STOP: Structured On-Policy Pruning of Long-Form Reasoning in Low-Data Regimes

Anonymous ACL submission

Abstract

Long chain-of-thought (Long CoT) reasoning improves performance on multi-step problems, but it also induces *overthinking*: models often generate low-yield reasoning that increases inference cost and latency. This inefficiency is especially problematic in *low-data* fine-tuning regimes, where real applications adapt reasoning models with limited supervision and cannot rely on large-scale teacher distillation or heavy test-time control. To address this, we propose STOP (Structured On-policy Pruning), an on-policy algorithm for analyzing and pruning long-form reasoning traces. STOP constructs self-distilled traces from the model. Then it maps each trace into a structured reasoning interface through node segmentation, taxonomy annotation, and reasoning-tree construction. On top of this interface, we introduce ECN (Earliest Correct Node), which retains the shortest prefix ending at the earliest node that both functions as an answering conclusion and yields the correct final answer, removing redundant post-solution reasoning while preserving semantic continuity. Experiments on DeepSeek-R1-Distill-Qwen-7B and DeepSeek-R1-Distill-LLaMA-3-8B across GSM8K, Math 500, and AIME 2024 show that STOP reduces generated tokens by 19.4–42.4% while largely preserving accuracy in low-data fine-tuning. Beyond efficiency, our analyses show that STOP induces much smaller distributional shift than teacher-guided pruning, improves the structural efficiency of generated reasoning, and reallocates reasoning effort away from redundant verification and backtracking toward more productive exploration.[1]

1 Introduction

Reasoning language models (RLMs) have recently demonstrated strong performance on complex multi-step tasks by generating long chain-of-thought (Long CoT) traces (Wei et al., 2022; Yao et al., 2024; Jaech et al., 2024; Guo et al., 2025). However, longer reasoning does not simply add useful intermediate steps—models often keep spending computation after the answer is already available, a phenomenon known as *overthinking* (Chen et al., 2025b). Such post-solution reasoning inflates token usage, increases latency, and raises deployment cost. The issue is especially acute for distilled reasoning models, where efficiency is often a primary motivation for deployment, and in low-data fine-tuning settings, where real applications have limited supervision and cannot afford large-scale distillation or expensive test-time control.

A growing body of work seeks to make long-form reasoning more efficient (Yang et al., 2025; Yi and Wang, 2025), but existing approaches involve important trade-offs. Compression-based methods(Ma et al., 2025) rewrite or compress CoT into shorter representations, which can reduce length but may blur or remove critical intermediate logic. Pruning-based methods(Zhao et al., 2025) remove selected reasoning segments, but aggressive deletion can break logical continuity or suppress beneficial self-correction. Token-budget methods(Han et al., 2025) explicitly constrain reasoning length, but require manually chosen budgets and may under-allocate computation on genuinely difficult problems. Moreover, many efficient-reasoning methods(Lippmann and Yang, 2025) rely on externally rewritten or teacher-generated traces. In low-data regimes, such supervision can be poorly matched to the student's native reasoning style, leading to substantial distributional shift during fine-tuning(Chen et al., 2025a).

In this work, we propose STOP (Structured On-policy Pruning), an on-policy framework for structured analysis and pruning of long reasoning traces.

[1] Our code is available at: https://anonymous.4open.science/r/ECN-STOP-7FC0/.

STOP has three tightly connected layers. First, it constructs supervision from *self-distilled* successful traces generated by the same student model that will later be fine-tuned. This keeps training trajectories closer to the student's native reasoning distribution. Second, it maps each trace into a *structured reasoning interface* through node segmentation, taxonomy annotation, and reasoning-tree construction. The resulting representation decomposes a trace into coherent reasoning units, labels their functional roles (e.g., clarification, exploration, verification, backtracking, conclusion), and provides a structural view of how the reasoning unfolds. Third, built on top of this interface, we introduce ECN (Earliest Correct Conclusion Node), a node-level pruning strategy that keeps only the minimal prefix up to the earliest node that already yields the correct answering conclusion.

Empirically, STOP yields significant efficiency gains across GSM8K, Math 500, and AIME 2024. Across DeepSeek-R1-Distill-Qwen-7B and DeepSeek-R1-Distill-LLaMA-3-8B, STOP reduces generated tokens by 19.4–42.4% while maintaining competitive accuracy. The results also reveal three broader patterns. First, self-distilled supervision is significantly more stable than teacher-guided supervision in the low-data regime, suggesting that on-policy trace construction is critical for preserving reasoning behavior. Second, node-level analysis shows that ECN does not merely truncate outputs: it systematically reduces redundant verification and backtracking while reallocating more of the reasoning budget toward exploration. Third, structural analysis of induced reasoning trees shows that STOP produces more focused and efficient reasoning morphologies, reducing redundant branching while largely preserving the core solution path. These findings suggest that effective pruning should be understood not only as length reduction, but as reorganization of the structure of generated reasoning.

In summary, our contributions are:

- We introduce STOP, a unified on-policy framework for long-form reasoning that combines self-distilled trace construction, a structured reasoning interface based on node segmentation / taxonomy annotation / tree construction, and node-level pruning.
- We propose ECN (Earliest Correct Node), a pruning strategy defined on top of the structured node interface, which retains the minimal prefix up to the earliest correct answering conclusion while preserving semantic continuity. ECN consistently outperforms existing baselines.
- We show that the same structured interface can be used to analyze reasoning behavior before and after pruning, revealing that our method does not merely shorten traces but shifts reasoning away from redundant verification and backtracking toward more productive exploration and more structurally efficient reasoning.
- We demonstrate that, in a low-data regime, STOP achieves large token reductions with competitive accuracy, and that self-distilled supervision is substantially more stable than teacher-guided supervision for fine-tuning distilled reasoning models.

2 Method

We introduce the details of STOP (Structured On-policy Pruning). As shown in Figure 1. STOP has three tightly connected components: (i) self-distilled trace construction; (ii) Structured reasoning interface; and (iii) ECN (Earliest Correct Node) pruning.

2.1 On-Policy Self-Distilled Trace Construction

The first component of STOP is **self-distilled trace construction**. Here, *self-distilled* means that the supervision traces are generated by the same student model that is later fine-tuned. This keeps the supervision distribution close to the student's native reasoning behavior and reduces the mismatch introduced by teacher-guided traces(Chen et al., 2025a).

Models and generation. We use two distilled reasoning models as students: DeepSeek-R1-Distill-Qwen-7B and DeepSeek-R1-Distill-LLaMA-3-8B. For each question in a shared set of 1,000 open-ended reasoning problems, we generate outputs with temperature 0.7 and sample multiple random seeds to obtain diverse reasoning traces. For teacher-guided baselines, we use DeepSeek-R1-0528 only as the trace generator.

Best-of-K filtering. We apply a Best-of-K ($K = 4$) filtering strategy to construct successful on-policy traces. We keep only candidates whose final answers match the ground truth under task-specific normalization. Among correct traces, we select *intermediate-length* samples rather than always taking the shortest or longest outputs. This avoids trivial short traces and excessively verbose

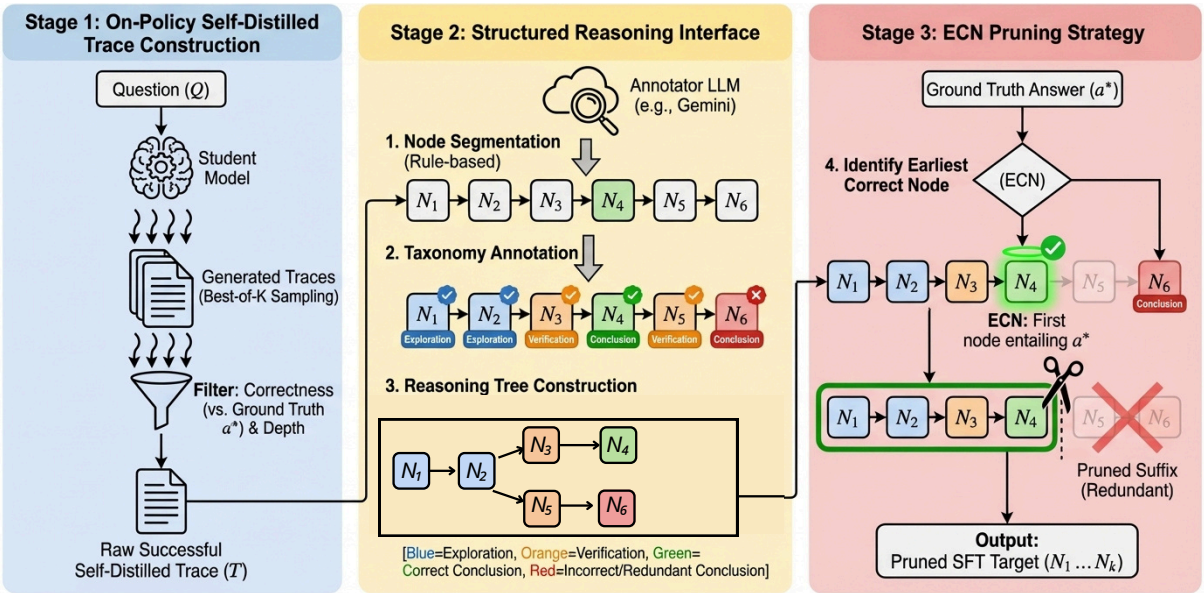

Figure 1: Overview of STOP. The framework first constructs successful self-distilled traces, then maps each trace into a structured reasoning interface through node segmentation, taxonomy annotation, and reasoning-tree construction. On top of this interface, it applies ECN pruning and supports analysis of reasoning behavior changes.

outliers, yielding a more balanced supervision set.

Why self-distilled. The goal of self-distilled trace construction is to keep fine-tuning on-policy. In low-data LoRA fine-tuning, teacher-generated traces may be stylistically and behaviorally out-of-distribution for the student model. By contrast, self-distilled traces are drawn from the student's own reasoning distribution, which helps reduce distributional shift and preserve its native reasoning behavior during adaptation.

2.2 Structured Reasoning Interface

The second component of STOP is a structured interface over each reasoning trace. This interface is built in three stages: node segmentation, taxonomy annotation, and reasoning-tree construction. Together, these steps produce a shared representation that supports both ECN pruning and before or after analysis of reasoning traces.

Node segmentation. We segment each reasoning trace into a node sequence using a deterministic, rule-based procedure. The goal is to create units: (i) large enough to preserve local semantic coherence, but (ii) small enough to support annotation, pruning, and analysis.

Segmentation rules. We detect discourse markers such as *However, But, Alternatively, So, Now* (case-insensitive) and punctuation-based boundaries such as sentence endings and line breaks. When a marker appears at the start of a sentence, we split the trace so that the full marker-starting sentence becomes a new node, while the preceding text remains in the previous node. Because pruning is defined only at node boundaries, this segmentation helps preserve semantic continuity and avoids producing fractured supervision targets.

Taxonomy annotation. Each node is annotated with a functional role that captures how it contributes to the reasoning process. We adopt the following 5-class taxonomy:

- **Backtracking:** revising earlier steps or assumptions to correct errors or resolve conflicts.
- **Verification:** checking or validating a claim or result without changing the main plan.
- **Exploration:** proposing or trying alternative hypotheses or solution routes.
- **Clarification:** restating assumptions, reducing ambiguity, or organizing problem.
- **Conclusion:** asserting an intermediate or final result derived from prior reasoning.

All taxonomy labels are produced in a single pass by `Gemini-2.5-Flash` (accessed June–August 2025) with few-shot prompts. To reduce variance, we use deterministic decoding. The prompt includes the original question, the full trace, the candidate node text, and a short label schema with examples. The model outputs a label and a brief justification, which is stored for audit but not used during training.

Reasoning tree construction. On top of the

node sequence, we construct an n-ary reasoning tree that provides a global structural view of the trace. Each tree node corresponds to one segmented reasoning unit, and a parent–child edge indicates that one node supports another as evidence, premise, sub-result, or elaboration. Although ECN is defined on the ordered node sequence, the tree is useful for analyzing structural redundancy and comparing reasoning behavior before and after pruning.

Tree construction is performed automatically using `Gemini-2.5-Flash` with a single-pass structured prompt. Given the ordered node sequence, the model outputs an n-ary JSON reasoning tree while preserving node order.

2.3 Earliest Correct Node Pruning Strategy

Earliest Correct Node(ECN) identification. Built on top of the structured node sequence, ECN requires identifying the earliest node that is both (i) an answering conclusion and (ii) correct. Among nodes labeled Conclusion, we further distinguish between intermediate conclusions, which support later reasoning but do not directly answer the question q, and answering conclusions, which are intended to answer q, even if the model later revises them. Given the ground-truth answer $a^\star$, we judge whether an answering-conclusion node entails the correct final answer. When the node contains an explicit answer, we extract the value and compare it with $a^\star$ under task-specific normalization. When the node is implicit or ambiguous, we prompt `Gemini-2.5-Flash` with $(q, a^\star, N_i)$ to determine whether N_i constitutes a correct answering conclusion. We use deterministic decoding for this judgment and further validate the results with human annotation in the appendix.

ECN definition. Let $\mathrm{CAC}(N_i)$ indicate that node N_i is judged to be a correct answering conclusion. We define

$$\mathcal{C} = \{\, i \in \{1, \ldots, m\} : \mathrm{CAC}(N_i) \,\}.$$

If $\mathcal{C} \neq \varnothing$, the ECN index is defined as

$$k = \min \mathcal{C}.$$

That is, ECN selects the earliest node in the trace that already yields the correct answer. If $\mathcal{C} = \varnothing$, the trace is excluded from ECN supervision.

ECN pruning. Given the node sequence $S = (N_1, \ldots, N_m)$ and the ECN index k, ECN pruning keeps the minimal prefix $\mathrm{ECN}(S) = (N_1, \ldots, N_k)$. Because pruning operates on coherent reasoning nodes rather than arbitrary token spans, the resulting target remains semantically continuous while removing redundant post-solution reasoning.

3 Experiment Setup

Models. We use two distilled reasoning models as students: DeepSeek-R1-Distill-Qwen-7B and DeepSeek-R1-Distill-LLaMA-3-8B. For self-distilled variants, the same student model is used both to generate training traces and to initialize subsequent LoRA fine-tuning. Teacher-guided baselines use DeepSeek-R1-0528 to generate the trace. This design isolates the effect of supervision source and pruning strategy from changes in model family or initialization.

Training data. We randomly sample 1000 questions from PRM-12 and refer to this setting as *low-data* fine-tuning. For each question, we generate candidate reasoning traces, retain successful traces using Best-of-K filtering, and transform the retained traces with STOP: deterministic node segmentation, taxonomy annotation, reasoning-tree construction, and ECN pruning. The resulting traces are used as SFT targets.

Evaluation benchmarks. We evaluate both problem-solving performance and reasoning efficiency on three benchmarks with different difficulty levels and reasoning styles: GSM8K, Math 500, and AIME 2024. Following prior work(Chen et al., 2025b), we evaluate the performance based on following metrics. 1. Answer Accuracy. For open-ended questions, correctness is determined by matching the final numerical answer with the ground truth. 2. Token quantity. We report the average number of tokens used in the complete output.

Structured trace interface and annotation. Our training targets are produced through the shared structured reasoning interface introduced in Section 2. We first segment each trace *deterministically* into semantically coherent reasoning nodes using discourse and formatting rules. We then use Gemini-2.5-Flash to assign taxonomy labels, construct reasoning trees, and, when needed, judge ambiguous candidate conclusion nodes for ECN identification. Thus, the same interface is used both to construct ECN-pruned supervision targets and to analyze reasoning behavior before and after pruning. Appendix A provides all prompts,

decoding settings, structured output formats, and post-processing rules.

Implementation details We apply LoRA-based(Hu et al., 2022) supervised fine-tuning on a single NVIDIA A100 (80GB) GPU with a maximum training sequence length of 8192 tokens. Unless otherwise specified, all hyperparameters are fixed across models and datasets; Full details are given in Appendix A.

Baselines. We compare the following baselines:

- Base: the original student model without additional fine-tuning.
- No Thinking: fine-tuning on outputs with all content inside the `<think>` tag removed, retaining only the final answer format.
- Teacher-Guided-Full: fine-tuning on full reasoning traces generated by the teacher model.
- Teacher-Guided Random Prune: fine-tuning on teacher-guided traces truncated at a random node, controlling for length reduction without a principled stopping rule.
- Teacher-Guided-ECN: fine-tuning on teacher-generated traces after ECN pruning.
- Self-distilled-Full: fine-tuning on full reasoning traces generated by the student itself.
- Self-distilled Token Skip (Xia et al., 2025): a compression baseline that shortens self-distilled traces through token-skipping style CoT compression.
- Self-distilled Random Prune: fine-tuning on self-distilled traces truncated at a random node boundary, controlling for length reduction without a principled stopping rule.
- Self-distilled-ECN (ours): fine-tuning on self-distilled traces after ECN pruning.

4 Results

STOP achieves strong efficiency–accuracy trade-offs across models and task difficulty. Table 1 compares our main method, Self-distilled-ECN, against the base models and alternative supervision/pruning strategies on GSM8K, Math 500, and AIME 2024, spanning easy, medium, and hard reasoning settings. Across both Qwen and LLaMA, Self-distilled-ECN consistently reduces output length while maintaining competitive accuracy. Relative to the Base models, token savings range from 19.4% to 42.4%. The trade-off is weakest on GSM8K and Math 500, where STOP preserves most of the original accuracy, and becomes sharper on the hardest setting, AIME 2024.

On GSM8K, Self-distilled-ECN preserves performance while substantially reducing token usage. For Qwen, it reaches 91.1% accuracy versus 90.1% for Base while reducing tokens by 27.8%. For LLaMA, it attains 83.2% versus 84.0% for Base, while reducing tokens by 40.1%. On Math 500, Self-distilled-ECN delivers the largest efficiency gains: for Qwen, it achieves 86.8% accuracy versus 87.8% for Base with a 42.4% token reduction; for LLaMA, it reaches 87.7% versus 87.9% for Base with a 36.5% reduction. On AIME 2024, Self-distilled-ECN remains effective for Qwen, improving accuracy from 36.7% to 46.7% while reducing tokens by 28.9%. For LLaMA, it reduces tokens by 19.4%, though with a modest accuracy drop from 43.3% to 40.0%. Overall, these results show that STOP consistently mitigates overthinking, while the efficiency–accuracy trade-off becomes more pronounced on harder problems.

ECN is consistently stronger than alternative pruning and compression baselines. Across both model families, ECN is the most effective pruning rule among the tested length-reduction baselines. Compared with Random Pruning and Token Skip, ECN achieves dramatically higher accuracy while using fewer tokens on nearly all benchmarks. For example, on Math 500 with Qwen, Self-distilled-ECN attains 86.8% accuracy with 2118 tokens, compared with 34.2% and 6252 tokens for Token Skip, and 81.8% and 1992 tokens for Random Pruning. A similar pattern holds for LLaMA, where Self-distilled-ECN reaches 87.7% accuracy with 2168 tokens, far outperforming both Token Skip (49.6%, 5418 tokens) and Random Pruning (74.0%, 1628 tokens). These comparisons show that the benefit of ECN does not come from shortening traces arbitrarily; it comes from stopping at a coherent node-level conclusion.

ECN is also effective beyond the self-distilled setting. When applied to teacher-guided traces, Teacher-Guided-ECN consistently reduces token usage relative to Teacher-Guided-Full and often improves accuracy as well. For Qwen on GSM8K, Teacher-Guided-ECN improves accuracy from 53.5% to 79.2% while reducing tokens from 2548 to 1168. On Qwen Math 500, it improves accuracy from 27.8% to 54.0% while reducing tokens from 6436 to 4512. Similar gains appear on AIME 2024 for both Qwen and LLaMA. This shows that ECN is a robust pruning strategy across multiple supervision sources, even though its strongest performance is obtained in the full self-distilled STOP framework.

Model	Method	AIME'24				GSM8K				Math 500			
		Acc	Δ	Tok	Δ%	Acc	Δ	Tok	Δ%	Acc	Δ	Tok	Δ%
Qwen	Base	.367	–	10848	–	.901	–	1311	–	.878	–	3678	–
	NT	.233	-.134	2411	-77.8	.873	-.028	192	-85.4	.772	-.106	678	-81.6
	TG-Full	.067	-.300	14854	+36.9	.535	-.366	2548	+94.4	.278	-.600	6436	+75.0
	TG-RP	.033	-.334	15344	+41.5	.427	-.474	6153	+369.4	.186	-.692	7319	+99.0
	TG-ECN	.100	-.267	13574	+25.1	.792	-.109	1168	-10.9	.540	-.338	4512	+22.7
	SD-Full	**.533**	**+.166**	9784	-9.8	**.912**	**+.011**	1174	-10.5	**.876**	-.002	2989	-18.7
	SD-Token Skip	.133	-.234	13353	+23.1	.645	-.256	3645	+178.0	.342	-.536	6252	+70.0
	SD-RP	.233	-.034	8069	-25.6	.884	-.017	1003	-23.5	.818	-.060	**1992**	**-45.8**
	SD-ECN (ours)	.467	+.100	**7716**	**-28.9**	.911	+.010	**946**	**-27.8**	.868	-.010	2118	-42.4
Llama	Base	.433	–	11855	–	.840	–	1535	–	.879	–	3413	–
	NT	.033	-.400	3988	-66.4	.764	-.076	257	-83.3	.608	-.271	989	-71.0
	TG-Full	.067	-.366	15332	+29.3	.613	-.227	4138	+169.6	.436	-.443	5710	+67.3
	TG-RP	.033	-.400	16167	+36.4	.606	-.234	4617	+200.8	.310	-.569	6650	+94.9
	TG-ECN	.167	-.266	14353	+21.1	.667	-.173	3292	+114.5	.430	-.449	5487	+60.8
	SD-Full	**.600**	**+.167**	9948	-16.1	.822	-.018	1514	-1.4	.822	-.57	3322	-2.7
	SD-Token Skip	.033	-.400	15735	+32.7	.674	-.166	3248	+111.6	.496	-.383	5418	+58.8
	SD-RP	.233	-.200	**5170**	**-56.4**	.807	-.033	**784**	**-48.9**	.740	-.139	**1628**	**-52.3**
	SD-ECN (ours)	.400	-.033	9555	-19.4	**.832**	**-.008**	920	-40.1	**.877**	**-.002**	2168	-36.5

Table 1: Main results showing the trade-off between reasoning accuracy and token efficiency across models, benchmarks, and training/pruning strategies. Method abbreviations are: NT = No Thinking; TG = Teacher-Guided; SD = Self-distilled; RP = Random Prune. Our main method, **SD-ECN**, is the primary instantiation of STOP. Δ denotes the accuracy change relative to the corresponding Base model, and Δ% denotes the relative token change. Best self-distilled results are shown in **bold**.

Self-distilled supervision is substantially more effective than teacher-guided supervision in the low-data regime. A pattern in Table 1 is that self-distilled supervision outperforms teacher-guided supervision under both full-trace and ECN training. For Qwen on AIME 2024, Self-distilled-Full reaches 53.3% accuracy compared to 6.7% for Teacher-Guided-Full, and Self-distilled-ECN reaches 46.7% compared to 10.0% for Teacher-Guided-ECN. The same pattern holds across GSM8K and Math 500 for both student models: self-distilled variants are consistently stronger than their teacher-guided counterparts, often by large margins. This indicates that in the low-data setting, staying close to the student's native reasoning distribution is more important than imitating a stronger but mismatched teacher.

To examine this effect at the token-distribution level, Figure 2 measures the forward KL divergence between the base model and models fine-tuned with self-distilled ECN or teacher-guided ECN. The teacher-guided model undergoes a larger distributional shift, with the KL divergence increasing from around 0.1 at 50 training steps to above 0.25 by 300 steps. In contrast, the self-distilled model remains closer to the base model throughout training, staying roughly in the 0–0.1 range. This suggests that self-distilled ECN preserves the original reasoning behavior of the student model better than teacher-guided supervision, helping explain why self-distilled STOP is more stable and effective in low-data fine-tuning.

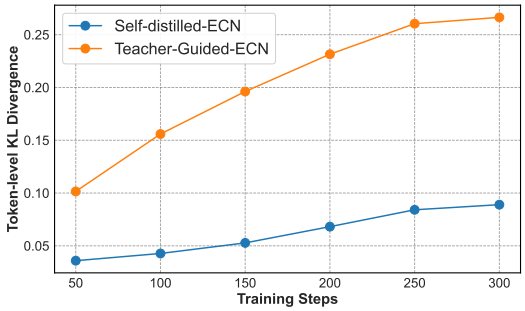

Figure 2: Token-level forward KL divergence between the reference model (DeepSeek-R1-Distill-Qwen-7B) and models fine-tuned with Self-distilled-ECN and Teacher-Guided-ECN.

To examine how self-distilled supervision behaves under different supervision budgets, Figure 3 plots performance at 200, 500, and the full dataset (about 900 examples). The main pattern is that scaling the budget does not produce a uniform monotonic gain across models and benchmarks. For Qwen, the 200-example setting is already highly competitive, matching or exceeding larger budgets on GSM8K and MATH-500, while AIME 2024 peaks at 500 examples. For Llama,

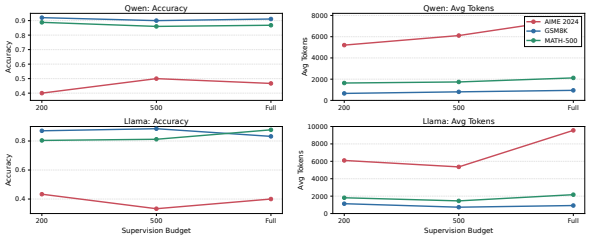

Figure 3: Accuracy gains and token savings of self-distilled models across three supervision budgets (200, 500, and Full) on AIME 2024, GSM8K, and MATH-500 for Qwen and Llama.

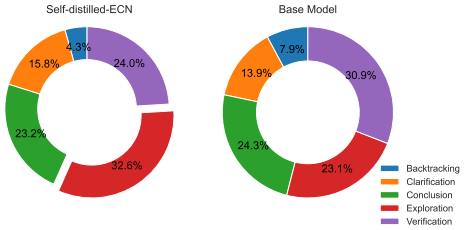

Figure 4: Reasoning node-type distribution on Math 500 Level 3 for Qwen Base and Self-distilled-ECN. Self-distilled-ECN shifts reasoning toward more *Exploration* and less *Verification*/*Backtracking*, while largely preserving *Conclusion*.

the 500-example setting gives the best balance on GSM8K, whereas the full budget is most beneficial on MATH-500. Token usage generally increases with larger supervision budgets, although some mid-budget settings remain more efficient than the full setting. Overall, these results suggest that self-distilled supervision is already effective in the low-data regime, and that increasing the budget beyond a few hundred examples does not consistently translate into better efficiency or accuracy.

5 Analysis

Rather than judging STOP solely by token reduction, we examine whether it changes the *structure* of reasoning(Besta et al., 2025). Using three views of self-distilled traces—taxonomic motifs(Gandhi et al., 2025), morphological patterns(Jiang et al., 2025), and pruning efficiency dynamics(Wei et al., 2025)—we find that STOP does more than compress outputs: it reorganizes reasoning into a more focused, stable, and structurally efficient form.

Taxonomic motifs: STOP reallocates reasoning from redundant checking to productive search. Figure 4 compares the node-type distribution of the Base Qwen model and the Self-distilled-ECN Qwen model on Math 500 Level 3. Self-distilled-ECN substantially increases *Exploration* from 23.1% to 32.6% (+9.5 points), suggesting that the model devotes more of its reasoning budget to core solution search rather than post-hoc deliberation. At the same time, *Verification* decreases from 30.9% to 24.0% (-6.9 points) and *Backtracking* drops from 7.9% to 4.3% (-3.6 points), consistent with ECN removing late-stage re-checking and unnecessary revisions after a correct answering conclusion has already appeared. Importantly, *Conclusion* remains nearly unchanged (24.3% to 23.2%), and *Clarification* slightly increases (13.9% to 15.8%), indicating that STOP preserves the core logical scaffolding while changing the semantic flow of reasoning. Thus, the gain is not mere compression; it is a redistribution of reasoning effort toward higher-yield cognitive operations.

Metric	ECN	Base	Δ
Total tree nodes	11.11	15.20	-4.10
Maximum depth	5.62	6.28	-0.66
Maximum width	3.47	5.23	-1.76
Width-to-depth ratio	0.93	1.15	-0.22
Mean leaf depth	4.29	4.60	-0.31
Leaf depth variance	4.23	2.59	1.65
Mean branching factor	1.69	1.90	-0.21
Maximum branching factor	3.23	4.61	-1.38
Branching variance	1.81	3.43	-1.62
Structural efficiency	0.65	0.53	0.13
Sackin index	16.73	30.91	-14.17
Generalized Colless index	5.11	11.29	-6.17

Table 2: Morphology metrics on Math 500 Level 3. See details in Appendix D.

Morphological patterns: STOP transforms "bushy" exploration into streamlined, goal-directed reasoning trees. Table 2 shows that STOP shifts the Base model from broad, hesitant branching to a more linear and efficient search strategy. Trees become more compact (15.20 → 11.11 nodes) mainly by reducing lateral expansion: maximum width drops sharply (5.23 → 3.47) while depth remains similar (6.28 → 5.62). As a result, the width-to-depth ratio falls from 1.15 to 0.93, indicating a move from parallel hypothesis testing toward more confident deduction, reinforced by lower mean and maximum branching factors. The trees are also more balanced and efficient, with structural efficiency rising from 0.53 to 0.65 and Sackin and Generalized Colless indices falling substantially (30.91 → 16.73; 11.29 → 5.11). Overall, STOP not only shortens reasoning traces but also concentrates computation on the essential inferential path.

Pruning efficiency dynamics: STOP shortens traces through stable structural adaptation rather than brittle truncation. Appendix Table 3 shows that the Earliest Final-Answer depth changes little from Base to Self-distilled-ECN (3.029 →

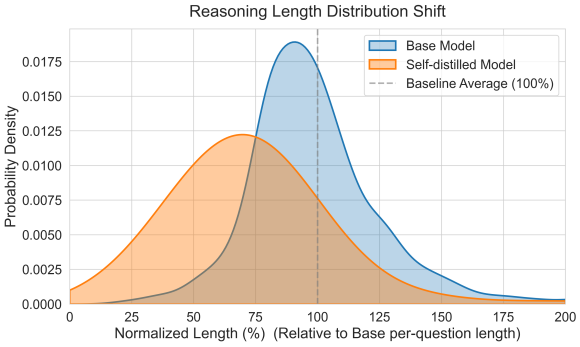

Figure 5: Response-length distribution of the base Qwen model and the self-distilled ECN-Qwen model on Math 500 (Level 3).

3.181), indicating that STOP does not reduce cost mainly by making answers appear much earlier. Instead, it cuts redundant reasoning after the answer first emerges: the post-answer source-node gap drops sharply from 12.143 to 5.762. This pattern aligns with Figure 5, where normalized reasoning-length distributions shift left under Self-distilled-ECN, showing a broad move toward shorter outputs rather than a few extreme truncations. Together, these results suggest that STOP changes the model's default reasoning policy by preserving similar answer-emergence depth while substantially reducing unnecessary post-answer expansion.

6 Related Work

Overthinking and Reasoning-Length Control Recent reasoning language models (RLMs) achieve strong performance on multi-step tasks by generating long chain-of-thought (CoT) traces (Jaech et al., 2024; Guo et al., 2025). However, extended reasoning often introduces substantial computational overhead and frequently contains repetitive or unnecessary steps, a phenomenon widely referred to as *overthinking* (Sui et al., 2025). Prior studies analyze this behavior and its failure modes across modern long-reasoning systems (Team et al., 2025; Chen et al., 2025b). A natural direction is to *compress* verbose natural-language CoT into shorter representations while preserving problem-solving ability. For example, Shen et al. (2025) propose continuous chain-of-thought via self-distillation, aiming to move from explicit natural-language reasoning to a more compact internal representation. Another line of work reduces reasoning cost by *pruning* particular parts of the reasoning trace or discouraging specific behaviors. Wang et al. (2025) observe that o1-style models may jump between reasoning paths instead of fully developing promising ones, suggesting that a large fraction of tokens can be spent on low-yield branches. Building on this perspective, Luo et al. (2025) propose O1-Pruner to reduce reasoning cost.

Teacher-Guided vs. Self-Distilled Supervision. A common strategy for reasoning adaptation is teacher-guided distillation, where a stronger teacher model provides high-quality reasoning traces as supervision for a student model (Li et al., 2025a). While effective, this setup often introduces distribution mismatch, since teacher and student models may differ substantially in capability, style, verbosity, and intermediate reasoning preferences, which can lead to unstable optimization or behavioral drift under direct trace imitation (Li et al., 2025b). In contrast, self-distillation and on-policy supervision use trajectories that are more consistent with the student's own distribution, thereby mitigating such mismatch and providing a more stable supervision signal (Chen et al., 2025a; Agarwal et al., 2024). This distinction is especially relevant in the low-data fine-tuning regime, where limited supervision leaves less room to accommodate teacher-student mismatch. Our method follows this intuition by constructing pruning targets from the student's own reasoning traces rather than externally provided teacher trajectories.

7 Conclusion

We introduced STOP, a unified on-policy framework for improving the efficiency of long-form reasoning in low-data regimes. STOP combines self-distilled trace construction, a structured reasoning interface based on node segmentation, taxonomy annotation, and tree construction, and ECN pruning, a node-level stopping rule that removes redundant post-solution reasoning while preserving semantic continuity. Across two distilled reasoning models and three reasoning benchmarks, STOP reduces generated tokens by 19.4–42.4% while maintaining competitive accuracy. Beyond efficiency, our analyses show that STOP systematically reorganizes the model's reasoning behavior: it produces more focused reasoning structures, reallocates effort away from redundant verification and backtracking, and preserves greater on-policy stability than teacher-guided supervision. These results suggest that efficient reasoning is not just a matter of cutting tokens, but of learning when and how to stop thinking productively.

Limitations

ECN is an oracle training-time stopping rule. ECN is defined using answer correctness: identifying the earliest *correct* answering conclusion requires access to the ground-truth answer or an external correctness judge. This makes ECN a strong supervision-construction method, but not a directly deployable inference-time stopping rule. A practical deployment version would require confidence estimation, learned stopping predictors, or verifier-based approximations.

The structured interface depends partly on external annotation. While node segmentation is deterministic, taxonomy annotation, tree construction, and some ambiguous conclusion judgments rely on `Gemini-2.5-Flash`. This introduces dependence on an external model with its own biases, errors, and cost. Mistakes in labeling or conclusion judgment can propagate into pruning decisions and affect the quality of the resulting supervision targets.

References

Rishabh Agarwal, Nino Vieillard, Yongchao Zhou, Piotr Stanczyk, Sabela Ramos Garea, Matthieu Geist, and Olivier Bachem. 2024. On-policy distillation of language models: Learning from self-generated mistakes. In *International Conference on Learning Representations*, volume 2024, pages 21246–21263.

Maciej Besta, Florim Memedi, Zhenyu Zhang, Robert Gerstenberger, Guangyuan Piao, Nils Blach, Piotr Nyczyk, Marcin Copik, Grzegorz Kwaśniewski, Jürgen Müller, and 1 others. 2025. Demystifying chains, trees, and graphs of thoughts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 47(12):10967–10989.

Howard Chen, Noam Razin, Karthik Narasimhan, and Danqi Chen. 2025a. Retaining by doing: The role of on-policy data in mitigating forgetting. *arXiv preprint*, arXiv:2510.18874.

Xingyu Chen, Jiahao Xu, Tian Liang, Zhiwei He, Jianhui Pang, Dian Yu, Linfeng Song, Qiuzhi Liu, Mengfei Zhou, Zhuosheng Zhang, Rui Wang, Zhaopeng Tu, Haitao Mi, and Dong Yu. 2025b. Do NOT think that much for 2+3=? on the overthinking of long reasoning models. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pages 9487–9499. PMLR.

Kanishk Gandhi, Ayush Chakravarthy, Anikait Singh, Nathan Lile, and Noah D Goodman. 2025. Cognitive behaviors that enable self-improving reasoners, or, four habits of highly effective stars. *arXiv preprint arXiv:2503.01307*.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, and 1 others. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

Tingxu Han, Zhenting Wang, Chunrong Fang, Shiyu Zhao, Shiqing Ma, and Zhenyu Chen. 2025. Token-budget-aware llm reasoning. In *Findings of the Association for Computational Linguistics: ACL 2025*.

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

Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, and 1 others. 2024. Openai o1 system card. *arXiv preprint arXiv:2412.16720*.

Gangwei Jiang, Yahui Liu, Zhaoyi Li, Wei Bi, Fuzheng Zhang, Linqi Song, Ying Wei, and Defu Lian. 2025. What makes a good reasoning chain? uncovering structural patterns in long chain-of-thought reasoning. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 6490–6514.

Yang Li, Youssef Emad, Karthik Padthe, Jack Lanchantin, Weizhe Yuan, Thao Nguyen, Jason Weston, Shang-Wen Li, Dong Wang, Ilia Kulikov, and 1 others. 2025a. Naturalthoughts: Selecting and distilling reasoning traces for general reasoning tasks. *arXiv preprint arXiv:2507.01921*.

Yuetai Li, Xiang Yue, Zhangchen Xu, Fengqing Jiang, Luyao Niu, Bill Yuchen Lin, Bhaskar Ramasubramanian, and Radha Poovendran. 2025b. Small models struggle to learn from strong reasoners. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 25366–25394.

Philip Lippmann and Jie Yang. 2025. Style over substance: Distilled language models reason via stylistic replication. *arXiv preprint arXiv:2504.01738*.

Haotian Luo, Li Shen, Haiying He, Yibo Wang, Shiwei Liu, Wei Li, Naiqiang Tan, Xiaochun Cao, and Dacheng Tao. 2025. O1-pruner: Length-harmonizing fine-tuning for o1-like reasoning pruning. *arXiv preprint arXiv:2501.12570*.

Xinyin Ma, Guangnian Wan, Runpeng Yu, Gongfan Fang, and Xinchao Wang. 2025. Cot-valve: Length-compressible chain-of-thought tuning. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6025–6035.

Zhenyi Shen, Hanqi Yan, Linhai Zhang, Zhanghao Hu, Yali Du, and Yulan He. 2025. Codi: Compressing chain-of-thought into continuous space via self-distillation. *arXiv preprint arXiv:2502.21074*.

Yang Sui, Yu-Neng Chuang, Guanchu Wang, Jiamu Zhang, Tianyi Zhang, Jiayi Yuan, Hongyi Liu, Andrew Wen, Shaochen Zhong, Na Zou, Hanjie Chen, and Xia Hu. 2025. Stop overthinking: A survey on efficient reasoning for large language models. *Transactions on Machine Learning Research*.

Kimi Team, Angang Du, Bofei Gao, Bowei Xing, Changjiu Jiang, Cheng Chen, Cheng Li, Chenjun Xiao, Chenzhuang Du, Chonghua Liao, and 1 others. 2025. Kimi k1. 5: Scaling reinforcement learning with llms. *arXiv preprint arXiv:2501.12599*.

Yue Wang, Qiuzhi Liu, Jiahao Xu, Tian Liang, Xingyu Chen, Zhiwei He, Linfeng Song, Dian Yu, Juntao Li, Zhuosheng Zhang, and 1 others. 2025. Thoughts are all over the place: On the underthinking of o1-like llms. *arXiv preprint arXiv:2501.18585*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, brian ichter, Fei Xia, Ed H. Chi, Quoc V Le, and Denny Zhou. 2022. Chain of thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*.

Zihao Wei, Liang Pang, Jiahao Liu, Jingcheng Deng, Shicheng Xu, Zenghao Duan, Jingang Wang, Fei Sun, Xunliang Cai, Huawei Shen, and Xueqi Cheng. 2025. The evolution of thought: Tracking llm overthinking via reasoning dynamics analysis.

Heming Xia, Chak Tou Leong, Wenjie Wang, Yongqi Li, and Wenjie Li. 2025. Tokenskip: Controllable chain-of-thought compression in llms. *arXiv preprint arXiv:2502.12067*.

Junjie Yang, Ke Lin, and Xing Yu. 2025. Think when you need: Self-adaptive chain-of-thought learning. *arXiv preprint arXiv:2504.03234*.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. 2024. Tree of thoughts: Deliberate problem solving with large language models. *Advances in Neural Information Processing Systems*, 36.

Jingyang Yi and Jiazheng Wang. 2025. Shorterbetter: Guiding reasoning models to find optimal inference length for efficient reasoning. *arXiv preprint arXiv:2504.21370*.

Shangziqi Zhao, Jiahao Yuan, Jinyang Wu, Zhenglin Wang, Guisong Yang, and Usman Naseem. 2025. Can pruning improve reasoning? revisiting long-cot compression with capability in mind for better reasoning. *arXiv preprint arXiv:2505.14582*.

A Reproducibility Details

A.1 Training Details

We fine-tune `DeepSeek-R1-Distill-Qwen-7B` and `DeepSeek-R1-Distill-Llama-8B` using LoRA with `bfloat16` precision. The maximum sequence length is 8192, the random seed is 1, and the per-device train/eval batch size is 1/1 with `gradient_accumulation_steps=16`. We use a learning rate of 1×10^{-4}, LoRA rank 8, LoRA alpha 32, and a warmup ratio of 0.05. Evaluation, checkpoint saving, and logging are performed every 100, 100, and 5 steps, respectively.

For the experiments reported in Table 1, we train self-distilled models for 6 epochs but teacher-guided models for 2 epochs. This choice is intentional: teacher-guided training showed severe overfitting at 6 epochs, and using 2 epochs yields a stronger and fairer teacher-guided baseline. However, for the supervision-budget comparison in Figure 3, teacher-guided models are trained for 6 epochs to keep the training schedule consistent across budget settings.

A.2 Inference Details

At inference time, we set the maximum generation length to 16384 tokens for AIME 2024 and 8192 tokens for GSM8K and Math 500 to avoid artificial truncation. We use the same decoding configuration across compared methods for fair evaluation.

A.3 Gemini API Configuration

Our annotation pipeline relies on a single external language model, **Gemini-2.5-Flash**, which is used to perform the structured annotation tasks in our pipeline, including taxonomy labeling, reasoning tree construction, and ECN conclusion correctness judgment.

To improve reproducibility, we use a fixed decoding configuration across all annotation stages.

Model. Gemini-2.5-Flash (Google Gemini API).

Decoding configuration. Temperature is set to 0.0, while top-p and top-k sampling are disabled. We do not impose a strict limit on the maximum output tokens, and the thinking budget is left unrestricted during annotation.

API usage and cost. All annotation calls are executed through the official Google Gemini Batch API. The same model version and decoding configuration are used throughout the entire dataset

construction pipeline. The total API cost for processing the 1000 training examples is less than $20.

A.4 Annotation Prompts

In this section, we provide the prompts used in our model-assisted annotation pipeline, including taxonomy labeling, tree construction, and ECN conclusion judgment.

Taxonomy labeling prompt:

```
You will be given a list of reasoning segment
    nodes, each labeled
(e.g., N1, N2, ...), representing steps in a
    reasoning trace.

Your task has to strictly follow three steps:

First, analyze each node:
Carefully read all reasoning nodes and provide a
     brief analysis of each
node's function in the reasoning process.

Second, assign reasoning strategies:
Using the taxonomy provided below, identify the
    primary reasoning
strategy each node represents. Some longer nodes
     may involve multiple
strategies if so, also indicate a secondary
    strategy. If no secondary
applies, leave it as "None".

Finally, strictly format your output as JSONL as
     below:

{"id": "N1", "taxonomy_primary_type": "
    verification", "taxonomy_secondary_type":
    null}
{"id": "N2", "taxonomy_primary_type": "
    verification", "taxonomy_secondary_type": "
    Exploration"}

Reasoning Taxonomy

There are 5 types of reasoning strategies:

Backtracking:
The node revisits and modifies a previous step
    or assumption to correct
an error, resolve a conflict, or incorporate a
    new insight that alters
the reasoning path.

Verification:
The node tests or confirms the correctness of a
    specific claim,
assumption, or result without modifying the
    reasoning path.

Exploration:
The node proposes new hypotheses, possibilities,
     or approaches to the
problem in an open-ended manner, without
    committing to a definitive
solution.

Clarification:
The node rephrases, restates, or defines terms,
    assumptions, or problem
constraints to reduce ambiguity and enhance
    understanding.

Conclusion:
The node synthesizes prior reasoning to assert a
     final or intermediate
solution, judgment, or result.

Node:
["N1: <think>\nFirst, I recognize that the area
    of a circle is given by
the formula \( A = \pi r^2 \). Given that the
    area is \( 49\pi \), I can
set up the equation \( 49\pi = \pi r^2 \). To
    find the radius, I'll divide
both sides of the equation by \( \pi \), which
    simplifies to \( r^2 = 49 \).
Taking the square root of both sides, I get \( r
     = \sqrt{49} \), and since
the radius can't be negative, \( r = 7 \).
    Therefore, the radius of the
circle is 7 units."]
```

Tree construction prompt:

```
You will be given a list of reasoning segment
    nodes, each labeled
(e.g., N1, N2, ...), representing steps in a
    mathematical reasoning
process.

Your task is to analyze the logical structure
    and organize these
segments step by step into an n-ary tree, where:

- Each node corresponds to one or more labels
  (e.g., "N1", or "N3, N4" if they share the
      same logical role).
- Labels must remain sequential. For example, N5
     cannot be placed
  before N4.
- The tree reflects the hierarchical and
    sequential logic of the reasoning.
- Child nodes represent the logic going to the
    next level, such as
  supporting arguments, elaborations, or
      consequences.
- Sibling nodes represent logic staying at the
    same level.

Each time you add a node to the tree, you have
    three options:
A. Add as a sibling to one of the ancestor nodes
     of the current leaf node.
B. Add as a sibling to the current leaf node.
C. Add as a child of the current leaf node.

Provide your thought process and then give the
    tree.
Format the tree output as a structured JSON tree
     using only the node
labels do not include the original text.

Example tree format:
{
  "id": "1",
  "label": "N1",
  "children": [
    {
```

```
      "id": "1.1",
      "label": "N2",
      "children": [
        {"id": "1.1.1", "label": "N3"},
        {"id": "1.1.2", "label": "N4"}
      ]
    }
  ]
}

Nodes:
["N1: <think>\nFirst, I recognize that the area
    of a circle is given by
the formula \( A = \pi r^2 \). Given that the
    area is \( 49\pi \), I can
set up the equation \( 49\pi = \pi r^2 \). To
    find the radius, I'll divide
both sides of the equation by \( \pi \), which
    simplifies to \( r^2 = 49 \).
Taking the square root of both sides, I get \( r
     = \sqrt{49} \), and since
the radius can't be negative, \( r = 7 \).
    Therefore, the radius of the
circle is 7 units."]
```

ECN conclusion judgment prompt:

```
Task:
You are given a reasoning trace divided into
    nodes (e.g., N1, N2, ...).
Each node represents a step in the reasoning
    process.

You are also given a list of conclusion nodes:
    ['N1'] identified by
their serial numbers.

For each conclusion node, do the following step
    by step:

1. Determine whether the conclusion node matches
     the Correct Answer.
   - If the conclusion exactly matches the
       Correct Answer, mark it as 1.
   - If it does not match, mark it as 0.

2. Determine the type of the conclusion node:
   - Intermediate Conclusion: a step that
       supports later reasoning but
     does not directly answer the question.
   - Answering Conclusion: a conclusion intended
        to answer the question,
     regardless of whether it is fully correct or
          ultimately the final answer.

3. Include the question and the Correct Answer:
   - Question: The area of a circle is \(49\pi\)
        square units. What is the
     radius of the circle, in units?
   - Correct Answer: 7

4. Export the result in JSONL format, including
    the node id,
   correctness, and type. Example:

{"conclusion_node": "N5", "is_correct": 0, "type
    ": "Intermediate Conclusion"}
{"conclusion_node": "N10", "is_correct": 1, "
    type": "Answering Conclusion"}
{"conclusion_node": "N12", "is_correct": 1, "
    type": "Answering Conclusion"}

Complete Nodes:
["N1: <think>\nFirst, I recognize that the area
    of a circle is given by
the formula \( A = \pi r^2 \). Given that the
    area is \( 49\pi \), I can
set up the equation \( 49\pi = \pi r^2 \). To
    find the radius, I'll divide
both sides of the equation by \( \pi \), which
    simplifies to \( r^2 = 49 \).
Taking the square root of both sides, I get \( r
     = \sqrt{49} \), and since
the radius can't be negative, \( r = 7 \).
    Therefore, the radius of the
circle is 7 units."]
```

B Additional Experimental Results

The training dynamics show that this structural shift remains stable under self-distilled supervision. Figure 6 plots token accuracy over training steps for four supervision variants: Self-distilled-Full, Self-distilled-ECN, Teacher-Guided-Full, and Teacher-Guided-ECN. Self-distilled-Full maintains the highest token accuracy throughout, indicating strong alignment between the student model and its own traces. Applying ECN under self-distilled supervision lowers token accuracy relative to Self-distilled-Full, but the curve remains stable and gradually improves, showing that pruning changes the supervision signal without destabilizing optimization. In contrast, teacher-guided supervision yields lower token accuracy, consistent with a larger off-policy mismatch between teacher traces and the student model. This interpretation is reinforced by the token-level KL analysis in Figure 2.

We provide additional visualizations that complement the main results. Figure 5 compares the response-length distribution of the base Qwen model and the self-distilled ECN-Qwen model on Math 500 (Level 3). Figure 6 shows token accuracy over training steps for different supervision and pruning strategies. Figure 3 compares self-distilled and teacher-guided training under different supervision budgets.

Metric	SD-ECN	Base	Δ
Earliest final-answer depth	3.181	3.029	+0.152
Post-answer source-node gap	5.762	12.143	-6.381

Table 3: Answer-emergence timing and post-answer continuation on 105 Math 500 Level 3 traces. Lower post-answer gap indicates less redundant continuation.

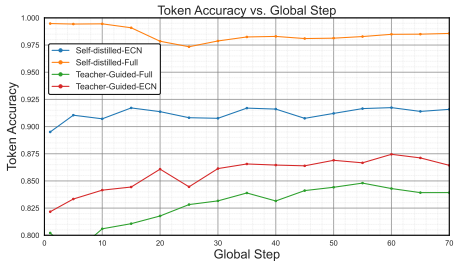

Figure 6: Token accuracy over training steps for different supervision and pruning strategies.

C Human Annotation Evaluation

To evaluate the reliability of the model-assisted annotation pipeline, we conduct a human evaluation comparing LLM-generated annotations with human annotations for two key tasks: taxonomy labeling and conclusion correctness judgment.

C.1 Annotation Protocol

To evaluate the reliability of the model-assisted annotation pipeline, we conduct a human evaluation on a randomly sampled subset of the training data. Specifically, we randomly sample 20 reasoning traces from the PRM-12k training set. After segmentation, these traces contain a total of 463 reasoning nodes, each associated with a taxonomy label predicted by the LLM.

One human annotator manually reviews and labels the sampled nodes using the same annotation guidelines described in Appendix A.4. For each node, the annotator assigns the primary taxonomy label (*Backtracking*, *Verification*, *Exploration*, *Clarification*, or *Conclusion*). For nodes identified as conclusions, the annotator additionally determines whether the node corresponds to the correct answer and whether it should be categorized as an *Intermediate Conclusion* or an *Answering Conclusion*.

C.2 Taxonomy Labeling Accuracy

We evaluate taxonomy labeling by measuring the agreement between the primary taxonomy labels generated by Gemini-2.5-Flash and the human annotations. On the sampled 463 reasoning nodes, the agreement rate between the LLM annotations and the human labels is 88.9%, indicating a high level of consistency.

Most disagreements arise from the heuristic segmentation procedure. In some cases, a single reasoning segment may simultaneously exhibit multiple reasoning functions (e.g., *Exploration* together with an *Intermediate Conclusion*). When such mixed reasoning occurs, the human annotator and the LLM may assign different primary labels, even though both labels are reasonable interpretations of the segment. Therefore, these cases should not necessarily be interpreted as clear annotation errors.

C.3 Conclusion Judgment Accuracy

We further evaluate the reliability of the ECN conclusion judgment step. For this evaluation, we randomly sample 50 reasoning traces and compare the LLM predictions with human annotations.

Two aspects are evaluated: (1) *correctness detection*, which determines whether a predicted conclusion matches the correct answer, and (2) *first-*

conclusion identification, which checks whether the earliest answering conclusion predicted by the LLM matches the one identified by the human annotator.

On the sampled traces, the correctness detection accuracy reaches 100%, and the first-conclusion identification accuracy also reaches 100%. These results suggest that the LLM-based annotation pipeline provides reliable signals for identifying the first correct answering conclusion used in our pruning method.

D Metric Definitions

This section defines the trace-level metrics used in our analysis. Each reasoning trace is first represented as a grouped tree, while the original source reasoning segments are retained as an ordered sequence. We distinguish between metrics computed on the grouped tree structure and metrics computed from the source-node order.

D.1 Notation

We model each reasoning trace as a rooted n-ary tree

$$T = (V, E),$$

where V is the set of grouped tree nodes, E is the set of parent-child edges, and the root has depth 0. Depth is measured in edges.

For any node $v \in V$, let $d(v)$ denote its depth and let

$$b(v) = |\mathrm{child}(v)|$$

denote its branching factor. We define the leaf set as

$$\mathcal{L} = \{\ell \in V : \mathrm{child}(\ell) = \varnothing\},$$

and the internal-node set as

$$\mathcal{I} = \{v \in V : |\mathrm{child}(v)| > 0\}.$$

Each grouped tree node may contain one or more original source reasoning segments. Let

$$S = (N_1, N_2, \ldots, N_m)$$

be the ordered source-node sequence for a trace. For each source node N_i, let $g(N_i) \in V$ denote the grouped tree node that contains it.

Unless otherwise specified, variance terms are computed as population variances. For degenerate cases, such as a single-node tree with no internal nodes, the corresponding internal-node metrics are set to zero.

D.2 Tree-Node Metrics

The metrics in this subsection are computed on the grouped tree T.

Total Tree Nodes. The total number of grouped tree nodes is

$$\mathrm{TotalTreeNodes}(T) = |V|.$$

Maximum Depth. The maximum root-to-leaf depth is

$$\mathrm{MaxDepth}(T) = \max_{\ell \in \mathcal{L}} d(\ell).$$

Maximum Width. For each depth h, let

$$W_h = |\{v \in V : d(v) = h\}|$$

be the number of grouped tree nodes at that depth. The maximum width is

$$\mathrm{MaxWidth}(T) = \max_{h} W_h.$$

Width-to-Depth Ratio. The width-to-depth ratio compares the widest layer of the tree with its maximum depth:

$$\mathrm{WDR}(T) = \frac{\mathrm{MaxWidth}(T)}{\mathrm{MaxDepth}(T)}.$$

For single-node trees, where $\mathrm{MaxDepth}(T) = 0$, we set $\mathrm{WDR}(T) = 0$.

Mean Leaf Depth. The mean depth of leaf nodes is

$$\mathrm{MeanLeafDepth}(T) = \frac{1}{|\mathcal{L}|} \sum_{\ell \in \mathcal{L}} d(\ell).$$

Leaf-Depth Variance. The variance of leaf depths is

$$\mathrm{LeafDepthVar}(T) = \frac{1}{|\mathcal{L}|} \sum_{\ell \in \mathcal{L}} (d(\ell) - \mathrm{MeanLeafDepth}(T))^2.$$

Mean Branching Factor. The mean branching factor over internal nodes is

$$\mathrm{MeanBranching}(T) = \frac{1}{|\mathcal{I}|} \sum_{v \in \mathcal{I}} b(v).$$

Maximum Branching Factor. The maximum branching factor over internal nodes is

$$\mathrm{MaxBranching}(T) = \max_{v \in \mathcal{I}} b(v).$$

Branching Variance. The variance of internal-node branching factors is

$$\mathrm{BranchingVar}(T) = \frac{1}{|\mathcal{I}|} \sum_{v \in \mathcal{I}} (b(v) - \mathrm{MeanBranching}(T))^2.$$

Structural Efficiency. Let

$$\mathrm{LongestPathNodes}(T) = \max_{\ell \in \mathcal{L}} |P(r, \ell)|$$

denote the maximum root-to-leaf path length counted in nodes rather than edges, where $P(r, \ell)$ is the set of nodes on the path from the root r to leaf ℓ. We define

$$\mathrm{StructuralEfficiency}(T) = \frac{\mathrm{LongestPathNodes}(T)}{\mathrm{TotalTreeNodes}(T)}.$$

This quantity measures the fraction of grouped tree nodes covered by the longest reasoning path.

Sackin Index. The Sackin index is the sum of leaf depths:

$$\mathrm{Sackin}(T) = \sum_{\ell \in \mathcal{L}} d(\ell).$$

Generalized Colless Index. For any internal node v, let $L(u)$ be the number of descendant leaves under child $u \in \mathrm{child}(v)$. We define the local imbalance at v as

$$\mathrm{LocalImbalance}(v) = \sum_{\substack{u_i, u_j \in \mathrm{child}(v) \\ i<j}} |L(u_i) - L(u_j)|.$$

The generalized Colless index is then

$$\mathrm{GenColless}(T) = \sum_{v \in \mathcal{I}} \mathrm{LocalImbalance}(v).$$

Larger values indicate greater imbalance among sibling subtrees.

D.3 Source-Node Metrics

The metrics in this subsection are defined from the ordered source-node sequence S. When depth information is needed, the corresponding source node is mapped back to its grouped tree node through $g(\cdot)$.

Final-Answer Proxy. Let $A^\star$ be the final-answer proxy extracted from the latest source node that contains either a boxed answer or the literal string `Final Answer`. If no such proxy is found, the trace is excluded from the answer-boundary metrics.

Earliest Final-Answer Node. Given $A^\star$, let

$$k^\star = \min\{i \in \{1, \ldots, m\} : A^\star \text{ appears in } N_i\}.$$

The earliest final-answer node is

$$\mathrm{EarliestFinalAnswerNode}(S) = N_{k^\star}.$$

Earliest Final-Answer Depth. After locating $N_{k^\star}$ in source-node order, we map it to the grouped tree and report its depth:

$$\mathrm{EarliestFinalAnswerDepth}(S, T) = d(g(N_{k^\star})).$$

Thus, the answer boundary is detected in source-node order and then evaluated with respect to the grouped-tree structure.

Post-answer Source-Node Gap. The remaining source-node distance after the earliest final-answer node is

$$\mathrm{PostAnswerSourceGap}(S) = m - k^\star.$$

Equivalently, this is the number of source nodes that appear after the earliest occurrence of the final-answer proxy.