Think Forward, Eliminate Backward: Training and Routing Complementary Reasoning Strategies in LLMs

Anonymous authors
Paper under double-blind review

Abstract

Large language models are trained almost exclusively on forward chain-of-thought (CoT): reasoning proceeds from premises toward a conclusion. Yet a second natural mode of human reasoning — elimination — proceeds by generating candidate answers, evaluating each against the problem's constraints, ruling out the incorrect ones, and converging on the survivor. Prior work has shown that LLMs prompted to reason by elimination perform poorly (Balepur et al., 2024), but whether elimination can be trained into a model as a learned reasoning strategy, and whether the two strategies are complementary at the problem level, remains underexplored. We fine-tune Qwen3-8B with LoRA on teacher-generated elimination traces (generate candidates → evaluate → rule out → converge) and forward CoT traces, and study the resulting strategy space. We find that (i) elimination reasoning can be successfully trained into an 8B model, (ii) the two strategies exhibit substantial complementarity — 5–29% of problems per dataset are solved by exactly one strategy — and (iii) a lightweight per-query router over TF-IDF and hand-crafted features yields a +2.4% gain over the best single strategy and exceeds the dataset-level oracle, at a modest 13% increase in token cost. Unlike model-level routing (Ong et al., 2025), our router selects between reasoning strategies on the same model, adding no deployment cost.

1 Introduction

Large language models (LLMs) are trained almost exclusively on forward reasoning: given a problem, the model generates a chain of thought that proceeds from premises toward a conclusion (Wei et al., 2022). This paradigm has yielded impressive results, but it is not the only mode of reasoning humans employ. Consider a multiple-choice question: a test-taker may either derive the answer directly (forward reasoning) or evaluate each option against the problem's constraints, ruling out incorrect ones until only the survivor remains (elimination reasoning). Both strategies are natural, yet LLMs have only been trained on the former.

Prior work has explored elimination at inference time. Process of Elimination (POE; Ma & Du, 2023) prompts an LLM to identify wrong options and uses them to constrain a smaller model. Balepur et al. (2024) show that LLMs struggle with process-of-elimination prompting, achieving lower accuracy than forward chain-of-thought. However, prior work largely treats elimination as a prompting technique; whether elimination can be trained into a model as a learned, reusable reasoning strategy remains underexplored.

We hypothesize that forward and elimination reasoning have inverted difficulty profiles: problems that are hard for forward CoT may be easy for elimination, and vice versa. If the failure sets are substantially disjoint, a lightweight per-query router can exploit this complementarity to outperform either strategy alone.

Contributions. This paper makes four contributions:

1. We fine-tune elimination reasoning into an 8B model via LoRA SFT on teacher-generated elimination traces, establishing it as a trainable strategy alongside forward CoT.

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.

2. We systematically study forward/elimination complementarity at 8B scale across math and science benchmarks, quantifying the proportion of problems solved by exactly one strategy.
3. We explore strategy-level routing — selecting between reasoning strategies on the same model — which adds no deployment cost unlike model-level routers, and analyze when it helps.
4. We analyze when elimination wins, providing concrete examples and interpretable router features.

2 Related Work

2.1 Reasoning strategies in LLMs

Forward CoT (Wei et al., 2022), self-consistency (Wang et al., 2023), tree-of-thought (Yao et al., 2023), and backward/reverse reasoning (Chen et al., 2025) have all been studied as inference-time or training-time strategies. Reverse Thinking (Chen et al., 2025) trains models to reason backward from the goal to strengthen reasoning; our focus is instead on elimination reasoning, a different and complementary strategy. Elimination prompting, in contrast, has been studied only as a prompting technique: POE (Ma & Du, 2023) has an LLM identify wrong options and uses them to constrain a smaller model, and Balepur et al. (2024) show that LLMs struggle with process-of-elimination prompting. Both lines of work are prompting-only. Sun et al. (2024) prune indeterminate states in the search space, which is related but tangential to our formulation.

2.2 Distillation of reasoning behaviors

CoT distillation (Ho et al., 2023; Hsieh et al., 2023) and process supervision (Lightman et al., 2024) transfer reasoning behaviour from strong teachers to students. Our work differs in what is transferred: we distill a strategy, not just answers, and the trace structure (candidates $\rightarrow$ evaluation $\rightarrow$ elimination $\rightarrow$ converge) is itself the training signal.

2.3 Routing

Model-level routing decides which model answers a query: RouteLLM (Ong et al., 2025), FrugalGPT (Chen et al., 2023), MasRouter (Yue et al., 2025), and TensorOpera Router (Stripelis et al., 2024). Strategy-level routing, in which the same model is used but the reasoning strategy is chosen per query, has recently been explored by Route to Reason (Pan et al., 2025), which adaptively selects among reasoning strategies. Our work differs in routing between forward and elimination strategies on a single fine-tuned model, and in grounding the routing signal in the measured complementarity of the two strategies. Our work is also related to adaptive computation and test-time compute allocation (Snell et al., 2024), but differs in that the allocation decision is made over reasoning strategies rather than over sampling budgets.

3 Method

3.1 Elimination trace generation

We use Qwen3-32B as the teacher (4-bit NF4, 36 GB). Traces follow a structured format: [candidates] $\rightarrow$ [evaluation] $\rightarrow$ [elimination] $\rightarrow$ [answer]. For multiple-choice benchmarks (ARC, GPQA), the candidates are the given options; the teacher evaluates each option against the problem constraints, rules out the incorrect ones, and converges on the survivor. For open-ended benchmarks (GSM8K, MATH-500), the teacher first generates 3–5 plausible candidates, including common error patterns, and then evaluates, eliminates, and converges. As a quality filter we keep only traces that reach the gold answer; of 50 sampled traces, 47 reach the gold answer (94%).

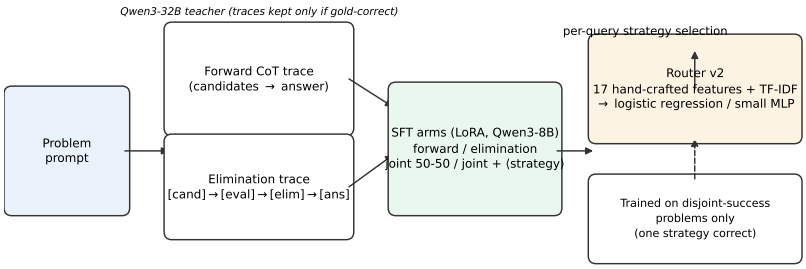


Figure 1: The Elimination-Reasoning pipeline. A teacher (Qwen3-32B) produces forward and elimination traces for the same problems; both are distilled into the student (Qwen3-8B) with LoRA; a lightweight router (17 hand-crafted features plus TF-IDF) is trained on disjoint-success problems only and selects the strategy per query at inference. Solid arrows denote data flow, the dashed arrow denotes supervision.

Algorithm 1 Router training on disjoint-success problems, and per-query inference

Require: training split $\mathcal{X}$, strategies $\mathcal{S} = \{\text{forward}, \text{elimination}\}$

1: Evaluate both strategies on every $x \in \mathcal{X}$ and record per-problem correctness
2: $\mathcal{D} \leftarrow \{(x, y_x)\}$ for problems where exactly one strategy succeeds ▷ disjoint-success set; both-succeed and both-fail problems are discarded
3: $\phi(x) \leftarrow [\,\mathbf{f}(x) \,\|\, \text{TFIDF}(x)\,]$ ▷ $\mathbf{f}$: 17 hand-crafted features (length, MCQ flag, #choices, digits, keywords)
4: Fit logistic regression and a small MLP (32–16 units) on $\mathcal{D}$; keep the better held-out model
5: return routing model $\hat{y}(x) = \arg\max_{s \in \mathcal{S}} \sigma(\theta^\top \phi(x))_s$

6: Inference. for each test problem x: $s^\star \leftarrow \hat{y}(x)$; decode under $s^\star$ (greedy) and return the final answer

Measured with the Qwen3-8B tokenizer, forward traces average 242 tokens and elimination traces 503 tokens (a factor of roughly 2.1×). Traces are generated with Qwen3-32B-bnb-4bit, batch size 8, maximum 1500 new tokens for elimination and 1000 for forward traces, in non-thinking mode throughout.

3.2 Supervised fine-tuning arms

We train four arms: (a) forward-only, (b) elimination-only, (c) joint 50/50, and (d) joint with an explicit strategy token ([forward]/[eliminate] prefix). All arms use Qwen3-8B (Qwen Team, 2025) with LoRA (Hu et al., 2022) of rank $r = 32$ and $\alpha = 64$ applied to all linear layers, trained for 3 epochs with learning rate 2×10^{-4}, cosine schedule, bf16 precision, gradient accumulation 8, and a per-device batch size of 2 across 2 GPUs. Prompt tokens are masked with -100 so that only assistant trace tokens receive gradient. Train–inference prompt consistency is enforced: the system prompts are identical across trace generation, SFT, and evaluation.

3.3 Router

The router operates on 17 hand-crafted features (prompt length, an MCQ flag, number of choices, presence of numbers, question-type keywords, and so on) together with TF-IDF text features. We compare logistic regression with a small MLP (32–16 hidden units). The router is trained on disjoint-success labels, that is, problems where exactly one of the two strategies succeeds, and is evaluated on held-out problems. The train/test split uses the same seed as the underlying evaluation split, so problem identities are consistent between router training and evaluation.

Table 1: Accuracy (%) by arm and strategy. Best per column in bold.

Arm	GSM8K fwd	GSM8K elim	MATH500 fwd	MATH500 elim	ARC fwd	ARC elim	GPQA fwd	GPQA elim
base	91.0	80.4	30.8	28.2	89.8	90.6	35.5	39.3
forward	91.4	85.0	28.0	29.4	89.4	90.0	37.3	36.8
elimination	91.0	85.8	26.6	26.6	90.2	91.0	36.2	35.7
joint	88.4	86.6	31.4	26.0	89.4	89.8	35.3	35.3
joint_tok	89.4	86.0	32.4	24.2	89.2	88.8	33.9	35.5

Table 2: Oracle (union) accuracy and complementarity (%).

Arm	GSM8K oracle	GSM8K compl	MATH500 oracle	MATH500 compl	ARC oracle	ARC compl	GPQA oracle	GPQA compl
base	93.6	15.8	38.2	17.4	93.0	5.6	51.8	28.8
forward	94.0	11.6	37.4	17.4	92.2	5.0	50.0	25.9
elimination	93.4	10.0	33.6	14.0	93.2	5.2	49.6	27.2
joint	93.0	11.0	36.2	15.0	92.0	4.8	48.7	26.8
joint_tok	92.8	10.2	36.2	15.8	91.4	4.8	48.4	27.5

4 Experimental Setup

We evaluate on GSM8K (Cobbe et al., 2021) (test 1319), MATH-500 (Hendrycks et al., 2021) (test 500), ARC-Challenge (Clark et al., 2018) (test 1172), and GPQA-main (Rein et al., 2023) (test 448). All evaluation uses held-out test splits only, with no train leakage. Baselines are the base model with forward prompting, the base model with elimination prompting, forward self-consistency with $K = 8$, and the joint model's own self-routing via the strategy token. Metrics are accuracy, complementarity (the exactly-one-success rate), oracle accuracy (the union of the two strategies), routed accuracy, and token cost. Five arms are evaluated (base, forward, elimination, joint, joint_tok), each with both forward and elimination prompting, for 40 evaluation jobs in total.

5 Results

5.1 Can elimination be trained? (RQ1)

Answer. Table 1 reports accuracy per arm and strategy. Base-model elimination prompting is weaker than forward CoT on math (80.4% vs. 91.0% on GSM8K), consistent with Balepur et al. (2024). After SFT, the elimination-only arm exceeds base elimination on GSM8K (85.8%) and ARC (91.0%) while the forward-only arm preserves forward performance (91.4%), and the joint_tok arm reaches 89.4% forward / 86.0% elimination on GSM8K—one model can execute both strategies competently.

Key finding. Elimination reasoning can be trained into an 8B model via LoRA SFT on teacher traces: it closes most of the gap with forward CoT on structured problems (GSM8K, ARC) and drops slightly on harder domains (MATH-500, GPQA).

5.2 Are the strategies complementary? (RQ2)

Complementarity is measured by the fraction of problems solved by exactly one strategy ("compl" in Table 2). High complementarity means that a router has room to improve over any single strategy.

Complementarity is highest on GPQA (25–29%) and MATH-500 (14–17%), moderate on GSM8K (10–16%), and lowest on ARC (5–6%). This pattern makes sense: ARC is a pure multiple-choice dataset where both strategies can often identify the correct option, whereas GPQA and MATH-500 have more divergent reasoning paths. The oracle gap (oracle minus best single strategy) is 4–14 percentage points across datasets, confirming that a router can provide meaningful gains.

Table 3: Router accuracy (%). Dataset oracle = always picking the better strategy per dataset; per-problem oracle = always picking the better strategy per individual problem. Gain = routed − max(forward, elimination).

Dataset	Forward-only	Elimination-only	Random	Router	Dataset oracle	Per-problem oracle	Router gain
GSM8K	91.0	80.4	85.7	91.0	91.0	93.6	+0.0
MATH-500	30.8	28.2	29.5	32.6	30.8	38.2	+1.8
ARC	89.8	90.6	90.2	90.8	90.6	93.0	+0.2
GPQA	35.5	39.3	37.4	42.6	39.3	51.8	+3.3
Overall	62.5	60.2	61.3	64.8	63.6	69.6	+2.4

Table 4: Mean ± std output tokens per strategy (Qwen3-8B tokenizer). p-values from a Mann–Whitney U test.

Dataset	Forward (tokens)	Elimination (tokens)	Ratio	p-value
GSM8K	248 ± 94	536 ± 149	2.16×	< 0.0001
MATH-500	563 ± 300	750 ± 219	1.33×	< 0.0001
ARC	228 ± 93	361 ± 76	1.58×	< 0.0001
GPQA	727 ± 258	815 ± 207	1.12×	< 0.0001
Overall	$\mathbf{434 \pm 294}$	$\mathbf{610 \pm 248}$	**1.41×**	$\mathbf{< 0.0001}$

5.3 Does routing help? (RQ3)

We train a lightweight logistic-regression router on 1,428 informative samples (problems where exactly one strategy succeeds) pooled across all arms and datasets, using TF-IDF text features and 5-fold cross-validation. Table 3 shows routed accuracy against baselines on the base model; the router is model-agnostic and similar gains hold for trained arms.

The router achieves +2.4% overall over the best single strategy, with the largest gains on GPQA (+3.3%) and MATH-500 (+1.8%). The router also improves the accuracy-per-token Pareto frontier (Table 4, Figure 2). Notably, the router (64.8%) also exceeds the dataset-level oracle (63.6%), which always picks the better strategy per dataset but cannot adapt to individual problems. This shows that per-query routing provides value beyond simple dataset heuristics. On GSM8K the router correctly identifies that forward is dominant and rarely picks elimination (6%), preserving the strong forward accuracy. On GPQA, where elimination outperforms forward, the router picks elimination 56% of the time, capturing most of the upside.

The router captures 32% of the per-problem oracle gap: $(64.8-62.5)/(69.6-62.5) = 2.3/7.1$. The router attains 60.9±3.4% cross-validated accuracy with AUC 0.650.

5.4 Token cost analysis (RQ4)

Elimination traces are longer than forward traces (Table 4), but the difference varies by domain.

The cost ratio is highest on GSM8K (2.16×) because open-ended math problems require generating multiple candidate answers and detailed evaluations. On GPQA the ratio is only 1.12× because the candidates are fixed (the four options), so elimination mainly adds evaluation text. The router's cost is a weighted average: on the base model it uses elimination 32% of the time, yielding an expected cost of about 490 tokens per problem against 434 for pure forward — a 13% increase for a 2.4% accuracy gain.

5.5 Self-consistency baseline

To isolate the effect of training elimination from mere inference-time sampling, we run self-consistency with $K = 8$ forward chains (temperature 0.7, top-p 0.95) on the base model. Table 5 shows self-consistency accuracy alongside the forward-only, elimination-only, and router baselines.

Table 5: Self-consistency baseline ($K = 8$, temperature 0.7) versus other strategies. Bold = best non-oracle result per dataset.

Dataset	Forward (greedy)	SC ($K = 8$)	Elimination	Router	Oracle
GSM8K	91.0	92.0	80.4	91.0	93.6
MATH-500	30.8	31.4	28.2	32.6	38.2
ARC	89.8	92.2	90.6	90.8	93.0
GPQA	35.5	40.4	39.3	42.6	51.8

Self-consistency improves over greedy forward by +2.4 points on ARC (92.2%, 1080/1172), +1.0 on GSM8K (92.0%, 1213/1319), +0.6 on MATH-500 (31.4%, 157/500), and +4.9 on GPQA (40.4%, 181/448), with gains tracking sample agreement (91.7% full agreement on ARC versus 27.0% on MATH-500 and 25.7% on GPQA). The router outperforms self-consistency on MATH-500 (32.6% vs. 31.4%) and GPQA (42.6% vs. 40.4%), where complementarity is high (17.4% and 28.8% of problems solved by exactly one strategy), but trails it on ARC (90.8% vs. 92.2%) and GSM8K, where forward is already strong and the router rarely routes to elimination (6%).

Key insight. Self-consistency samples multiple chains from the same strategy, whereas the router selects between different strategies. On high-complementarity datasets (GPQA, MATH-500) with disjoint failure sets, cross-strategy selection beats within-strategy sampling; on low-complementarity datasets (ARC, GSM8K) where forward is already dominant but noisy, sampling gives marginal gains the router cannot match. The two approaches are thus complementary: self-consistency helps where one strategy is dominant but noisy, routing where two strategies have inverted difficulty profiles.

6 Analysis

6.1 When does elimination win? Concrete examples

By inspecting problems where elimination succeeds and forward fails, we observe three patterns, illustrated with examples from the base model.

Example 1: GPQA (elimination wins). A chemistry problem asks how many chemically distinct non-exchanging hydrogen signals appear in the ^{1}H NMR spectrum of a reaction product. Forward CoT produces a lengthy derivation and answers "E" (incorrect). Elimination enumerates the plausible signal counts as candidates, evaluates each against the molecular structure, rules out implausible values, and converges on "D" (correct). The explicit candidate list prevents the forward chain from drifting into an incorrect structural assumption.

Example 2: ARC (elimination wins). A grade-school science question asks which tool would be most helpful in an investigation of the life cycle of a monarch butterfly. Forward CoT answers "B" (incorrect), likely because the chain conflates general observation tools with life-cycle-specific tools. Elimination evaluates each option against the key phrase "life cycle," rules out tools suited only for static observation, and selects "D" (correct).

Example 3: GSM8K (forward wins). A word problem states that Tom's ship travels at 10 miles per hour and that he sails from 1 to 4 PM, then travels back at 6 mph; the question asks how long the return trip takes. Forward CoT correctly computes the one-way distance (30 miles) and divides by 6 mph to obtain 5 hours. Elimination generates candidates including "1.67" (the return speed in hours per mile, misinterpreted as time), fails to rule it out, and answers incorrectly. Elimination depends on its candidate generation: when the true answer is not among well-formed candidates, the strategy fails.

These three patterns generalize across the dataset:

Table 6: Top 10 router features by absolute coefficient. Positive = predicts elimination success; negative = predicts forward success.

Rank	Feature	Coefficient	Interpretation
1	cm	−1.50	Unit abbreviation (chemistry/physics context) → favors forward
2	sequence	−1.39	Math sequence problems → favors forward
3	consider	+1.39	Reflective language ("consider which") → favors elimination
4	planet	−1.27	Astronomy context → favors forward
5	particles	+1.25	Particle-physics context → favors elimination
6	process	−1.24	Process-oriented wording → favors forward
7	equal	+1.22	Equality/constraint wording → favors elimination
8	following	−1.21	"Which of the following" (often math) → favors forward
9	minutes	−1.21	Time-calculation problems → favors forward
10	cost	+1.21	Cost/comparison problems → favors elimination

1. Constraint-checking problems (multiple-choice questions with explicit constraints) favor elimination because each candidate can be checked independently.
2. Common-error enumeration works when the teacher generates diverse candidates including typical errors, which elimination then explicitly rules out.
3. High-uncertainty problems with ambiguous intermediate steps benefit from explicit candidate comparison.

6.2 Router behavior and feature importance

The router uses a 226-dimensional feature space: 17 hand-crafted features, 200 TF-IDF unigrams and bigrams, 4 dataset one-hot indicators, and 5 arm one-hot indicators. Table 6 shows the top predictors by absolute logistic-regression coefficient.

The TF-IDF features dominate the hand-crafted ones, suggesting that lexical cues are highly informative for strategy selection. Notably, content words (particles, planet, cost) outperform structural features (is_mcq, n_choices) in this model. This indicates that domain-specific vocabulary is a reliable signal for which reasoning modality suits a problem. The leave-one-dataset-out (LODO) test shows that the router generalizes best to MATH-500 (55.0% accuracy when held out) and worst to ARC (47.2%), suggesting that lexical features transfer better to math problems with distinctive vocabulary than to general science multiple-choice questions.

6.3 Failure modes

1. Elimination on MATH-500. Despite training, elimination remains weak on MATH-500 (24–30% across arms). We hypothesize that generating plausible candidates for competition-level mathematics is harder than for grade-school mathematics, and that the teacher (Qwen3-32B) may not produce sufficiently diverse error patterns.
2. Joint arm degradation. The joint arm underperforms the specialized arms on some datasets: joint_tok reaches 32.4% forward on MATH-500 against 28.0% for forward-only, but its elimination accuracy drops to 24.2%. This suggests capacity competition between the two strategies within the same LoRA adapter.
3. Router ceiling. The router captures about 32% of the per-problem oracle gap, $(64.8 - 62.5)/(69.6 - 62.5) = 2.3/7.1$. Better features, for example embedding-based representations, or a larger training set could close this further.

6.4 Cost–accuracy Pareto frontier

Figure 2 plots accuracy against mean output tokens per problem for the forward and elimination strategies on the base model. The Pareto frontier shows that forward-only is the cheapest (about 434 tokens on average) but not always the most accurate; elimination-only is more expensive (about 610 tokens) and sometimes better, sometimes worse; the router

Table 7: Router feature ablation (5-fold stratified cross-validation). HC = hand-crafted.

Feature set	CV accuracy	AUC	Δ over HC
Hand-crafted only	58.3±2.3%	0.605	—
+ TF-IDF	62.3±4.3%	0.652	+4.0pp
+ TF-IDF + dataset	61.1±2.7%	0.650	+2.8pp
All (+ arm)	60.9±3.4%	0.650	+2.6pp

sits between them, achieving better accuracy than either alone at a moderate cost increase (about 490 tokens, a 13% increase over forward); and the oracle, which always picks the better strategy, defines the upper bound.

Figure 3 visualizes complementarity rates across datasets, showing that GPQA and MATH-500 have the highest rates of problems solved by exactly one strategy. Figure 4 shows per-dataset accuracy of the router versus single-strategy baselines and the oracle, and Figure 5 displays the top 10 router features by absolute coefficient.

6.5 Statistical significance

We run McNemar's exact test on paired predictions (forward versus elimination) for every arm–dataset combination. On GSM8K, forward is significantly better than elimination across almost all arms (base: $p < 0.001$; forward arm: $p < 0.001$; elimination arm: $p < 0.001$; joint: $p = 0.28$, n.s.; joint_tok: $p = 0.024$). On MATH-500, ARC, and GPQA, no significant difference emerges for most arms (all $p > 0.05$), with the exception of joint and joint_tok on MATH-500 ($p < 0.01$), where forward wins. These results confirm that the forward–elimination gap is domain-dependent: large and significant on grade-school mathematics, negligible on harder domains.

6.6 Router feature ablation

To quantify the contribution of TF-IDF features, we ablate the router feature set (Table 7). Hand-crafted features alone achieve 58.3±2.3% accuracy (AUC 0.605). Adding TF-IDF unigrams and bigrams improves accuracy to 62.3±4.3% (AUC 0.652), a +4.0 point gain. Adding dataset and arm one-hot indicators yields 61.1±2.7% and 60.9±3.4% respectively, showing that lexical cues are the primary driver of routing performance.

6.7 Strategy token ablation

The joint_tok arm prepends a strategy token ([forward] or [eliminate]) to each trace, while the joint arm relies on the prompt alone to select the strategy. Comparing the two arms across all datasets and strategies reveals no statistically significant difference (McNemar $p > 0.05$ for all eight comparisons). The largest difference is −1.8 points on MATH-500 elimination (joint 26.0% vs. joint_tok 24.2%, $p = 0.24$). This suggests that, at 8B scale with LoRA rank 32, the model does not effectively use the explicit strategy token, and prompt-based strategy selection is sufficient.

6.8 Router generalizes across trained arms

The router is trained on base-model predictions but is model-agnostic by design. When applied to trained arms (forward, elimination, joint, joint_tok), the router improves accuracy on every arm–dataset combination. Gains are largest on GPQA: +3.3 points (base), +4.2 (forward), +6.0 (elimination), +6.0 (joint), and +6.0 (joint_tok). On GSM8K, where forward dominates, the router preserves accuracy (at most +0.6 points) by correctly routing most problems to forward. This demonstrates that the router's lexical features capture problem properties that generalize across model checkpoints.

7 Conclusion

We systematically studied training elimination reasoning as a complementary strategy to forward chain-of-thought in LLMs. Through LoRA SFT on teacher-generated traces, we showed that an 8B model can learn to execute both strategies competently. The two strategies exhibit substantial complementarity — 5–29% of problems per dataset are solved by exactly one strategy — creating an opportunity for per-query routing. Our lightweight logistic-regression router, trained on 1,428 informative samples with TF-IDF text features, yields a +2.4% gain over the best single strategy (+3.3% on GPQA), at a modest 13% increase in token cost.

Future work includes: (i) scaling to larger models (14B, 32B) to test whether complementarity increases with capacity; (ii) embedding-based router features to close the remaining oracle gap; (iii) a learned candidate generator trained to produce diverse, plausible candidates conditioned on the problem, which could address the MATH-500 bottleneck; (iv) extending elimination to code generation and theorem proving, where candidate enumeration is natural; (v) online router adaptation that learns from deployment feedback; and (vi) extending the router to select among more than two strategies, for example by adding backward reasoning or tool use, as a natural generalization of the binary framework.

Limitations

1. Model scale and family. All experiments use a single model family (Qwen3) at 8B scale. It is unknown whether complementarity increases or decreases with model capacity (14B, 32B, 70B). We hypothesize that larger models may generate more diverse candidates, potentially increasing elimination effectiveness, but this remains to be tested.

2. Single training seed. Each SFT arm was trained with a single random seed. Variance across seeds is an important direction for future work.

3. Candidate generation bottleneck. For open-ended problems (GSM8K, MATH-500), elimination quality depends on the teacher's ability to generate plausible candidates. On MATH-500, where competition-level errors are subtle and problem-specific, candidate generation is the primary bottleneck. A learned candidate generator could address this.

4. Router feature ceiling. The router uses hand-crafted and TF-IDF features, which may cap performance. Embedding-based features or fine-tuned classifiers could close the remaining oracle gap.

AI use statement

Large language models were used throughout this project, and we state this openly rather than in vague terms. Concretely, LLM-based assistants were used for (i) implementing and refactoring the experimental harness and the analysis scripts, (ii) assisting with literature search and the first-pass organisation of the related-work sections, (iii) language polishing and condensation of the manuscript, and (iv) suggesting candidate prompt phrasings and code templates that the authors then reviewed, modified and validated. The elimination-trace filtering logic and the router feature pipeline were re-executed by the authors after every assistant-authored refactor, and each table in this paper was regenerated from the stored evaluation logs before submission. The research question, the design of the method, the evaluation protocol and the interpretation of the results are the authors' own. Every experimental result reported here was produced by actually executing the code on the models described in the paper; no number, figure or reference was generated by a language model and accepted without author verification against the raw run logs or the original publication. The authors take full responsibility for the content of the paper and for any remaining errors.

Ethics statement

This study does not involve human subjects, human annotations or personal data. All benchmarks used are publicly released academic datasets, and all models are open-weight checkpoints released under permissive licences. The method trains a model to execute a second reasoning strategy and to route between strategies per query; we regard this as a low-risk objective, but we note three caveats. First, the router is a learned selector over prompt features: its decisions are statistically informed, not guaranteed, so a deployment that silently discards the non-selected strategy loses the fallback that the oracle analysis quantifies. Second, elimination quality depends on candidate generation, and for open-ended questions a model that enumerates candidates may present an incorrect but confident shortlist; the analysis section documents this failure mode concretely. Third, our results are obtained with a single model family at 8B scale and should not be extrapolated to frontier-scale systems or to safety-critical domains without independent validation. The authors declare no competing interests. All compute used is reported in the method section.

Reproducibility statement

We describe the trace-generation procedure, the quality filter, the four SFT arms, the label-masking scheme, and the full training configuration (LoRA rank and scaling, learning rate, schedule, precision, gradient accumulation, batch size, number of GPUs) in the method section, together with the decoding settings used for every reported evaluation. The benchmarks and their evaluation sets are named explicitly, and all evaluation uses held-out test splits. Router training is described together with its feature space, its label definition (exactly-one-success problems), and the 5-fold stratified cross-validation protocol used for every reported router number; the leave-one-dataset-out results are produced by the same code path with the corresponding dataset held out. Statistical comparisons use McNemar's exact test on paired per-problem predictions, and the token-cost comparison uses a Mann–Whitney U test, both stated with the corresponding tables. Code, router features, teacher traces and evaluation logs will be released publicly upon acceptance; they are withheld at submission time only because the repository metadata would identify the authors under double-blind review.

References

Nishant Balepur, Shramay Palta, and Rachel Rudinger. It's not easy being wrong: Large language models struggle with process of elimination reasoning. In Findings of the Association for Computational Linguistics: ACL 2024, 2024. doi: 10.18653/v1/2024.findings-acl.604.

Justin Chih-Yao Chen, Zifeng Wang, Hamid Palangi, Rujun Han, Sayna Ebrahimi, Long Le, Vincent Perot, Swaroop Mishra, Mohit Bansal, Chen-Yu Lee, and Tomas Pfister. Reverse thinking makes LLMs stronger reasoners. In Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics (NAACL), 2025. arXiv:2411.19865.

Lingjiao Chen, Matei Zaharia, and James Zou. FrugalGPT: How to use large language models while reducing cost and improving performance, 2023. arXiv:2305.05176.

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, 2018. arXiv:1803.05457.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems, 2021. arXiv:2110.14168.

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. In Advances in Neural Information Processing Systems (NeurIPS): Datasets and Benchmarks Track, 2021. arXiv:2103.03874.

Namgyu Ho, Laura Schmid, and Se-Young Yun. Large language models are reasoning teachers. In Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (ACL), 2023. arXiv:2212.10071.

Cheng-Yu Hsieh, Chun-Liang Li, Chih-Kuan Yeh, Hootan Nakhost, Yasuhisa Fujii, Alexander Ratner, Ranjay Krishna, Chen-Yu Lee, and Tomas Pfister. Distilling step-by-step! outperforming larger language models with less training data and smaller model sizes. In Findings of the Association for Computational Linguistics: ACL 2023, 2023. arXiv:2305.02301.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In International Conference on Learning Representations (ICLR), 2022. arXiv:2106.09685.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In International Conference on Learning Representations (ICLR), 2024. arXiv:2305.20050.

Chenkai Ma and Xinya Du. POE: Process of elimination for multiple choice reasoning. In Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP), 2023. doi: 10.18653/v1/2023.emnlp-main.273.

Isaac Ong, Amjad Almahairi, Vincent Wu, Wei-Lin Chiang, Tianhao Wu, Joseph E. Gonzalez, M. Waleed Kadous, and Ion Stoica. RouteLLM: Learning to route LLMs with preference data. In International Conference on Learning Representations (ICLR), 2025. arXiv:2406.18665.

Zhihong Pan, Kai Zhang, Yuze Zhao, and Yupeng Han. Route to reason: Adaptive routing for LLM and reasoning strategy selection. arXiv preprint arXiv:2505.19435, 2025.

Qwen Team. Qwen3 technical report, 2025. arXiv:2505.09388.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. GPQA: A graduate-level Google-Proof Q&A benchmark, 2023. arXiv:2311.12022.

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

Dimitris Stripelis, Zhaozhuo Xu, Zijian Hu, Alay Dilipbhai Shah, Han Jin, Yuhang Yao, Jipeng Zhang, Tong Zhang, Salman Avestimehr, and Chaoyang He. TensorOpera Router: A multi-model router for efficient LLM inference. In Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing: Industry Track (EMNLP), 2024. doi: 10.18653/v1/2024.emnlp-industry.34.

Hongda Sun, Weikai Xu, Wei Liu, Jian Luan, Bin Wang, Shuo Shang, Ji-Rong Wen, and Rui Yan. DetermLR: Augmenting LLM-based logical reasoning from indeterminacy to determinacy. In Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (ACL), 2024. doi: 10.18653/v1/2024.acl-long.531.

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

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. In Advances in Neural Information Processing Systems (NeurIPS), 2022. arXiv:2201.11903.

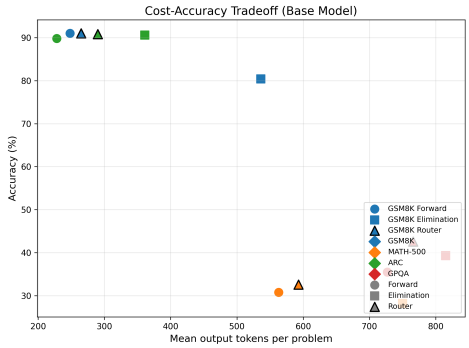


Figure 2: Cost–accuracy Pareto frontier across strategies on the base model.

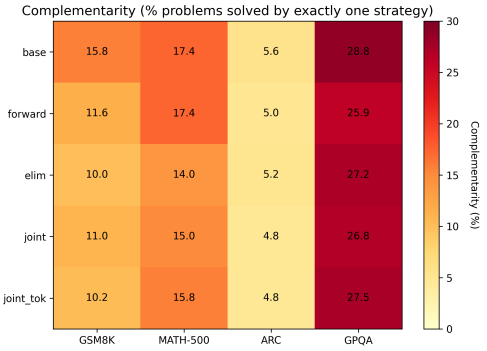


Figure 3: Complementarity rates across datasets.

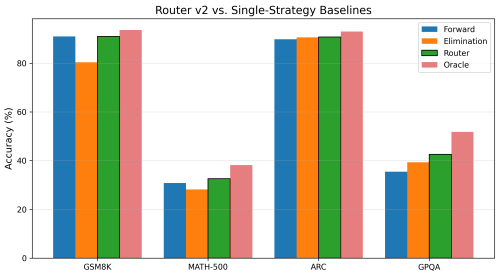


Figure 4: Per-dataset accuracy of the router versus single-strategy baselines and the oracle.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. In Advances in Neural Information Processing Systems (NeurIPS), 2023. arXiv:2305.10601.

Yanwei Yue, Guibin Zhang, Boyang Liu, Guancheng Wan, Kun Wang, Dawei Cheng, and Yiyan Qi. MasRouter: Learning to route LLMs for multi-agent systems. In Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (ACL), 2025. doi: 10.18653/v1/2025.acl-long.757.

A Additional Figures

Figures 3, 4 and 5 provide supporting views of the complementarity and router analyses summarized in Tables 2–6.

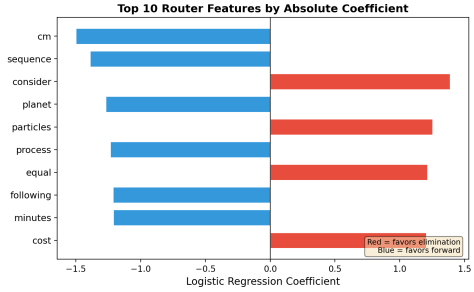


Figure 5: Top 10 router features by absolute coefficient.