DeepSeek-V4-Flash</i> |                                   |                                   |
| Zero-shot                | 85.44 $\pm$ 0.8                   | 54.00 $\pm$ 2.8                   |
| Majority@5               | 86.69 $\pm$ 0.7                   | 62.00 $\pm$ 3.8                   |
| Skywork-V2               | 86.76 $\pm$ 0.6                   | 65.33 $\pm$ 5.1                   |
| GenPRM                   | 84.93 $\pm$ 0.9                   | 65.33 $\pm$ 3.0                   |
| HERMES                   | 85.29 $\pm$ 0.6                   | 60.67 $\pm$ 2.8                   |
| <b>FIVER</b>             | <b>88.97 <math>\pm</math> 0.6</b> | <b>72.00 <math>\pm</math> 3.0</b> |

Table 9: Five-seed mean accuracy and standard deviation across reasoning models (%).

Across the five seeds, FIVER has the highest mean accuracy on both datasets under all four reasoning models.

### H.2 Test-Time Scaling

Cumulative accuracy counts a problem as correct if any of the first  $k$  independently sampled Qwen3-8B trajectories is correct.

| Method  | MATH-398     | MinervaMath  | AIME 2025    | HM2         |
|---------|--------------|--------------|--------------|-------------|
| FIVER@1 | 88.44        | 69.85        | 40.00        | 6.64        |
| FIVER@2 | 88.69        | 70.22        | 40.00        | 6.64        |
| FIVER@3 | 88.94        | 70.22        | 40.00        | 7.11        |
| FIVER@4 | 89.20        | 70.59        | 40.00        | 7.11        |
| FIVER@5 | 89.45        | 70.96        | 43.33        | 7.58        |
| FIVER@6 | <b>89.70</b> | <b>71.32</b> | <b>43.33</b> | <b>7.58</b> |
| FIVER@7 | <b>89.70</b> | <b>71.32</b> | <b>43.33</b> | <b>7.58</b> |
| FIVER@8 | <b>89.70</b> | <b>71.32</b> | <b>43.33</b> | <b>7.58</b> |

Table 10: Cumulative FIVER accuracy with Qwen3-8B (%).

From FIVER@1 to FIVER@8, cumulative accuracy improves by 1.26, 1.47, 3.33, and 0.94 percentage points on MATH-398, MinervaMath, AIME 2025, and HM2, respectively.