

Adaptive Test-Time Compute for Pedagogically Grounded Reasoning in LLMs

Stanford CS224N Custom

Medhya Goel Jason Chon Isha Jain

Department of Computer Science

Stanford University

{medhya, jchon5, ishajain}@stanford.edu

TA mentor: Minsik Oh. No external collaborators/mentors. No one is sharing the project. We aren't using late days.

Abstract

Large language models (LLMs) show promise as scalable tutors for mathematical reasoning, but models that generate high-quality explanations are often too expensive for widespread educational use. We study compute-efficient methods to improve the pedagogical quality of smaller open-source LLMs. We frame the task as a teacher–student setup on the MATH dataset, where a stronger model produces explanations and a weaker model uses them to solve problems. We explore two approaches. First, we use token-level entropy as an uncertainty signal to adaptively allocate test-time compute, regenerating explanations only when the teacher appears uncertain. Second, we train a verifier to predict whether a teacher explanation will enable downstream student success, optimizing for pedagogical usefulness beyond just answer correctness. Teacher explanations significantly improve student accuracy over a no-explanation baseline, and entropy-based filtering further improves performance while allocating compute selectively. A finetuned verifier also substantially outperforms a prompted baseline, achieving 71.88% test accuracy. These results suggest that adaptive compute and pedagogically grounded verification can improve explanation quality in smaller open-source LLMs.

1 Introduction

Large language models (LLMs) have rapidly improved their ability to explain complex concepts and provide step-by-step reasoning, demonstrating the potential to expand access to high-quality educational support, particularly for learners without access to human instructors. However, most high-performing models capable of generating pedagogically useful explanations remain too expensive for low-resource settings.

While substantial work has focused on improving reasoning accuracy in LLMs, relatively little attention has been paid to improving the quality of explanations that help learners understand the reasoning process (without simply revealing the final answer) in a computationally efficient way. Allocating additional test-time compute can significantly improve performance on open-ended tasks, which pedagogical explanation generation is. However, traditional approaches allocate a fixed compute budget to every problem, regardless of the model's uncertainty or the difficulty of the reasoning task. Additionally, traditional methods of training verifier models to score response quality require complex training setups and expensive models.

In this work, we investigate adaptive test-time compute for pedagogically grounded reasoning in large language models. We investigate two main approaches. First, we use token-level entropy to dynamically allocate reasoning effort for pedagogical explanations. Second, we finetune a verifier model that evaluates explanations based on their ability to support downstream learner success rather than simply verifying correctness. We demonstrate that entropy-based adaptive compute

allocation improves explanation quality and efficiency when using smaller open-source models and that verifier-based approaches can significantly improve understanding.

2 Related Work

2.1 Test-Time Compute for Reasoning

Recent work shows that allocating additional compute at inference time can significantly improve reasoning performance in large language models (LLMs). Chain-of-Thought prompting encourages models to generate intermediate reasoning steps, improving performance on complex tasks such as mathematics and symbolic reasoning [1]. Self-Consistency further improves performance by sampling multiple reasoning trajectories and selecting the most consistent answer [2]. Other approaches explore scaling reasoning through search or iterative refinement, such as Tree-of-Thought reasoning [3].

Other works investigate dynamically deciding when to generate additional solutions or verify existing ones. *When to Solve vs. When to Verify: Compute-Optimal Problem Solving* studies how systems can allocate test-time compute between generation and verification to optimize reasoning performance under a fixed budget [4]. Related work also explores internal uncertainty signals such as token-level entropy to characterize reasoning difficulty and guide adaptive inference strategies [5]. Our work builds on this line of research by studying adaptive compute allocation for pedagogical explanation generation rather than answer prediction.

2.2 Verifier-Based Reasoning Systems

A growing body of work introduces verifier models that evaluate candidate solutions. Early work trained verifiers to evaluate mathematical reasoning produced by generator models, forming a prover-verifier framework for solving math problems [6]. Subsequent work introduced process reward models that evaluate intermediate reasoning steps rather than final answers, improving reasoning reliability in mathematical problem solving [7].

Recent research explores generative verification pipelines in which LLMs generate candidate reasoning traces and separate models critique or verify them [4]. These approaches primarily focus on verifying correctness of reasoning or final answers. In contrast, our work introduces a verifier trained to evaluate the *pedagogical quality* of explanations, focusing on whether an explanation improves downstream student understanding rather than whether it produces the correct solution.

2.3 LLMs for Mathematical Explanation and Tutoring

Large language models are increasingly studied as tools for automated tutoring and step-by-step explanation generation. Benchmarks such as the MATH dataset evaluate models on complex mathematical reasoning tasks [8], while more recent work investigates how LLMs can generate explanations that support learning rather than simply produce correct answers. Some approaches incorporate structured problem-solving frameworks, such as Polya-style reasoning strategies, to improve pedagogical explanations generated by language models [9, 10].

However, most prior work evaluates explanation quality using human judgments or tutoring benchmarks rather than incorporating automated verification mechanisms. Our work bridges this gap by introducing a verifier trained on student success signals and combining it with adaptive test-time compute. This allows explanation quality to be evaluated in terms of knowledge transfer while keeping inference costs manageable for smaller open-source models.

3 Approach

We consider two main approaches to the general task of generating pedagogically useful explanations on complex reasoning tasks. Our framework relies on two models. The teacher model is a large, higher-capacity pretrained model (Llama 3.1-8b-Instruct). The student model is a smaller, weaker pretrained model (Llama 3.2-3b-Instruct). For all models and all experiments, responses are generated probabilistically with a temperature of 0.8, top_p of 0.9, maximum of 1024 new tokens, and random

seed of 42 to ensure reproducibility. All prompts are in the Appendix. All of our experiments were run using code that we wrote. We used libraries like HuggingFace.

3.1 Token Entropy-Based Methods

First, we treat generation-time token entropy as a proxy for model uncertainty and generate additional explanations for the same problem until the teacher model is sufficiently certain. Intuitively, when the model’s probability distribution over the vocabulary is more uniform, the model is less confident about the next token. Such uncertainty may correspond to weaker reasoning trajectories.

Formally, let $p_\theta(w \mid x, y_{<t})$ denote the model’s predicted probability of token w at timestep t . The token-level entropy is defined as

$$H_t = - \sum_{w \in \mathcal{V}} p_\theta(w \mid x, y_{<t}) \log p_\theta(w \mid x, y_{<t}), \quad (1)$$

where $\mathcal{V}$ is the vocabulary.

Given an explanation of length T , we compute a scalar uncertainty score from the token-level entropies. We formulated two aggregation strategies:

- **Mean entropy:**

$$s_{\text{mean}} = \frac{1}{T} \sum_{t=1}^T H_t$$

- **Tail entropy:**

$$s_{\text{p90}} = \text{percentile}_{90}(\{H_t\}_{t=1}^T)$$

Tail entropy captures localized, high-uncertainty regions that may occur during critical reasoning steps.

3.2 Verifier Model

Previous works such as (Tao & Li, 2025)[11] have explored the idea of aligning more powerful, larger models by utilizing weaker, smaller models. Our approach is an extension of this idea, taking advantage of a computationally cheap model to efficiently train a larger model into a competent verifier. Specifically, we optimize the verifier to classify whether explanations generated by the teacher, when given to the student, enable the student to reach the correct answer. The verifier is thus trained to identify pedagogical explanations that are high quality in the sense that they enable a weaker model to derive an answer that it normally may not have the ability to. The verifier is a large pretrained model (Llama 3.1-8b-Instruct, separate weights from the teacher) with an added linear classification head, finetuned through LoRA on a binary classification task. See the Verifier Training for more details.

3.3 Baselines

As baselines, we evaluated the accuracy of the teacher model and student model on a 500-question random subsample of the full dataset (with the same vanilla prompt to solve the question rather than explain it), as well as the accuracy of the student model with and without a teacher explanation (with the same non-vanilla prompt instructing the student model to use the explanation to solve the problem). We pass the first teacher-generated explanation per question to the student model.

Experiment Name	With Teacher-Generated Explanation?	Accuracy
Vanilla Student	No	0.464
Vanilla Teacher	No	0.556
Student	No	0.364
Student	Yes	0.486

Table 1: Baseline accuracies of teacher and student model

The baseline results verify that the teacher model is significantly stronger than student model out of the box and even naively-selected teacher model explanations improved student model accuracy.

For the verifier approach, we additionally evaluate the ability of the teacher model (whose weights serve as the pretrained base for the verifier) to classify explanation quality before any finetuning as a baseline. We cannot directly run the pretrained model in a classification setting because this would involve using an untrained linear head, so we instead provide the model with the explanation or the problem and explanation and prompt it to output whether or not it believes the explanation would help a weaker model reach a correct answer. This baseline is shown in Table 4.

4 Entropy Experiments

4.1 Data

We use the MATH dataset [8] which contains 12,500 competition-level mathematical problems of varying difficulty across 7 mathematical subjects and 6 difficulty levels, along with ground-truth solutions to each. Each solution can be written as a number or numerical expression.

4.2 Procedure & Evaluation method

For the entropy experiments, we randomly subsample 500 questions and perform the following procedure using 8 different entropy experimental conditions for each problem x_i :

1. Prompt the teacher model (Llama-3.1-8B-Instruct) to generate a pedagogical explanation e_i and final answer describing how to solve x_i .
2. During generation, we compute token-level entropy from the teacher model’s output distribution and aggregate it into a scalar explanation score $s(e_i)$ using either mean entropy or the 90th percentile (P90) entropy across tokens.
3. Depending on the experimental condition, we apply an entropy threshold rule to determine whether the explanation should be accepted or whether another explanation should be generated. If the threshold is not satisfied, the teacher regenerates an explanation up to a maximum of $K = 5$ passes.
4. The selected explanation e_i (with final answer removed) is then provided to the student model (Llama-3.2-3B-Instruct), which produces a numerical answer a_i to the problem.
5. Let s_i denote the ground-truth numerical solution from the MATH dataset. Student accuracy for an example is computed as $\mathbb{1}[a_i = s_i]$ and is averaged over the whole subset.

We compare student accuracy using the selected explanations with baseline across all runs. We also look at average number of forward passes used, recovery rate, threshold satisfaction rate, and qualitatively analyze the teacher explanations (are they too specific, vague, or pedagogical enough).

4.3 Experimental Details

All experiments were run with the same hyperparameters mentioned in Approach. We evaluate two filtering settings, with 4 experiments each. In the *noSkip* setting, all explanations are eligible to be passed to the student regardless of whether the teacher’s predicted answer is correct. In the *skipWrong* setting, explanations are only passed to the student when the teacher’s predicted answer is correct. The *skipWrong* condition assumes oracle access to teacher correctness and is used only for analysis, allowing us to isolate the effect of entropy independent of teacher accuracy.

For each filtering setting, we try 4 different thresholds.

Condition	Threshold Definition	Threshold Value
Medium Entropy	Mean entropy, midpoint	0.2197
Low Entropy	Mean entropy, correct-only median	0.1618
High Entropy	Mean entropy, incorrect-only median	0.2775
Low Entropy (P90)	P90 entropy, correct-only median	0.6811

Table 2: Summary of entropy-gated experiment configurations.

The Low Entropy threshold value is defined as the median of the mean token-level entropy of explanations that led to the student model answering correctly in the baseline. High Entropy is the same, but for explanations that led the student to be incorrect. Medium Entropy is the midpoint

between Low and High Entropy. Low Entropy (P90) is the median of P90 entropy in the correct explanations from the baseline.

5 Verifier Training

5.1 Data

We construct our own classification dataset for the verifier finetuning consisting of tuples (x_i, e_i, y_i) where x_i is the problem string, e_i is a pedagogical explanation, and $y_i \in \{0, 1\}$ is the target label. For each problem x_i in the MATH dataset, we:

1. Prompt the teacher to generate a pedagogical explanation e_i of how to solve x_i
2. Prompt the student to solve x_i relying solely on e_i . We capture the student’s numerical answer a_i
3. Let s_i be the ground-truth numerical solution to x_i from the MATH dataset. We define $y_i = \mathbb{1}[s_i = a_i]$ where $\mathbb{1}$ is the indicator function. Throughout this report, we refer to explanations where $y_i = 1$ as "high quality" or "helpful"

We run this process on the train and test split datasets of MATH. To extract teacher explanations, we prompt the model to enclose its explanations in special tags. For 1.6% of the train set and 1.14% of the test set, the teacher’s responses do not fit these requirements. These examples are filtered out from the final dataset. We then randomly split off 20% of the train set to create a validation dataset using a set random seed. After filtering and splitting, the train, validation, and test datasets are 5911, 1469, and 4943 examples long, respectively.

5.2 Evaluation method

We evaluate the finetuned verifier primarily on its accuracy and loss on the held-out test set. We measure several other metrics for additional quantitative and qualitative analysis: precision, recall, F1 score, prediction confidence, and the model confusion matrix.

5.3 Experimental details

The verifier model is trained to optimize binary cross entropy loss. Linear warmup occurs over the first 10% of training steps followed by linear learning rate decay. We try two different finetuning setups on the base verifier: providing only the explanation or providing both the problem and explanation in a template string. For both, the wall-clock time on an A100 was approximately 90 minutes.

Learning rate	Epochs	Batch size	LoRA rank	LoRA alpha	LoRA dropout	Weight decay
$5e^{-5}$	2	4	16	32	0.15	0.1

Table 3: Hyperparameters for finetuning the verifier.

6 Results & Analysis

6.1 Entropy Experiments

Low entropy often corresponds to more structured reasoning, but it does not guarantee correctness: the teacher is occasionally confident and wrong. Figure 1 illustrates this limitation. Entropy-based filtering can nevertheless improve explanations by allocating additional compute when the model appears uncertain. Figure 2 shows an example of this process. The initial generation (Pass 1) has relatively high entropy and produces a generic explanation that leads to an incorrect answer. After additional sampling, a later generation (Pass 3) has substantially lower entropy and produces a more concrete, problem-specific explanation, which is selected by the entropy-gating procedure. Figure 2 shows that experiments with more passes tend to achieve larger improvements. Some configurations with similar numbers of passes yield substantially different gains like Incorrect median SkipWrong, indicating that additional compute alone does not explain the improvements. Stricter entropy thresholds is related to lower threshold-met rates (if entropy threshold is met within 5 passes) and, thus, higher average passes. For details on this, refer to Table 7 in Appendix.

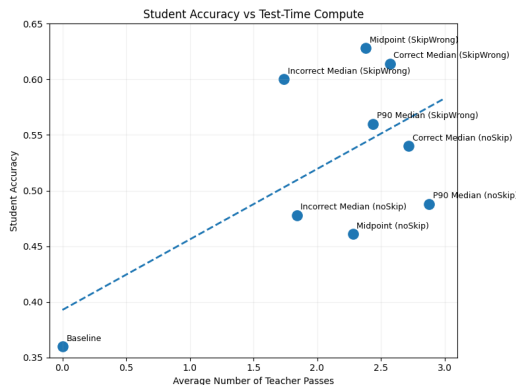
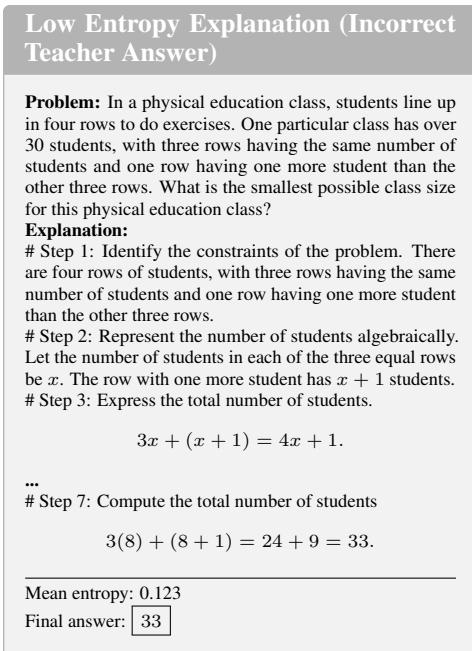


Figure 1: Left: Example of a low-entropy teacher explanation that nevertheless produces an incorrect answer (some text is removed for space). Right: Student accuracy improvement vs. test-time compute (average number of teacher passes).

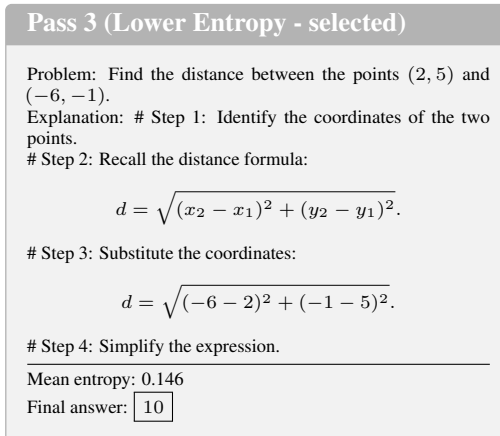
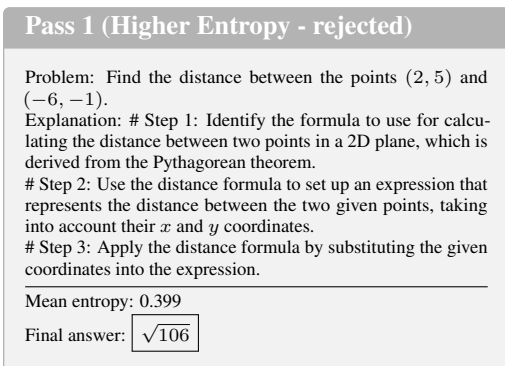


Figure 2: Example of entropy-guided explanation refinement across teacher passes for the same problem.

Figure 3 summarizes student accuracy across the entropy-gated strategies. Providing explanations improves student accuracy relative to the baseline, and the High Entropy condition performs substantially worse than the Low and Medium entropy conditions. This supports the hypothesis that lower-entropy explanations tend to be more useful for the student model. Across most configurations, the SkipWrong setting also outperforms NoSkip, indicating that teacher correctness remains a strong signal. However, this pattern does not hold for the High Entropy condition. In that setting, the student sometimes succeeds even when the teacher’s answer is incorrect. In the NoSkip High Entropy experiment, for example, the teacher produced an incorrect answer but the student recovered the correct solution in 120/500 cases, suggesting that higher-entropy explanations may sometimes expose exploratory reasoning paths that still help the student reach the correct answer.

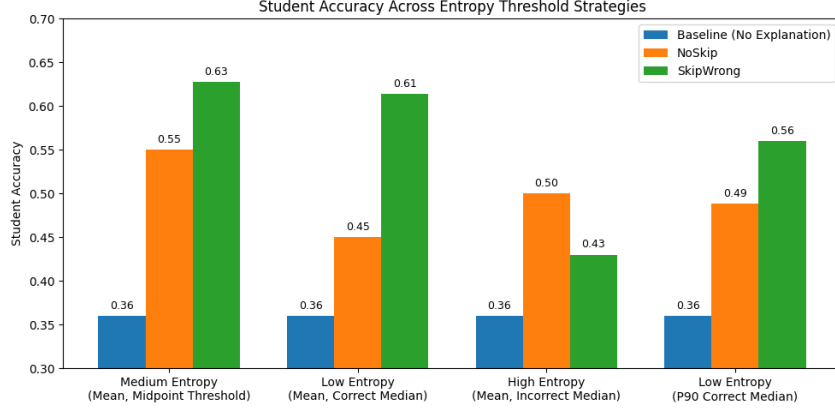


Figure 3: Student accuracy across entropy-gated strategies. Condition definitions are provided in Section 4.3.

Overall, explanations recover between 33–49% of baseline mistakes, indicating that entropy-filtered explanations frequently enable the student model to correct its initial errors. For more recovery rate results, see Table 6 in Appendix. Surprisingly, Medium Entropy does better than Low Entropy. This result suggests that allowing moderately "uncertain" explanations can sometimes be more useful pedagogically for the student.

6.2 Verifier

As shown in Table 4, we find that finetuning improves the performance of the model by up to 30.08%. Additionally, providing both the problem and explanation during finetuning leads to better verifier performance. As a result, we treat the verifier trained with this input configuration as our final model, and the remaining metrics are only measured for it. Training curves are available in the Appendix.

Model	Input	Loss	Acc.
Verifier	Explanation	0.6089	68.23%
Verifier	Problem + Expl.	0.5466	71.88%
Baseline	Explanation	0.9580	48.8%
Baseline	Problem + Expl.	0.9584	48.8%

Table 4: Verifier and baseline performance.

Precision	Recall	F1
0.4946	0.6410	0.5584

Table 5: Additional verifier metrics.

Table 5 shows that precision, recall, and F1 score are all suitable with higher recall suggesting the model can often accurately identify helpful, higher quality explanations. Figure 5 shows that the model is often correctly confident when identifying helpful explanations.

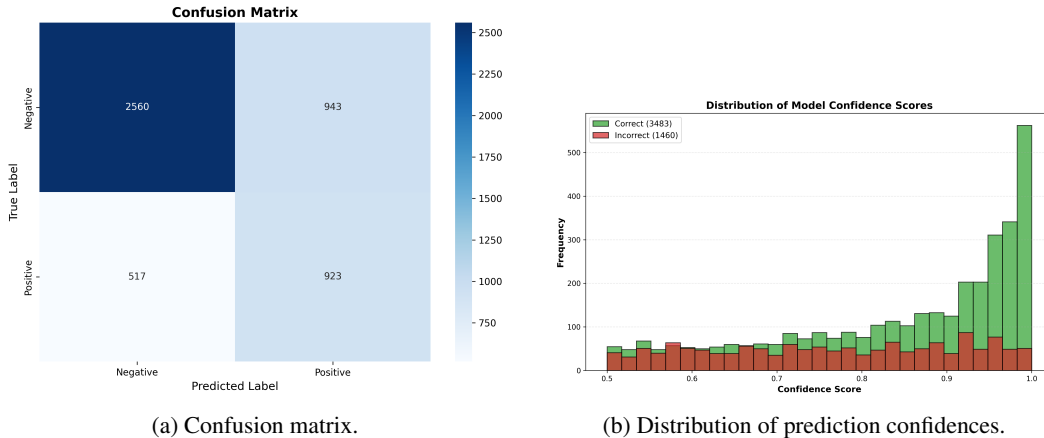


Figure 4: Metrics related to the predictions of the final verifier.

Predicted as Helpful to Student	Predicted as Unhelpful to Student
Problem: Find x such that $\log_5(x - 18) = 2$. Explanation: # Step 1: Recall the definition of logarithms, which states that if $\log_b(y) = z$, then $b^z = y$. # Step 2: Apply this definition to the given equation to rewrite it in exponential form, with base 5. # Step 3: Solve the resulting exponential equation for x by isolating the variable.	Problem: Let $f(x)$ and $g(x)$ be even functions. Is $f(x)g(x)$ even, odd, or neither? Enter "odd", "even", or "neither". Explanation: # Step 1: Recall the definition of even and odd functions. An even function satisfies $f(-x) = f(x)$ for all x in its domain, while an odd function satisfies $f(-x) = -f(x)$ for all x in its domain. # Step 2: Since $f(x)$ and $g(x)$ are even functions, we know that $f(-x) = f(x)$ and $g(-x) = g(x)$. # Step 3: To determine if $f(x)g(x)$ is even, odd, or neither, we need to examine the properties of the product of two functions. We can use the property that the product of two even functions is even.

Figure 5: Examples of teacher explanations predicted as helpful and unhelpful by the verifier.

As seen in Figure 5, the model does not appear to rely on superficial features such as explanation length, as explanations of varying lengths are classified into both categories. Instead, the predictions suggest the verifier has learned more generalizable indicators of explanation quality, although what exactly these are remains unclear through simple manual inspection. Notably, explanations that explicitly reveal the final answer are frequently classified as helpful, indicating a potential bias toward explanations that implicitly expose the solution.

These results were better than expected given that explanation quality is subjective and whether a given explanation will help a student model is inherently stochastic, to some extent. These factors, as well as the limited dataset, created a somewhat low signal task. Nevertheless, the verifier appears to have learned well, suggesting that leveraging weaker models to supervise stronger ones may be drastically cheaper than traditional verifier training methods while still leading to robust verifiers.

7 Conclusion

We studied adaptive test-time compute for pedagogically grounded reasoning in LLMs using two complementary approaches: entropy-based explanation selection and verifier-based explanation filtering. Teacher explanations significantly improved student performance over a no-explanation baseline, and entropy served as a useful uncertainty signal for allocating additional generation passes. Lower and medium-entropy explanations generally produced stronger student results than high-entropy explanations, though entropy alone was not a perfect proxy for correctness. We also trained a verifier to predict whether a teacher explanation would enable a weaker student model to reach the correct answer. The finetuned verifier substantially outperformed the baseline. Crucially, such a verifier can be used in inference-time compute methods, such as by deciding whether an explanation is good enough to output or if more attempts are needed, or by serving as the scorer in a Best-of-N approach. Overall, our results indicate that adaptive compute allocation and learned explanation verification are promising strategies for improving explanation quality in smaller open-source models. Key limitations include the use of small evaluation subsets, oracle filtering in the skipWrong analysis, and measuring explanation quality only through downstream student success. Future work could evaluate these methods on broader educational tasks and integrate entropy-based methods with verifier scoring in a unified adaptive pipeline.

Team contributions

Jason helped with setting up baselines, cloud infrastructure, and prompting in the early stages. Later, he primarily focused on designing the verifier training framework and creating and running the finetuning code. Isha helped with setting and running initial baselines, formulating and creating the 8 entropy related experiments code, and running and analyzing it. Medhya helped with implementing, scaling, debugging, and running pipelines and cloud infrastructure, as well as analysis for the report.

References

- [1] Jason Wei, Xuezhi Wang, Dale Schuurmans, et al. Chain-of-thought prompting elicits reasoning in large language models. *NeurIPS*, 2022.
- [2] Xuezhi Wang, Jason Wei, Dale Schuurmans, et al. Self-consistency improves chain of thought reasoning in language models. *ICLR*, 2023.
- [3] Shunyu Yao, Dian Yu, Jeffrey Zhao, et al. Tree of thoughts: Deliberate problem solving with large language models. *NeurIPS*, 2023.
- [4] Nishad Singhi, Hritik Bansal, Arian Hosseini, Aditya Grover, Kai-Wei Chang, Marcus Rohrbach, and Anna Rohrbach. When to solve vs. when to verify: Compute-optimal problem solving. *arXiv preprint arXiv:2504.01005*, 2025.
- [5] Shenzhi Wang, Le Yu, Chang Gao, Chujie Zheng, Shixuan Liu, Rui Lu, Kai Dang, Xionghui Chen, Jianxin Yang, Zhenru Zhang, Yuqiong Liu, An Yang, Andrew Zhao, Yang Yue, Shiji Song, Bowen Yu, Gao Huang, and Junyang Lin. Beyond the 80/20 rule: High-entropy minority tokens drive effective reinforcement learning for llm reasoning. *arXiv preprint arxiv:2506.01939*, 2025.
- [6] Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.
- [7] Hunter Lightman, Vineet Kosaraju, Yuri Burda, et al. Let’s verify step by step. *arXiv preprint arXiv:2305.20050*, 2023.
- [8] Dan Hendrycks, Collin Burns, Saurav Kadavath, et al. Measuring mathematical problem solving with the math dataset. *NeurIPS*, 2021.
- [9] George Polya. *How to Solve It*. Princeton University Press, 1945.
- [10] Unggi Lee, Yeil Jeong, Chohui Lee, Gyuri Byun, Yunseo Lee, Minji Kang, and Minji Jeon. Llama-polya: Instruction tuning for large language model based on polya’s problem-solving. *arXiv preprint arxiv:2602.10597*, 2026.
- [11] Leitian Tao and Yixuan Li. Your weak llm is secretly a strong teacher for alignment. In Y. Yue, A. Garg, N. Peng, F. Sha, and R. Yu, editors, *International Conference on Learning Representations*, volume 2025, pages 20100–20122, 2025.

A Appendix

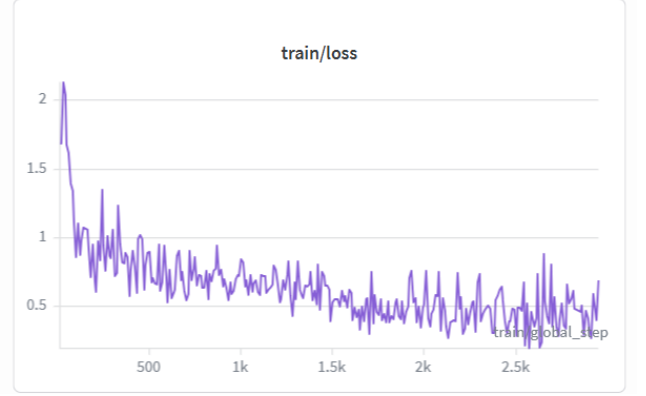
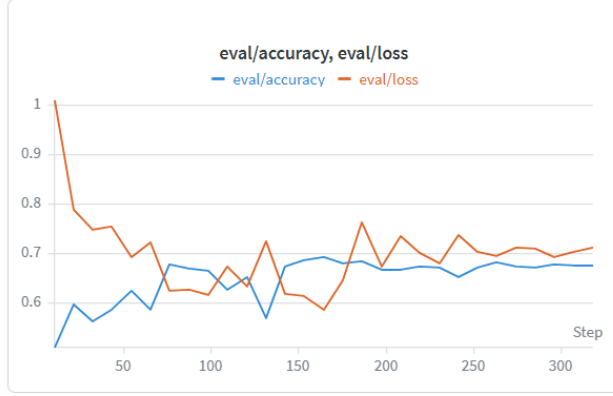
A.1 Models

All baselines and entropy experiments were run on an L4 NVIDIA GPU. Verifier finetuning was executed on an NVIDIA A100.

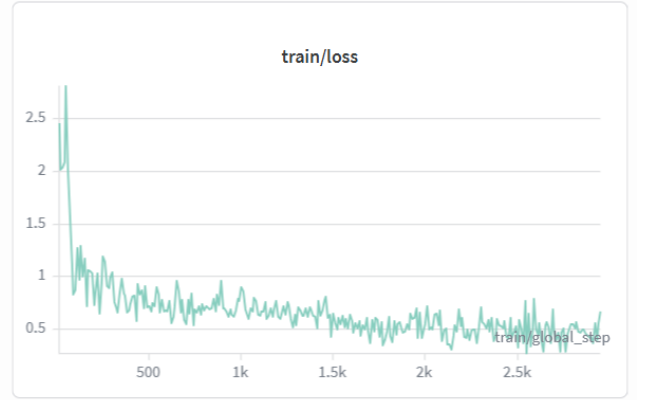
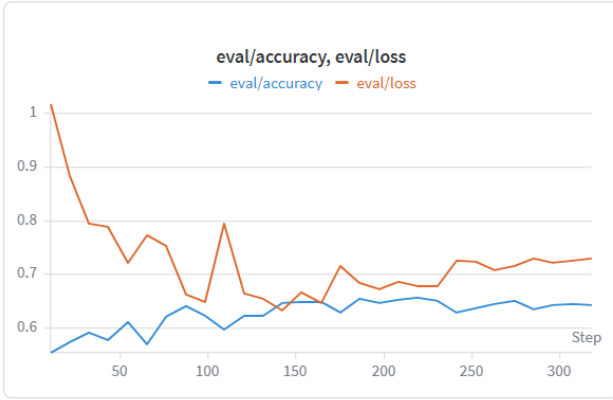
A.2 Teacher Model Explanation Generation Hyperparameters

- Maximum new tokens: `inference.max_new_tokens = 1024`.
- Sampling temperature: `inference.temperature = 0.8`.
- Nucleus sampling: `inference.top_p = 0.9`.
- Batch size (teacher–student pipeline): 1

A.3 Verifier training curves



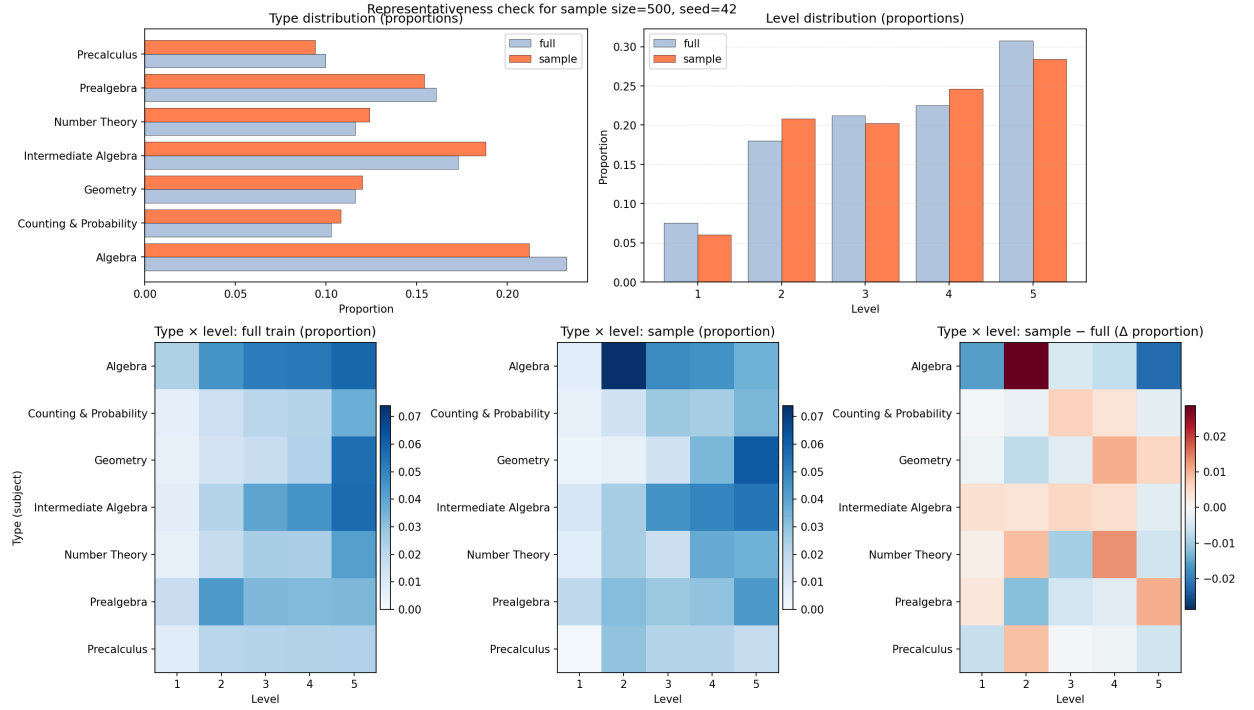
(a) Input contains problem and explanation.



(b) Input only contains explanation.

Figure 6: Verifier training curves under different input settings.

A.4 500 questions are representative subsample of full dataset



A.5 Prompts used for models across experiments

Vanilla Prompt Template

```
<|begin_of_text|><|start_header_id|>system<|end_header_id|>
You are an expert in mathematics and problem solving. Below, the user
has asked you a math problem.
Reason through it carefully and step by step.

The final part of your response should be the numerical answer to the
problem included in \boxed{{}}.
<|eot_id|><|start_header_id|>user<|end_header_id|>
Problem: {problem}
<|eot_id|><|start_header_id|>assistant<|end_header_id|>
```

Teacher Model Prompt Template

```
<|begin_of_text|><|start_header_id|>system<|end_header_id|>
You are an expert in mathematics with a focus on teaching other users
and models how to reason about math.
A user has asked the math problem below. Your task is to generate one
CONCISE explanation for how to
arrive at the problem's solution that accomplishes the following:
```

- 1) The explanation would allow another user/model to figure out how to answer the problem correctly
- 2) The explanation is pedagogical, consisting of high-level approaches and conceptual strategy without writing out any of the actual intermediate mathematical steps. Mentioning high-level, general formulas, theorems, or identities is fine
- 3) DO NOT INCLUDE THE ANSWER IN THE EXPLANATION.

Please format your response as follows, making sure to include all special tags, especially the end tag "</explain>":

--

```
<explain>
# Step 1: ...
# Step 2: ...
...
</explain>
```

The final part of your response should be the numerical answer included in `\boxed{{}}`.

--

```
<|eot_id|><|start_header_id|>user<|end_header_id|>
Problem: {problem}
<|eot_id|><|start_header_id|>assistant<|end_header_id|>
```

Student Model Prompt Template

<|begin_of_text|><|start_header_id|>system<|end_header_id|>
You are a skilled math problem solver. Below, the user has given you a math problem, possibly along with an expert explanation that outlines the strategy, relevant formulas, and common pitfalls for the problem.

If no explanation is provided, reason through the problem on your own and provide your final numerical answer at the end of your response in `\boxed{}`. If there is an explanation, follow the below instructions:

Instructions:

- 1) Read the explanation thoroughly before attempting the problem
- 2) Follow the general strategy outlined in the explanation
- 3) Provide your final numerical answer at the end of your response in `\boxed{}`

Crucially, you must act as a student learning from a teacher's explanation. Do not rely on knowledge of mathematical concepts or steps not mentioned in the explanation.

<|eot_id|><|start_header_id|>user<|end_header_id|>

Problem: {problem}

Expert Explanation:

{explanation}

<|eot_id|><|start_header_id|>assistant<|end_header_id|>

Verifier Baseline Teacher Model Prompt Template (only explanation)

<|begin_of_text|><|start_header_id|>system<|end_header_id|>

You are an expert judge of mathematical explanations.

You will be given:

- 1) A math problem.
- 2) A short explanation that a student used while solving the problem.

Your task is to decide whether this explanation is good enough that a student following it carefully would arrive at the true final numerical answer.

Respond with exactly one word on the first line:
either "CORRECT" or "INCORRECT".

Do NOT show your reasoning, and do NOT restate the problem or explanation.

<|eot_id|><|start_header_id|>user<|end_header_id|>

Student explanation:

{explanation}

<|eot_id|><|start_header_id|>assistant<|end_header_id|>

Verifier Baseline Teacher Model Prompt Template (problem and explanation)

```
<|begin_of_text|><|start_header_id|>system<|end_header_id|>
You are an expert judge of mathematical explanations.

You will be given:
1) A math problem.
2) A short explanation that a student used while solving the problem.

Your task is to decide whether this explanation is good enough that a
student
following it carefully would arrive at the true final numerical answer.

Respond with exactly one word on the first line:
either ‘‘CORRECT’’ or ‘‘INCORRECT’’.

Do NOT show your reasoning, and do NOT restate the problem or
explanation.
<|eot_id|><|start_header_id|>user<|end_header_id|>

Problem:
{problem}
Student explanation:
{explanation}
<|eot_id|><|start_header_id|>assistant<|end_header_id|>
```

A.6 Entropy Experiment Recovery Rates

Experiment	Recovery Rate (%)
Mean Entropy – Midpoint (NoSkip)	33.1
Mean Entropy – Correct Median (NoSkip)	39.7
Mean Entropy – Midpoint (SkipWrong)	49.5
Mean Entropy – Correct Median (SkipWrong)	48.8
High Entropy – Allow Both	38.1
High Entropy – SkipWrong	40.0
Low Entropy (P90) – NoSkip	36.9
Low Entropy (P90) – SkipWrong	39.4

Table 6: Recovery rate measures the proportion of baseline errors that are corrected after the student receives the teacher explanation.

Entropy Condition	Threshold Value	Threshold Met Rate	Avg. Teacher Passes
Medium Entropy (Mean, Midpoint)	0.2197	0.61	2.28
Low Entropy (Mean, Correct Median)	0.1618	0.19	2.72
High Entropy (Mean, Incorrect Median)	0.2775	0.83	1.84
Low Entropy (P90, Correct Median)	0.6811	0.32	2.88

Table 7: Threshold satisfaction rate and average number of teacher passes for NoSkip experiments.