Learning a Continue-Thinking Token for Enhanced Test-Time Scaling

Liran Ringel[*1] **Elad Tolochinsky**[*1] **Yaniv Romano**[1,2]
[1]Department of Computer Science, Technion – Israel Institute of Technology
[2]Department of Electrical and Computer Engineering, Technion – Israel Institute of Technology
liranringel@cs.technion.ac.il, elad.t@cs.technion.ac.il, yromano@technion.ac.il

Abstract

Test-time scaling has emerged as an effective approach for improving language model performance by utilizing additional compute at inference time. Recent studies have shown that overriding end-of-thinking tokens (e.g., replacing </think> with "Wait") can extend reasoning steps and improve accuracy. In this work, we explore whether a dedicated *continue-thinking* token can be *learned* to trigger extended reasoning. We augment distilled versions of `DeepSeek-R1` with a single learned <|continue-thinking|> token, training only its embedding via reinforcement learning while keeping the model weights frozen. Our experiments show that this learned token achieves improved accuracy on standard math benchmarks compared to both the baseline model and a test-time scaling approach that uses a fixed token (e.g., "Wait") for budget forcing. In particular, we observe that in cases where the fixed-token approach enhances the base model's accuracy, our method achieves a markedly greater improvement. For example, on the GSM8K benchmark, the fixed-token approach yields a 1.3% absolute improvement in accuracy, whereas our learned-token method achieves a 4.2% improvement over the base model that does not use budget forcing.

1 Introduction

Language models have demonstrated impressive reasoning abilities through test-time compute scaling (Snell et al., 2024; Welleck et al., 2024; OpenAI, 2024; Anthropic, 2025). Two dominant paradigms have emerged for this process (Anthropic, 2025): *parallel* and *sequential*. The *parallel* approach involves generating multiple samples and selecting the best response based on a majority vote (Lewkowycz et al., 2022) or a model-provided score (Anthropic, 2025). In contrast, the *sequential* approach—central to this work and popularized by OpenAI's o1 model (OpenAI, 2024)—generates a single sample and encourages the model to revisit, backtrack, validate, and refine its reasoning before producing a final answer, typically resulting in a long chain-of-thought output (Wei et al., 2022). Models that follow this paradigm are commonly referred to as *reasoning models*, and they are often trained using reinforcement learning with verifiable rewards (Lambert et al., 2024; Guo et al., 2025; Su et al., 2025; Wang et al., 2025).

A key property of reasoning models is their ability to decide when to stop thinking, typically by generating an explicit end-of-thinking token (e.g., </think>). However, since this decision is made by the model, users have no direct control over the amount of reasoning performed. Recently, budget forcing was introduced in *s1: simple test-time scaling* (Muennighoff et al., 2025), a sequential test-time scaling approach that provides direct control over the model's computation time. By replacing end-of-thinking tokens with "Wait" tokens during generation, the authors showed that longer chain-of-thought reasoning could be achieved, resulting in improved accuracy. Conversely, early termination can be enforced by appending a </think> token once the model reaches its compute budget, prompting it to generate the final answer. Budget forcing was quickly shown to perform well in various settings (Aggarwal and Welleck, 2025; Huang et al., 2025); see section 3 for a detailed discussion.

In this paper, we introduce a systematic approach for learning a special <|continue-thinking|> token as an alternative to the "Wait" or related fixed tokens such as "Alternatively" or "Hmm" suggested by Muennighoff et al. (2025). Our primary goal is to rigorously investigate whether the simple practice of using a fixed "Wait" token for budget forcing can be improved by learning a dedicated token embedding. In line with this goal, we deliberately constrain ourselves to the same budget

[*]Equal contribution.

Proceedings of the 14th International Joint Conference on Natural Language Processing and the 4th Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics, pages 3324–3345

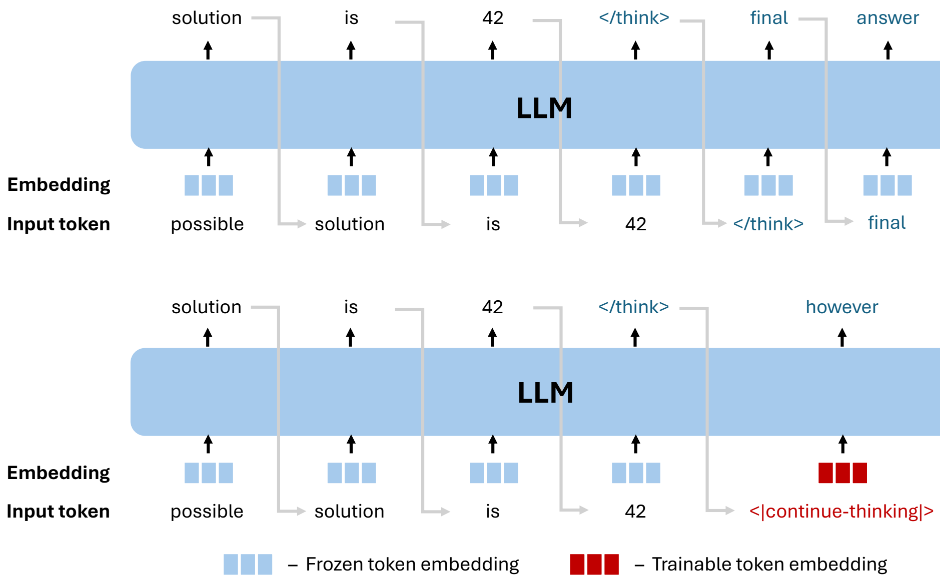

Figure 1: Text generation with budget forcing: Whenever the model outputs a `</think>` token, we replace it with a `<|continue-thinking|>` token and feed that to the model.

forcing algorithm.

As illustrated in Figure 1, we introduce a special, learned `<|continue-thinking|>` token into the model's vocabulary. During generation, we modify the model's output generation process to replace any occurrence of the `</think>` token with our `<|continue-thinking|>` token, as long as the token budget is not exhausted and the maximum number of forced continuations has not been reached. During training, we optimize only the embedding vector of the `<|continue-thinking|>` token while keeping all other model parameters frozen.

We train this new token using reinforcement learning (RL), specifically the group relative policy optimization (GRPO) procedure (Shao et al., 2024). While supervised fine-tuning is a possible alternative, designing effective supervised reasoning demonstrations is difficult. In contrast, RL allows the training process to explore novel continuations aimed at improving task accuracy, making it a more natural fit for our setting. Notably, our method leverages a frozen model backbone, optimizing only a single parameter vector of the size of the model's hidden dimension. As a result, the optimizer's additional memory overhead is limited to just this vector, rather than all model parameters. This allows us to allocate more GPU memory for longer context windows during RL training.

We apply our proposed method to investigate the influence of the learned `<|continue-thinking|>` token on the reasoning process. Our results show that learning this token can significantly enhance model performance, yielding greater gains than the "Wait" or related tokens used in s1's budget-forcing method. Notably, we find that whenever budget-forcing with "Wait" provides an improvement over the baseline, our learned token achieves even greater gains—up to a 320% increase in relative accuracy improvements, and as much as a 4% absolute improvement in overall accuracy. Conversely, in cases where budget-forcing with a fixed token (such as "Wait") does not improve performance compared to the baseline, the learned token similarly does not offer a statistically significant benefit. Results are reported in Table 1.

To summarize, our contributions have the following key features:

- We introduce the concept of learning a specialized `<|continue-thinking|>` token as an effective mechanism for budget-controlled compute scaling.
- Our experiments demonstrate that, in cases where budget-forcing with a fixed token such as "Wait" improves accuracy, learning the token yields even greater gains, while requiring only a single additional token embedding.
- We use an external LLM to compare outputs with ground truth, mitigating the limitations of standard math benchmark evaluations that rely

Dataset	Baseline (w/o BF)	Alternatively	Hmm	Wait	Critique	Learned
DS-R1-Qwen-1.5B						
$C_{\max} = 3$						
AIME24	23.85 ± 0.66	23.49 ± 0.69	24.48 ± 0.63	23.91 ± 0.70	23.12 ± 0.71	24.06 ± 0.68
AIME25	23.33 ± 0.52	23.80 ± 0.56	23.85 ± 0.53	23.49 ± 0.49	22.45 ± 0.49	22.40 ± 0.44
GSM8K	78.08 ± 0.36	78.83 ± 0.34	79.47 ± 0.34	80.14 ± 0.24	76.10 ± 0.59	**83.68** ± 0.38
MATH500	79.88 ± 0.33	80.33 ± 0.35	80.84 ± 0.39	81.22 ± 0.30	79.49 ± 0.32	**82.51** ± 0.34
BBH	45.71 ± 0.60	46.75 ± 0.43	47.06 ± 0.60	47.58 ± 0.52	46.00 ± 0.72	47.70 ± 0.62
DS-R1-Qwen-7B						
$C_{\max} = 3$						
AIME24	46.09 ± 0.80	46.46 ± 0.73	47.24 ± 0.71	47.19 ± 0.77	46.41 ± 0.75	46.82 ± 0.76
AIME25	36.41 ± 0.66	36.51 ± 0.57	36.51 ± 0.61	36.77 ± 0.63	37.29 ± 0.71	36.09 ± 0.58
GSM8K	88.81 ± 0.30	89.27 ± 0.27	91.23 ± 0.26	91.74 ± 0.34	88.89 ± 0.39	**94.91** ± 0.18
MATH500	89.39 ± 0.31	89.84 ± 0.27	91.74 ± 0.19	92.17 ± 0.17	89.06 ± 0.27	**92.89** ± 0.20
BBH	71.05 ± 0.75	72.53 ± 0.72	72.91 ± 0.63	73.85 ± 0.51	71.01 ± 0.76	74.75 ± 0.60
DS-R1-Llama-8B						
$C_{\max} = 3$						
AIME24	35.68 ± 0.78	36.93 ± 0.75	38.23 ± 0.73	38.39 ± 0.79	35.73 ± 0.71	**41.30** ± 0.91
AIME25	29.32 ± 0.61	29.64 ± 0.57	30.10 ± 0.61	30.26 ± 0.64	29.90 ± 0.58	30.26 ± 0.65
GSM8K	78.43 ± 0.40	79.82 ± 0.41	85.22 ± 0.22	84.90 ± 0.43	77.79 ± 0.65	**93.78** ± 0.10
MATH500	79.56 ± 0.31	81.86 ± 0.39	86.21 ± 0.32	86.01 ± 0.32	80.27 ± 0.40	**90.04** ± 0.25
BBH	79.73 ± 0.37	81.32 ± 0.30	81.79 ± 0.46	81.97 ± 0.32	80.26 ± 0.42	**83.69** ± 0.37

Table 1: Accuracy (pass@1) results for $B_{\max} = 9216$ and different numbers of forced thinking continuations $C_{\max}$. Results are obtained via regex-based evaluation and an LLM evaluator if the model fails to generate an answer in the correct format. See Table 5, Table 6, and Table 7 for the full results.

on rigid answer formats, assuming the final answer is in `\boxed{}`. Our experiments show that this rigid-format evaluation approach can misrepresent reasoning ability.

The code used for training and evaluation is available at https://github.com/liranringel/learning-continue-thinking-token.

2 Method

2.1 Single Token Optimization

Let $\pi(x)$ be a pretrained language model that takes a prompt x as input and generates an output. Key to our method is the introduction of a special token, <|continue-thinking|>, which we add to the model's vocabulary to promote longer reasoning traces at test time. We denote the embedding vector of this token by $\theta_T \in \mathbb{R}^d$, where d is the embedding dimension of the model.

We refer to the adapted model with the new token as π_{θ_T}. All model parameters are frozen except for θ_T, so the embedding of the <|continue-thinking|> token is the only parameter updated during training.

Our objective is to maximize the reasoning performance of π_{θ_T} by optimizing the embedding vector θ_T. Formally, we aim to optimize the following:

$$\theta_T^* := \arg\max_{\theta_T} \mathbb{E}_{x \sim Q, o \sim \mathrm{BF}(\pi_{\theta_T}, x)}[R(x, o)], \quad (1)$$

where x is a question sampled from a distribution Q, and o is the response generated by running the budget forcing algorithm on x using the model π_{θ_T}. Note that o is not a standard sample from $\pi_{\theta_T}(y \mid x)$; rather, its distribution is induced by the budget forcing algorithm $\mathrm{BF}(\pi_{\theta_T}, x)$, which we present below. The function $R(x, o)$ represents the reward associated with the generated output. For example, for mathematical questions, $R(x, o)$ could be an indicator function that equals 1 if o contains the correct answer to question x, and 0 otherwise.

In more detail, the budget forcing algorithm $\mathrm{BF}(\pi_{\theta_T}, x)$ modifies the generation process by enforcing additional reasoning steps before producing a final answer. During generation, whenever the model outputs an end-of-thinking </think> token, it is replaced with the learned <|continue-thinking|> token, as a way to force longer reasoning traces. This process repeats until one of the following conditions is met:

1. The number of forced thinking continuations reaches the preset maximum $C_{\max}$.

2. The total number of generated tokens reaches the budget limit B_{max}, in which case the reasoning process is immediately terminated and a </think> token is inserted. This ensures that the model does not generate beyond the allowed compute budget.

During training, we set $C_{\text{max}} = 1$, meaning that only a single forced continuation is allowed per input. At test time, however, we also evaluate the model with $C_{\text{max}} > 1$ to assess its ability to generalize to multiple forced continuations.

2.2 Training

Throughout this paper, we apply our method to distilled versions of DeepSeek-R1, specifically its Qwen 1.5B, Qwen 7B, and Llama 8B variants (Dubey et al., 2024; Yang et al., 2024b; Guo et al., 2025), training only the embedding of a newly introduced token, while keeping all other parameters frozen. We set the number of forced continuations to $C_{\text{max}} = 1$ and the budget limit to $B_{\text{max}} = 8192$ for the 1.5B model and $B_{\text{max}} = 10240$ for larger models. We initialize the new token's embedding using that of the word "Wait." The reward function used during training is the sum of two binary components: (i) a format reward that verifies whether the answer is in the expected format (specifically, wrapped in `\boxed{}`), and (ii) a correctness reward, which checks whether the generated answer matches the ground truth. We implemented our training code on top of Open-R1 (HuggingFace, 2025) and TRL (von Werra et al., 2020). For GRPO, we set $G = 16$ generations per example and used a batch size of 16. We performed 64 gradient accumulation steps, resulting in an effective batch size of 1,024 generations (i.e., 64 examples with 16 generations each) per optimizer update. The training dataset is `DeepScaleR-Preview-Dataset` (Luo et al., 2025), which is a collection of 40,000 math questions compiled from various datasets. We used 8 NVIDIA A100 GPUs with 80GB memory for training the 1.5B model and 8 NVIDIA H200 GPUs for training the larger models. The total training time for each model was about 5 days. See Table 4 for a full list of parameters and system prompts used during training.

3 Related Work

Budget forcing, introduced by Muennighoff et al. (2025), is a simple and effective method for scaling compute at test time. L1 (Aggarwal and Welleck, 2025) extends this idea using reinforcement learning to train models that satisfy user-specified reasoning lengths, enabling flexible cost-performance trade-offs. The m1 method by Huang et al. (2025) further explores the application of budget-forcing in medical QA. Jin et al. (2025) introduce a variant of budget forcing for greedy decoding, comparing multi-word phrases instead of single tokens like "Wait." Collectively, these works highlight budget forcing as a promising direction for compute budget control.

The concept of incorporating additional tokens has been explored in several prior studies. The works reported in (Goyal et al., 2024) and (Wang et al., 2023) introduce learnable tokens into reasoning traces to improve the model's accuracy. In (Goyal et al., 2024; Pfau et al., 2024), additional tokens are inserted at random positions during training and appended to the prompt during inference, allowing the model to artificially increase the number of activations at test time. Wang et al. (2023) included a 'planning token' at the start of each reasoning step. Unlike our approach, these methods rely on supervised fine-tuning to learn token representations. In contrast, our method utilizes RL to optimize the new token embedding.

Finally, various works on scaling test time compute motivate our choice of token learning using RL: (1) Recent empirical findings show that RL-based fine-tuning leads to better generalization (Chu et al., 2025); (2) theoretical analysis shows that RL enjoys higher expected cumulative reward (Setlur et al., 2025), and (3) learning to use additional tokens, which is related to our approach of designing a `<|continue-thinking|>` token, has been observed to be a hard learning problem when using supervised learning (Pfau et al., 2024).

4 Experiments

4.1 Evaluation protocol

We evaluate our model on three widely adopted mathematical reasoning datasets: GSM8K-Platinum (Cobbe et al., 2021; Vendrow et al., 2025), a revised version of the original GSM8K dataset containing 1209 grade-school level math problems; MATH500 (Lightman et al., 2023; Muennighoff et al., 2025), a 500-question subset of the MATH (Hendrycks et al., 2021) dataset; and AIME24 (Muennighoff et al., 2025) and AIME25 datasets, which contain 30 math problems from the

Dataset	Baseline (w/o BF)	Alternatively	Hmm	Wait	Critique	Learned
`DS-R1-Qwen-1.5B`						
$C_{\max} = 2$						
AIME25	23.18 ± 0.50 (86)	23.02 ± 0.49 (85)	23.23 ± 0.51 (86)	23.23 ± 0.50 (87)	22.14 ± 0.49 (85)	22.80 ± 0.47 (86)
GSM8K	66.03 ± 0.40 (84)	64.82 ± 0.25 (82)	64.09 ± 0.38 (80)	64.43 ± 0.48 (79)	69.99 ± 0.44 (88)	**78.27** ± 0.58 (94)
MATH500	77.56 ± 0.32 (95)	77.97 ± 0.35 (95)	78.88 ± 0.32 (95)	78.83 ± 0.38 (95)	77.56 ± 0.28 (95)	**81.37** ± 0.38 (97)
$C_{\max} = 3$						
AIME25	23.18 ± 0.50 (85)	23.44 ± 0.56 (87)	23.49 ± 0.49 (86)	23.12 ± 0.46 (86)	22.14 ± 0.46 (86)	22.40 ± 0.44 (86)
GSM8K	66.03 ± 0.40 (83)	63.79 ± 0.57 (80)	64.27 ± 0.28 (79)	62.94 ± 0.50 (77)	69.35 ± 0.56 (87)	**80.87** ± 0.42 (96)
MATH500	77.56 ± 0.32 (94)	78.16 ± 0.39 (95)	78.60 ± 0.40 (94)	78.75 ± 0.32 (94)	78.08 ± 0.28 (96)	**81.75** ± 0.34 (96)
`DS-R1-Qwen-7B`						
$C_{\max} = 2$						
AIME24	44.74 ± 0.77 (71)	45.47 ± 0.78 (71)	45.94 ± 0.78 (72)	45.62 ± 0.79 (72)	45.42 ± 0.77 (71)	45.52 ± 0.80 (72)
AIME25	35.47 ± 0.62 (67)	35.68 ± 0.59 (68)	35.00 ± 0.62 (66)	35.62 ± 0.66 (67)	36.82 ± 0.67 (67)	34.90 ± 0.60 (68)
GSM8K	88.81 ± 0.30 (99)	88.82 ± 0.39 (99)	90.58 ± 0.29 (99)	90.85 ± 0.39 (99)	89.12 ± 0.14 (99)	**94.64** ± 0.19 (99)
MATH500	88.94 ± 0.30 (97)	88.90 ± 0.24 (97)	91.08 ± 0.19 (97)	91.24 ± 0.20 (97)	88.74 ± 0.26 (97)	**92.65** ± 0.18 (97)
$C_{\max} = 3$						
AIME24	44.74 ± 0.77 (71)	45.52 ± 0.72 (72)	46.15 ± 0.72 (71)	46.30 ± 0.74 (73)	45.21 ± 0.80 (73)	45.94 ± 0.79 (72)
AIME25	35.47 ± 0.62 (67)	35.47 ± 0.57 (66)	35.52 ± 0.62 (67)	35.73 ± 0.64 (66)	36.41 ± 0.68 (67)	35.16 ± 0.58 (66)
GSM8K	88.81 ± 0.30 (99)	89.15 ± 0.25 (99)	90.45 ± 0.24 (98)	91.51 ± 0.35 (99)	88.74 ± 0.38 (99)	**94.84** ± 0.18 (99)
MATH500	88.94 ± 0.30 (97)	89.42 ± 0.27 (97)	91.20 ± 0.16 (97)	91.78 ± 0.18 (97)	88.70 ± 0.26 (97)	**92.62** ± 0.20 (97)

Table 2: Accuracy (pass@1) results for $B_{\max} = 9216$ and different numbers of forced thinking continuations $C_{\max}$. Results are obtained via a regex-based evaluation only. The percentage of final answers enclosed in `\boxed{}` is shown in parentheses. See Table 8 and Table 9 for the full results.

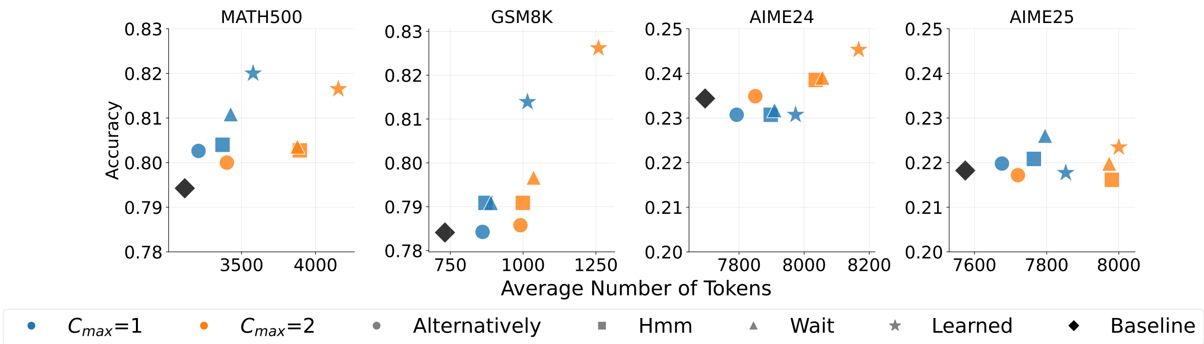

Figure 2: Accuracy of different methods as a function of the average number of tokens generated by each method. Results for all datasets are obtained using `DeepSeek-R1-Distill-Qwen-1.5B` and $B_{\max} = 8192$.

2024 and 2025 American Invitational Mathematics Examination, a national-level mathematics competition in the United States. We implemented the evaluation pipeline using a modified version of the LM-Evaluation-Harness library (Gao et al., 2024).

To further assess the generalization of our approach beyond mathematical reasoning, we evaluated all models on a subset of 16 tasks from BIG-Bench Hard (BBH) (Suzgun et al., 2022), a benchmark comprising challenging problems that probe symbolic, logical, and commonsense reasoning abilities. To ensure reliable automatic evaluation, we restricted BBH to questions with multiple-choice answers. Importantly, all models were trained exclusively on math questions, so performance on BBH provides a measure of cross-domain generalization.

Our method is compared against three fixed tokens that have previously demonstrated strong performance (Muennighoff et al., 2025), a baseline configuration that does not employ budget-forcing, and an explicit guiding prompt: "Critique your previous step and try again".

Due to the relatively small size of some of the datasets, we generate multiple responses per question to compute standard error as a way to enhance the statistical reliability of the reported results. Specifically, we found that generating the following number of completions produces error bars that allow us to distinguish between statistically significant and insignificant results: 16 samples per question for MATH500, 64 for AIME, 6 samples for GSM8K, and 10 samples for BBH. We report (pass@1) accuracy along with the standard error for each setting. Accuracy is computed using a regex-based evaluation script that extracts the final answer from the model's output and checks for an exact match with the ground truth. The evaluation returns 'True' if the extracted answer is correct, and 'False' otherwise—including cases where the regex fails to match a valid answer. In case the regex fails to match, we employ an external LLM to assess if the generated answer was semantically equivalent to the ground truth. This hybrid evaluation strategy was adopted based on our manual inspection, which indicated that regex matching is more reliable than LLM-based comparison when the regex successfully parses the output; see subsection 4.2 for a detailed discussion. We utilized `Qwen/Qwen2.5-7B-Instruct` (Yang et al., 2024a) as our evaluator LLM. See Table 4 for the instruction prompt we used for the LLM evaluation.

During inference, we verify the generalization of our method by also using inference configurations not used during training. Concretely, we set the reasoning budgets $B_{\text{max}} = 8192, 9216$ and the maximal number of forced continuations $C_{\text{max}} = 1, 2, 3$. We also provide an additional 1,024 tokens reserved for generating the final answer. For inference, we used the same system prompt as the one used for training; see Table 4 in the Appendix. Due to the prolonged length of the generated answers, evaluation took approximately 370 GPU hours, using 8 NVIDIA A40 GPUs.

4.2 LLM-Based Verification

It is common practice to use regex-based functions to check the correctness of model outputs when evaluating language models on mathematical benchmarks. However, this approach can distort the evaluation of model performance, as it conflates output formatting with actual reasoning ability. In our experiments, evaluating with only regex-based functions suggested that the learned token led to substantial performance gains. However, a more detailed analysis using an evaluator LLM revealed that much of this improvement was due to better adherence to formatting rather than genuine reasoning improvements. As shown in Table 2, the results obtained using only regex-based evaluation are significantly higher than those reported in Table 1, which use LLM-based evaluation. For example, on GSM8K with $C_{\text{max}} = 2$ and $B_{\text{max}} = 8192$, the improvement of our method over the baseline is more than three times higher when using regex-based evaluation compared to LLM-based evaluation. Similarly, on the MATH500 dataset with $C_{\text{max}} = 1$ and $B_{\text{max}} = 9216$, regex-based evaluation indicates a statistically significant improvement, whereas LLM-based evaluation shows that this improvement is not actually present.

To further assess the trustworthiness of our evaluation procedure, we manually inspected answers generated by `DS-R1-Qwen-7B` along with their corresponding LLM-based evaluation results for 80 randomly selected questions from the MATH500 dataset and found that the LLM evaluation aligned with human assessment in all cases.

Nonetheless, we acknowledge that LLM-based verification is itself imperfect. We adopt it as a more reliable alternative to purely regex-based evaluation, improving the robustness and credibility of our reported results.

4.3 Results

The evaluation results are depicted in Table 1. Our findings indicate that when budget forcing does not improve upon the baseline, the learned token similarly offers no statistically significant advantage. However, on datasets where budget forcing yields improvements, our learned token demonstrates a substantial performance increase, achieving up to a 320% relative gain over the best fixed token and a 4% absolute improvement in accuracy. Notably, although our model was trained with $C_{\text{max}} = 1$, we observe that increasing C_{max} to 2 and 3 during inference often leads to further improvements in accuracy. This suggests that the learned `<|continue-thinking|>` token can generalize to settings with multiple forced continuations, even though the model was not explicitly trained for them.

To gain deeper insights into the behavior of the different methods, we analyzed the generated token distributions. Figure 2 depicts the accuracy of each method as a function of the average number of tokens generated per dataset. We observed that using the learned token consistently resulted in longer reasoning traces, suggesting that the performance improvement can be attributed to this increased reasoning length. Furthermore, Figure 2 shows that for the AIME datasets, all methods generated a high average number of tokens. This can likely be attributed to the significant difficulty of the AIME problems for our model, which might also explain why our method did not yield substantial improvements on these datasets.

Indeed, Wu (2025) has also noted that budget forcing provides inconsistent or minimal benefits on AIME benchmarks. Since AIME consists of

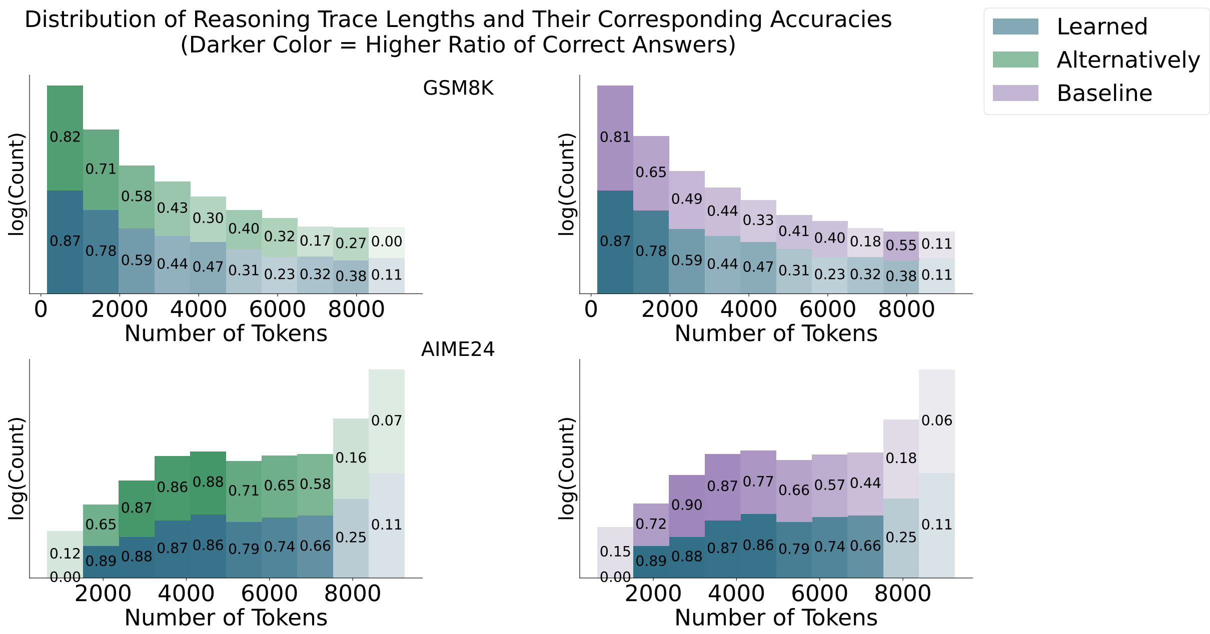

Figure 3: Comparison of generated sequence length distributions and their corresponding accuracies. Stacked bars represent the logarithmic count of answers within each length bin, with darker segments indicating a higher proportion of correct answers (fraction shown within each bin). Top row: GSM8K, Bottom row: AIME24. Left: Learned `<|continue-thinking|>` token vs. "Alternatively." Right: Learned `<|continue-thinking|>` token vs. baseline model without budget forcing. Data was obtained using `DS-R1-Qwen-1.5B` and $C_{\max} = 2$, $B_{\max} = 8192$.

only 30 challenging questions, each additional correct answer significantly increases the overall score (approximately +3.3% per question). Therefore, the moderate improvements provided by budget forcing may be insufficient to consistently reach the threshold required to solve the next level of question difficulty, which might explain the limited effectiveness observed both here and in prior studies.

A comparison of the generated answer length distributions for the learned token, the baseline, and the "Alternatively" token is shown in Figure 3. The accuracy improvements observed with the learned token are consistent across different generated lengths, not just on average. This suggests that the enhanced performance is attributable to a genuine improvement in the model's reasoning capabilities, beyond merely generating longer responses. In Table 3, we show the probabilities that a correct answer under the baseline will be incorrect with the learned token. We can see that, for both GSM8K and MATH500, it is much more likely that the learned token leads to improvement. For the AIME dataset, we see that the learned token and the baseline are comparable, which is expected, since both methods have similar accuracy.

To better understand how the learned token

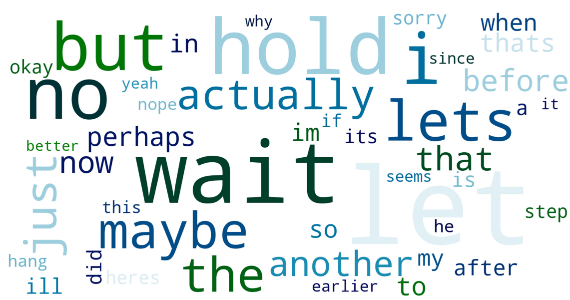

Figure 4: Word cloud of the first token generated immediately after injecting the learned `<|continue-thinking|>` token, across all datasets from `DS-R1-Qwen-1.5B`.

influences the model's reasoning process, we visualize in Figure 4 a word cloud showing the first word generated immediately after the `<|continue-thinking|>` token. The most common continuations often prompt the model to self-verify or reconsider its previous steps, indicating that the token effectively encourages reflective reasoning and backtracking.

The reasoning trace depicted in Figure 5, taken from the GSM8K dataset, showcases how the `<|continue_thinking|>` token influences the final answer. The blue portion highlights the model's initial reasoning, which leads to an in-

Probabilities	AIME24	AIME25	GSM8K	MATH500
$C_{\max} = 2$, $B_{\max} = 8192$				
P(Learned ✗, Baseline ✓)	0.01	0.01	0.03	0.02
P(Learned ✓, Baseline ✗)	0.02	0.02	0.07	0.04
$C_{\max} = 3$, $B_{\max} = 8192$				
P(Learned ✗, Baseline ✓)	0.09	0.06	0.02	0.01
P(Learned ✓, Baseline ✗)	0.09	0.06	0.07	0.04
$C_{\max} = 2$, $B_{\max} = 9216$				
P(Learned ✗, Baseline ✓)	0.08	0.06	0.02	0.02
P(Learned ✓, Baseline ✗)	0.08	0.06	0.07	0.04
$C_{\max} = 3$, $B_{\max} = 9216$				
P(Learned ✗, Baseline ✓)	0.09	0.06	0.02	0.02
P(Learned ✓, Baseline ✗)	0.09	0.06	0.07	0.04

Table 3: Probability of observing a correct answer from the learned token and an incorrect answer from the baseline and vice versa when using `DeepSeek-R1-Distill-Qwen-1.5B`. See Table 10 in the Appendix for the full results.

Question: Lloyd has an egg farm. His chickens produce 252 eggs per day and he sells them for $2 per dozen. How much does Lloyd make on eggs per week?

`<think> First, I need to calculate the daily revenue from egg sales. [...] Finally, to determine the weekly revenue, I'll multiply the daily revenue by the number of days in a week, which is 7. So, $1,134 multiplied by 7 equals $7,938.`

`<|continue_thinking|> no, let me try that again. [...] Therefore, the correct answer is $294 per week.`

`</think> Lloyd sells his eggs at $2 per dozen. To determine his weekly revenue:`

1. **Calculate weekly egg production**: - 252 eggs/day * 7 days = 1,764 eggs/week

[...]

Answer: Lloyd makes \boxed{294} dollars per week from his egg sales.

Figure 5: GSM8K reasoning trace from `DS-R1-Qwen-1.5B` demonstrating the positive impact of `<|continue_thinking|>` token. Blue indicates the original reasoning, yielding an incorrect answer of 7,938. Green shows the continuation after the special token was added, leading to the correct answer of 294.

correct result. However, the introduction of the `<|continue_thinking|>` token prompts a re-examination of the solution, guiding the model to the correct conclusion. See Appendix D for the full reasoning traces and additional examples.

5 Conclusions

In this work, we have demonstrated that learning a dedicated `<|continue-thinking|>` token yields significant effectiveness, specifically in scenarios where the baseline budget forcing method already provides performance improvements. We introduced a training methodology for this token that exhibits promising generalization capabilities across different inference settings. Our analysis indicates that the observed performance gains are not primarily due to better adherence to output formatting, but rather stem from the elicited longer reasoning traces and a genuine enhancement in the model's underlying reasoning capabilities. Furthermore, this improvement remains relevant on average and conditionally across varying reasoning lengths, suggesting its utility even when generating shorter completions. While our method of learning a specialized `<|continue-thinking|>` token is relatively simple, practitioners can readily ascertain its potential benefit for their specific scenario by first employing the vanilla budget-forcing technique with a fixed token, such as "Wait"; an observed performance increase with this baseline strongly suggests that training a dedicated `<|continue-thinking|>` token would be worthwhile. Finally, we emphasize the critical importance of rigorous evaluation for drawing meaningful conclusions and propose a refined evaluation scheme designed to mitigate some of the inherent limitations associated with relying solely on regex-based assessments.

Future Directions One promising future direction involves exploring the efficacy of learning distinct `<|continue-thinking|>` tokens tailored to different positions within the generated sequence or investigating the benefits of learning these tokens jointly. For example, one can add a special token that will be used for the first forced continuation and a second new token that will be used for the second forced continuation. More broadly, an exciting avenue is to train specialized tokens to enable additional forms of controllable reasoning, such as controlling answer length or enabling early termination. Given our focus on the sequential approach to test-time scaling, another compelling future direction would be to explore the integration of our learned token methodology with parallel test-time scaling paradigms. Finally, extending the scope of this research to diverse domains, potentially even those lacking explicit verifiable rewards, such as in the context of LLM alignment, presents an intriguing area for future exploration. We believe the code provided with this paper will enable researchers to pursue these research directions efficiently.

6 Limitations

While our proposed method demonstrates promising results, it is subject to several limitations. First, as our analysis indicates, the effectiveness of the learned token appears to be contingent on the baseline performance of the budget forcing technique itself. If standard budget forcing does not yield improvements, our learned <|continue-thinking|> token is unlikely to provide a significant advantage. Second, the process of learning the token embedding necessitates a training phase, which is inherently more computationally demanding and requires access to the model's weights compared to simply employing fixed tokens. Finally, our method requires the addition of a new token to the model's vocabulary. This modification might not be feasible or permitted when utilizing LLMs through certain API interfaces, which often provide restricted access to the model's architecture, thus preventing vocabulary modifications.

7 Acknowledgments

This research was supported by the European Union (ERC, SafetyBounds, 101163414). Views and opinions expressed are however those of the authors only and do not necessarily reflect those of the European Union or the European Research Council Executive Agency. Neither the European Union nor the granting authority can be held responsible for them. This research was also partially supported by the Israel Science Foundation (ISF grant 729/21). Y. R. acknowledges additional support from the Career Advancement Fellowship at the Technion.

References

Pranjal Aggarwal and Sean Welleck. 2025. L1: Controlling how long a reasoning model thinks with reinforcement learning. *arXiv preprint arXiv:2503.04697*.

Anthropic. 2025. Claude's extended thinking. https://www.anthropic.com/news/visible-extended-thinking. Accessed: 2025-05-18.

Tianzhe Chu, Yuexiang Zhai, Jihan Yang, Shengbang Tong, Saining Xie, Dale Schuurmans, Quoc V Le, Sergey Levine, and Yi Ma. 2025. SFT memorizes, RL generalizes: A comparative study of foundation model post-training. *arXiv preprint arXiv:2501.17161*.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, and 1 others. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, and 1 others. 2024. The llama 3 herd of models. *arXiv e-prints*, pages arXiv–2407.

Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, and 5 others. 2024. The language model evaluation harness.

Sachin Goyal, Ziwei Ji, Ankit Singh Rawat, Aditya Krishna Menon, Sanjiv Kumar, and Vaishnavh Nagarajan. 2024. Think before you speak: Training language models with pause tokens. In *International Conference on Learning Representations*.

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*.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. Measuring mathematical problem solving with the MATH dataset. In *Neural Information Processing Systems, Datasets and Benchmarks Track*.

Xiaoke Huang, Juncheng Wu, Hui Liu, Xianfeng Tang, and Yuyin Zhou. 2025. m1: Unleash the potential of test-time scaling for medical reasoning with large language models. *arXiv preprint arXiv:2504.00869*.

HuggingFace. 2025. Open R1: A fully open reproduction of DeepSeek-R1.

Hyunbin Jin, Je Won Yeom, Seunghyun Bae, and Taesup Kim. 2025. “well, keep thinking”: Enhancing llm reasoning with adaptive injection decoding. *arXiv preprint arXiv:2503.10167*.

Nathan Lambert, Jacob Morrison, Valentina Pyatkin, Shengyi Huang, Hamish Ivison, Faeze Brahman, Lester James V. Miranda, Alisa Liu, Nouha Dziri, Shane Lyu, Nora Kassner, Timo Schick, Marzieh Saeidi, Noah A. Smith, and Matt Gardner. 2024. Tülu 3: Pushing frontiers in open language model post-training. *arXiv preprint arXiv:2411.15124*.

Aitor Lewkowycz, Anders Johan Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay Venkatesh Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, Yuhuai Wu, Behnam Neyshabur, Guy Gur-Ari, and Vedant Misra. 2022. Solving quantitative reasoning problems with language models. In *Advances in Neural Information Processing Systems*.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2023. Let’s verify step by step. In *International Conference on Learning Representations*.

Michael Luo, Sijun Tan, Justin Wong, Xiaoxiang Shi, William Y Tang, Manan Roongta, Colin Cai, Jeffrey Luo, Tianjun Zhang, and Li Erran Li. 2025. Deepscaler: Surpassing o1-preview with a 1.5B model by scaling RL. https://www.notion.so/Deepscaler-Surpassing-o1-preview-with-a-1-5B-model-by-scaling-RL-<exact-url>. Notion Blog.

Niklas Muennighoff, Zitong Yang, Weijia Shi, Xiang Lisa Li, Li Fei-Fei, Hannaneh Hajishirzi, Luke Zettlemoyer, Percy Liang, Emmanuel Candès, and Tatsunori Hashimoto. 2025. s1: Simple test-time scaling. *arXiv preprint arXiv:2501.19393*.

OpenAI. 2024. Learning to reason with llms. https://openai.com/index/learning-to-reason-with-llms/. Accessed: 2025-05-18.

Jacob Pfau, William Merrill, and Samuel R Bowman. 2024. Let’s think dot by dot: Hidden computation in transformer language models. *arXiv preprint arXiv:2404.15758*.

Amrith Setlur, Nived Rajaraman, Sergey Levine, and Aviral Kumar. 2025. Scaling test-time compute without verification or RL is suboptimal. *arXiv preprint arXiv:2502.12118*.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Y Wu, and 1 others. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*.

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

Yi Su, Dian Yu, Linfeng Song, Juntao Li, Haitao Mi, Zhaopeng Tu, Min Zhang, and Dong Yu. 2025. Expanding RL with verifiable rewards across diverse domains. *arXiv preprint arXiv:2503.23829*.

Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc V Le, Ed H Chi, Denny Zhou, , and Jason Wei. 2022. Challenging big-bench tasks and whether chain-of-thought can solve them. *arXiv preprint arXiv:2210.09261*.

Joshua Vendrow, Edward Vendrow, Sara Beery, and Aleksander Madry. 2025. Do large language model benchmarks test reliability? *arXiv preprint arXiv:2502.03461*.

Leandro von Werra, Younes Belkada, Lewis Tunstall, Edward Beeching, Tristan Thrush, Nathan Lambert, Shengyi Huang, Kashif Rasul, and Quentin Gallouédec. 2020. Trl: Transformer reinforcement learning. https://github.com/huggingface/trl. GitHub repository.

Xinyi Wang, Lucas Caccia, Oleksiy Ostapenko, Xingdi Yuan, William Yang Wang, and Alessandro Sordoni. 2023. Guiding language model reasoning with planning tokens. *arXiv preprint arXiv:2310.05707*.

Yiping Wang, Qing Yang, Zhiyuan Zeng, Liliang Ren, Lucas Liu, Baolin Peng, Hao Cheng, Xuehai He, Kuan Wang, Jianfeng Gao, and 1 others. 2025. Reinforcement learning for reasoning in large language models with one training example. *arXiv preprint arXiv:2504.20571*.

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*.

Sean Welleck, Amanda Bertsch, Matthew Finlayson, Hailey Schoelkopf, Alex Xie, Graham Neubig, Ilia Kulikov, and Zaid Harchaoui. 2024. From decoding to meta-generation: Inference-time algorithms for large language models. *arXiv preprint arXiv:2406.16838*.

Guojun Wu. 2025. It’s not that simple. an analysis of simple test-time scaling. *arXiv preprint arXiv:2507.14419*.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, and 23 others. 2024a. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*.

An Yang, Beichen Zhang, Binyuan Hui, Bofei Gao, Bowen Yu, Chengpeng Li, Dayiheng Liu, Jianhong Tu, Jingren Zhou, Junyang Lin, Keming Lu, Mingfeng Xue, Xingzhang Ren, and Zhenru Zhang. 2024b. Qwen2.5-math technical report: Toward mathematical expert model via self-improvement. *arXiv preprint arXiv:2409.12122*.

A Training and Evaluation Parameters

Table 4 summarizes the full list of parameters we used for training and evaluation.

GRPO Parameters	
G	16
ε	0.2
β	0
μ	4
Training Parameters	
Optimizer	AdamW
Learning rate	$1e-4$ with cosine scheduler
Temperature	0.9
Training and Inference Prompt	You are a helpful AI Assistant. First, think through the reasoning inside <think>...</think>. Then, always present the final answer in \boxed{}
Evaluation Parameters	
Temperature	0.9
LLM evaluator prompt	Given a math question, a correct answer, and a student's final answer (which may include explanations), determine if the correct answer appears in some form within the student's answer (ignoring trivial differences like formatting or wording). Output True if the student's answer is correct, otherwise output False. Output nothing else. START-OF-QUESTION question: %(question)s END-OF-QUESTION correct answer: %(correct_answer)s student's solution: START-OF-STUDENT-SOLUTION %(student_solution)s END-OF-STUDENT-SOLUTION Output True if the student's solution equivalent the correct answer and False otherwise.

Table 4: Full list of training and evaluation parameters

B Artifacts Used

The following datasets, software libraries and models were used during this research, all artifacts were used in accordance with their respective licenses.

- **Datasets**: `DeepScaleR-Preview` Dataset, licensed under the MIT license.[1] `GSM8K-Platinum`, licensed under CC BY-SA 4.0.[2] `AIME24`, licensed under Apache-2.0,[3] and AIME25.[4]
- **Models**: `DeepSeek-R1-Distill-Qwen-1.5B`, `DeepSeek-R1-Distill-Qwen-7B` and `DeepSeek-R1-Distill-Llama-8B` licensed under the MIT license.[5][6][7] `Qwen2.5-7B-Instruct`, licensed under the Apache-2.0 license.[8]
- **Software Packages**: LM-evaluation-harness licensed under the MIT license.[9] vLLM licensed under the Apache-2.0.[10] Open r1, licensed under the Apache 2.0 license,[11] and trl, licensed under Apache 2.0.[12]

C Full Experimental Results

We provide the complete set of results for all our experiments. Table 5, Table 6, and Table 7 show the complete set of results when using an LLM evaluator in all configurations ($C_{\text{max}} = 1, 2, 3$, $B_{\text{max}} = 8192, 9216$). Table 8 and Table 9 show the complete set of results for regex-only evaluation. Table 10 reports, for each configuration, the probability that the learned token yields a correct answer when the baseline does not, and the probability that the baseline yields a correct answer when the learned token does not.

D Generated Answers Examples

We include examples of reasoning traces of cases when using the learned token resulted in a correct answer while the baseline did not, and vice versa. In all figures, blue text indicates the original reasoning trace that is common for both the learned model and the baseline model and green text indicates the reasoning trace that was generated after

[1] https://hf.co/datasets/agentica-org/DeepScaleR-Preview-Dataset
[2] https://hf.co/datasets/madrylab/gsm8k-platinum
[3] https://hf.co/datasets/simplescaling/aime24_nofigures
[4] https://hf.co/datasets/math-ai/aime25
[5] https://hf.co/deepseek-ai/DeepSeek-R1-Distill-Qwen-1.5B
[6] https://hf.co/deepseek-ai/DeepSeek-R1-Distill-Qwen-7B
[7] https://hf.co/deepseek-ai/DeepSeek-R1-Distill-Llama-8B
[8] https://hf.co/Qwen/Qwen2.5-7B-Instruct
[9] https://github.com/EleutherAI/lm-evaluation-harness
[10] https://github.com/vllm-project/vllm
[11] https://github.com/huggingface/open-r1
[12] https://github.com/huggingface/trl

Dataset	Baseline (w/o BF)	Alternatively	Hmm	Wait	Critique	Learned
$C_{\max}=1,\ B_{\max}=8192$						
AIME24	23.44 ± 0.73	23.07 ± 0.72	23.07 ± 0.70	23.18 ± 0.69	23.33 ± 0.51	23.07 ± 0.71
AIME25	21.82 ± 0.67	21.98 ± 0.56	22.08 ± 0.53	22.6 ± 0.51	22.76 ± 0.65	21.77 ± 0.55
GSM8K	78.41 ± 0.25	78.43 ± 0.30	79.09 ± 0.42	79.09 ± 0.54	76.76 ± 0.24	**81.39**± 0.40
MATH500	79.43 ± 0.26	80.26 ± 0.36	80.40 ± 0.29	81.09 ± 0.32	80.35 ± 0.36	**82.00** ± 0.29
$C_{\max}=2,\ B_{\max}=8192$						
AIME24	23.44 ± 0.73	23.54 ± 0.82	23.85 ± 0.76	24.06 ± 0.76	23.28 ± 0.62	24.53 ± 0.75
AIME25	21.88 ± 0.67	21.72 ± 0.66	21.61 ± 0.67	21.98 ± 0.68	22.24 ± 0.63	22.34 ± 0.71
GSM8K	78.41 ± 0.25	78.58 ± 0.34	79.09 ± 0.28	79.71 ± 0.31	76.80 ± 0.46	**82.63** ± 0.20
MATH500	79.43 ± 0.26	80.00 ± 0.19	80.28 ± 0.29	80.36 ± 0.27	79.62 ± 0.31	**81.67** ± 0.21
$C_{\max}=3,\ B_{\max}=8192$						
AIME24	23.44 ± 0.73	23.02 ± 0.58	23.12 ± 0.62	23.49 ± 0.54	23.49 ± 0.75	23.33 ± 0.73
AIME25	21.82 ± 0.67	22.08 ± 0.65	21.67 ± 0.68	22.03 ± 0.61	22.86 ± 0.60	22.03 ± 0.60
GSM8K	78.41 ± 0.25	79.00 ± 0.13	79.65 ± 0.26	80.16 ± 0.32	76.58 ± 0.16	**83.17** ± 0.32
MATH500	79.42 ± 0.26	79.60 ± 0.32	80.71 ± 0.30	80.86 ± 0.18	80.24 ± 0.29	**82.29** ± 0.24
$C_{\max}=1,\ B_{\max}=9216$						
AIME24	23.85 ± 0.66	23.28 ± 0.63	24.17 ± 0.63	23.91 ± 0.62	23.59 ± 0.66	24.01 ± 0.61
AIME25	23.33 ± 0.52	23.33 ± 0.55	23.80 ± 0.60	23.70 ± 0.56	22.24 ± 0.55	23.28 ± 0.53
GSM8K	78.08 ± 0.36	78.83 ± 0.35	78.91 ± 0.33	79.11 ± 0.42	76.28 ± 0.43	**81.67** ± 0.47
MATH500	79.88 ± 0.33	80.09 ± 0.33	80.69 ± 0.32	80.82 ± 0.30	80.29 ± 0.24	81.46 ± 0.39
BBH	45.71 ± 0.60	46.24 ± 0.60	46.53 ± 0.64	47.79 ± 0.70	45.91 ± 0.77	47.07 ± 0.66
$C_{\max}=2,\ B_{\max}=9216$						
AIME24	23.85 ± 0.66	23.59 ± 0.68	24.37 ± 0.68	24.06 ± 0.75	22.86 ± 0.75	24.06 ± 0.69
AIME25	23.33 ± 0.52	23.39 ± 0.51	23.75 ± 0.53	23.54 ± 0.51	22.55 ± 0.52	23.07 ± 0.49
GSM8K	78.08 ± 0.36	78.96 ± 0.21	79.67 ± 0.35	79.86 ± 0.39	76.92 ± 0.45	**82.53** ± 0.48
MATH500	79.88 ± 0.33	80.14 ± 0.38	81.19 ± 0.31	81.21 ± 0.28	79.14 ± 0.29	**82.30** ± 0.34
BBH	45.71 ± 0.60	46.58 ± 0.50	46.78 ± 0.76	47.36 ± 0.62	45.97 ± 0.62	47.39 ± 0.57
$C_{\max}=3,\ B_{\max}=9216$						
AIME24	23.85 ± 0.66	23.49 ± 0.69	24.48 ± 0.63	23.91 ± 0.70	23.12 ± 0.71	24.06 ± 0.68
AIME25	23.33 ± 0.52	23.80 ± 0.56	23.85 ± 0.53	23.49 ± 0.49	22.45 ± 0.49	22.40 ± 0.44
GSM8K	78.08 ± 0.36	78.83 ± 0.34	79.47 ± 0.34	80.14 ± 0.24	76.10 ± 0.59	**83.68** ± 0.38
MATH500	79.88 ± 0.33	80.33 ± 0.35	80.84 ± 0.39	81.22 ± 0.30	79.49 ± 0.32	**82.51** ± 0.34
BBH	45.71 ± 0.60	46.75 ± 0.43	47.06 ± 0.60	47.58 ± 0.52	46.00 ± 0.72	47.70 ± 0.62

Table 5: Accuracy (pass@1) results for `DeepSeek-R1-Distill-Qwen-1.5B` with different token budget limits $B_{\max}$ and different numbers of forced thinking continuations $C_{\max}$. Results are obtained via regex-based evaluation and an LLM evaluator if the model fails to generate an answer in the correct format.

a forced continuation. Figure 6 and Figure 7 show full reasoning traces for the learned model and the baseline model respectively on a question in which the baseline model was incorrect and adding the learned token allowed the model to continue reasoning through the problem and reach the correct answer. Figure 8 and Figure 9 show full reasoning traces for the learned model and the baseline model, respectively, on a question in which both models were correct. Note that in this case, the learned token only adds a negligible number of tokens to the answer.

Dataset	Baseline (w/o BF)	Alternatively	Hmm	Wait	Critique	Learned
$C_{\max}=1$, $B_{\max}=8192$						
AIME24	44.90 $\pm$ 0.75	45.16 $\pm$ 0.79	45.10 $\pm$ 0.75	45.57 $\pm$ 0.77	46.00 $\pm$ 0.82	46.20 $\pm$ 0.82
AIME25	35.83 $\pm$ 0.57	35.78 $\pm$ 0.60	35.36 $\pm$ 0.58	35.78 $\pm$ 0.66	35.83 $\pm$ 0.57	35.89 $\pm$ 0.59
GSM8K	88.60 $\pm$ 0.35	88.50 $\pm$ 0.31	90.56 $\pm$ 0.20	89.74 $\pm$ 0.12	89.23 $\pm$ 0.28	**94.60** $\pm$ 0.22
MATH500	89.39 $\pm$ 0.26	89.09 $\pm$ 0.29	90.55 $\pm$ 0.25	90.34 $\pm$ 0.18	89.20 $\pm$ 0.27	**92.64** $\pm$ 0.24
$C_{\max}=2$, $B_{\max}=8192$						
AIME24	44.90 $\pm$ 0.75	45.00 $\pm$ 0.76	45.68 $\pm$ 0.79	45.78 $\pm$ 0.82	46.20 $\pm$ 0.84	45.63 $\pm$ 0.84
AIME25	35.83 $\pm$ 0.57	36.04 $\pm$ 0.64	35.99 $\pm$ 0.67	35.68 $\pm$ 0.67	35.00 $\pm$ 0.54	35.94 $\pm$ 0.60
GSM8K	88.60 $\pm$ 0.35	89.16 $\pm$ 0.17	91.33 $\pm$ 0.21	90.89 $\pm$ 0.15	88.68 $\pm$ 0.23	**95.00** $\pm$ 0.16
MATH500	89.39 $\pm$ 0.26	89.18 $\pm$ 0.26	91.16 $\pm$ 0.20	91.19 $\pm$ 0.21	89.14 $\pm$ 0.28	**92.67** $\pm$ 0.24
$C_{\max}=3$, $B_{\max}=8192$						
AIME24	44.90 $\pm$ 0.75	45.26 $\pm$ 0.82	45.36 $\pm$ 0.86	46.46 $\pm$ 0.79	45.78 $\pm$ 0.78	45.63 $\pm$ 0.80
AIME25	35.83 $\pm$ 0.57	35.83 $\pm$ 0.54	36.04 $\pm$ 0.63	36.35 $\pm$ 0.57	35.73 $\pm$ 0.63	35.99 $\pm$ 0.56
GSM8K	88.60 $\pm$ 0.35	89.15 $\pm$ 0.22	91.85 $\pm$ 0.24	91.43 $\pm$ 0.32	88.72 $\pm$ 0.25	**95.18** $\pm$ 0.17
MATH500	89.39 $\pm$ 0.26	89.26 $\pm$ 0.26	91.78 $\pm$ 0.28	91.75 $\pm$ 0.18	88.85 $\pm$ 0.26	**92.64** $\pm$ 0.22
$C_{\max}=1$, $B_{\max}=9216$						
AIME24	46.09 $\pm$ 0.80	46.77 $\pm$ 0.77	46.61 $\pm$ 0.77	47.08 $\pm$ 0.77	46.61 $\pm$ 0.77	46.25 $\pm$ 0.80
AIME25	36.41 $\pm$ 0.66	35.78 $\pm$ 0.58	36.15 $\pm$ 0.55	36.35 $\pm$ 0.64	34.74 $\pm$ 0.62	36.46 $\pm$ 0.63
GSM8K	88.81 $\pm$ 0.30	88.59 $\pm$ 0.20	90.58 $\pm$ 0.25	90.28 $\pm$ 0.25	89.07 $\pm$ 0.10	**94.40** $\pm$ 0.29
MATH500	89.41 $\pm$ 0.30	89.28 $\pm$ 0.19	91.00 $\pm$ 0.20	90.61 $\pm$ 0.25	89.44 $\pm$ 0.23	**92.90** $\pm$ 0.20
BBH	71.05 $\pm$ 0.75	71.70 $\pm$ 0.74	72.36 $\pm$ 0.65	72.55 $\pm$ 0.64	71.22 $\pm$ 0.72	73.91 $\pm$ 0.78
$C_{\max}=2$, $B_{\max}=9216$						
AIME24	46.09 $\pm$ 0.80	46.41 $\pm$ 0.73	46.72 $\pm$ 0.75	46.88 $\pm$ 0.73	46.56 $\pm$ 0.75	46.46 $\pm$ 0.81
AIME25	36.41 $\pm$ 0.66	36.61 $\pm$ 0.60	36.46 $\pm$ 0.64	36.30 $\pm$ 0.65	37.34 $\pm$ 0.66	35.73 $\pm$ 0.62
GSM8K	88.81 $\pm$ 0.30	88.92 $\pm$ 0.39	91.08 $\pm$ 0.26	91.05 $\pm$ 0.38	89.18 $\pm$ 0.16	**94.69** $\pm$ 0.21
MATH500	89.41 $\pm$ 0.30	89.47 $\pm$ 0.24	91.53 $\pm$ 0.19	91.70 $\pm$ 0.22	89.16 $\pm$ 0.25	**93.04** $\pm$ 0.17
BBH	71.05 $\pm$ 0.75	72.33 $\pm$ 0.43	72.58 $\pm$ 0.68	73.36 $\pm$ 0.50	71.21 $\pm$ 0.67	**74.53** $\pm$ 0.62
$C_{\max}=3$, $B_{\max}=9216$						
AIME24	46.09 $\pm$ 0.80	46.46 $\pm$ 0.73	47.24 $\pm$ 0.71	47.19 $\pm$ 0.77	46.41 $\pm$ 0.75	46.82 $\pm$ 0.76
AIME25	36.41 $\pm$ 0.66	36.51 $\pm$ 0.57	36.51 $\pm$ 0.61	36.77 $\pm$ 0.63	37.29 $\pm$ 0.71	36.09 $\pm$ 0.58
GSM8K	88.81 $\pm$ 0.30	89.27 $\pm$ 0.27	91.23 $\pm$ 0.26	91.74 $\pm$ 0.34	88.89 $\pm$ 0.39	**94.91** $\pm$ 0.18
MATH500	89.39 $\pm$ 0.31	89.84 $\pm$ 0.27	91.74 $\pm$ 0.19	92.17 $\pm$ 0.17	89.06 $\pm$ 0.27	**92.89** $\pm$ 0.20
BBH	71.05 $\pm$ 0.75	72.53 $\pm$ 0.72	72.91 $\pm$ 0.63	73.85 $\pm$ 0.51	71.01 $\pm$ 0.76	74.75 $\pm$ 0.60

Table 6: Accuracy (pass@1) results for `DeepSeek-R1-Distill-Qwen-7B` with different token budget limits $B_{\max}$ and different numbers of forced thinking continuations $C_{\max}$. Results are obtained via regex-based evaluation and an LLM evaluator if the model fails to generate an answer in the correct format.

Dataset	Baseline (w/o BF)	Alternatively	Hmm	Wait	Critique	Learned
$C_{\max}=1$						
AIME24	35.68 $\pm$ 0.78	36.15 $\pm$ 0.78	38.12 $\pm$ 0.75	37.34 $\pm$ 0.75	36.56 $\pm$ 0.70	**41.61** $\pm$ 0.82
AIME25	29.32 $\pm$ 0.61	29.74 $\pm$ 0.58	29.27 $\pm$ 0.60	29.84 $\pm$ 0.61	29.69 $\pm$ 0.64	30.05 $\pm$ 0.66
GSM8K	78.43 $\pm$ 0.40	78.22 $\pm$ 0.41	83.15 $\pm$ 0.40	81.28 $\pm$ 0.34	78.04 $\pm$ 0.56	**93.47** $\pm$ 0.12
MATH500	79.56 $\pm$ 0.31	80.64 $\pm$ 0.43	84.34 $\pm$ 0.28	83.41 $\pm$ 0.34	80.19 $\pm$ 0.35	**89.71** $\pm$ 0.25
BBH	79.73 $\pm$ 0.37	80.58 $\pm$ 0.36	80.92 $\pm$ 0.44	80.98 $\pm$ 0.37	79.92 $\pm$ 0.40	**83.25** $\pm$ 0.24
$C_{\max}=2$						
AIME24	35.68 $\pm$ 0.78	36.67 $\pm$ 0.78	37.71 $\pm$ 0.77	37.76 $\pm$ 0.86	36.15 $\pm$ 0.74	**41.87** $\pm$ 0.87
AIME25	29.32 $\pm$ 0.61	29.43 $\pm$ 0.56	29.06 $\pm$ 0.57	30.31 $\pm$ 0.60	29.48 $\pm$ 0.57	30.57 $\pm$ 0.61
GSM8K	78.43 $\pm$ 0.40	79.56 $\pm$ 0.41	84.12 $\pm$ 0.28	83.82 $\pm$ 0.28	77.93 $\pm$ 0.46	**93.73** $\pm$ 0.07
MATH500	79.56 $\pm$ 0.31	81.09 $\pm$ 0.36	85.40 $\pm$ 0.24	85.11 $\pm$ 0.37	80.34 $\pm$ 0.26	**89.91** $\pm$ 0.24
BBH	79.73 $\pm$ 0.37	81.01 $\pm$ 0.28	81.43 $\pm$ 0.47	81.71 $\pm$ 0.34	80.31 $\pm$ 0.29	**83.61** $\pm$ 0.32
$C_{\max}=3$						
AIME24	35.68 $\pm$ 0.78	36.93 $\pm$ 0.75	38.23 $\pm$ 0.73	38.39 $\pm$ 0.79	35.73 $\pm$ 0.71	**41.30** $\pm$ 0.91
AIME25	29.32 $\pm$ 0.61	29.64 $\pm$ 0.57	30.10 $\pm$ 0.61	30.26 $\pm$ 0.64	29.90 $\pm$ 0.58	30.26 $\pm$ 0.65
GSM8K	78.43 $\pm$ 0.40	79.82 $\pm$ 0.41	85.22 $\pm$ 0.22	84.90 $\pm$ 0.43	77.79 $\pm$ 0.65	**93.78** $\pm$ 0.10
MATH500	79.56 $\pm$ 0.31	81.86 $\pm$ 0.39	86.21 $\pm$ 0.32	86.01 $\pm$ 0.32	80.27 $\pm$ 0.40	**90.04** $\pm$ 0.25
BBH	79.73 $\pm$ 0.37	81.32 $\pm$ 0.30	81.79 $\pm$ 0.46	81.97 $\pm$ 0.32	80.26 $\pm$ 0.42	**83.69** $\pm$ 0.37

Table 7: Accuracy (pass@1) results for `DeepSeek-R1-Distill-Llama-8B` with $B_{\max}=9216$. Results are obtained via regex-based evaluation and an LLM evaluator if the model fails to generate an answer in the correct format.

Dataset	Baseline (w/o BF)	Alternatively	Hmm	Wait	Critique	Learned
$C_{\max} = 1$ $B_{\max} = 8192$						
AIME24	22.86 ± 0.75 (81)	22.71 ± 0.74 (81)	22.34 ± 0.68 (81)	22.60 ± 0.71 (82)	22.50 ± 0.53 (80)	22.66 ± 0.69 (82)
AIME25	21.56 ± 0.63 (85)	21.67 ± 0.55 (84)	21.72 ± 0.51 (84)	22.19 ± 0.52 (82)	21.98 ± 0.66 (83)	21.61 ± 0.55 (84)
GSM8K	66.43 ± 0.50 (84)	64.57 ± 0.33 (81)	65.01 ± 0.56 (81)	64.27 ± 0.37 (80)	68.57 ± 0.40 (86)	**75.71** ± 0.31 (92)
MATH500	77.19 ± 0.30 (94)	77.89 ± 0.34 (94)	77.94 ± 0.23 (94)	78.59 ± 0.29 (94)	78.35 ± 0.34 (94)	**80.84** ± 0.27 (96)
$C_{\max} = 2$ $B_{\max} = 8192$						
AIME24	22.86 ± 0.75 (81)	23.02 ± 0.83 (80)	23.23 ± 0.78 (80)	23.28 ± 0.75 (82)	22.71 ± 0.61 (79)	24.22 ± 0.76 (81)
AIME25	21.56 ± 0.63 (85)	21.41 ± 0.65 (84)	21.15 ± 0.67 (84)	21.25 ± 0.64 (84)	21.82 ± 0.64 (84)	21.82 ± 0.68 (83)
GSM8K	66.43 ± 0.50 (84)	64.52 ± 0.16 (81)	64.56 ± 0.16 (80)	63.55 ± 0.36 (79)	70.03 ± 0.59 (87)	**78.37** ± 0.40 (94)
MATH500	77.19 ± 0.30 (94)	77.64 ± 0.22 (94)	77.99 ± 0.30 (94)	78.05 ± 0.26 (94)	77.85 ± 0.31 (95)	**80.74** ± 0.18 (96)
$C_{\max} = 3$ $B_{\max} = 8192$						
AIME24	22.86 ± 0.75 (80)	22.55 ± 0.51 (79)	22.45 ± 0.57 (79)	22.92 ± 0.52 (81)	22.92 ± 0.74 (81)	22.92 ± 0.74 (81)
AIME25	21.56 ± 0.63 (85)	21.82 ± 0.67 (83)	21.41 ± 0.67 (85)	21.72 ± 0.61 (83)	22.50 ± 0.57 (83)	21.56 ± 0.58 (84)
GSM8K	66.43 ± 0.50 (83)	64.21 ± 0.46 (80)	64.42 ± 0.29 (79)	62.99 ± 0.43 (77)	70.25 ± 0.07 (88)	**80.11** ± 0.30 (95)
MATH500	77.19 ± 0.30 (94)	77.45 ± 0.31 (94)	78.54 ± 0.30 (94)	78.26 ± 0.20 (93)	78.31 ± 0.23 (95)	**81.60** ± 0.22 (96)
$C_{\max} = 1$ $B_{\max} = 9216$						
AIME24	23.18 ± 0.64 (82)	22.86 ± 0.62 (82)	23.54 ± 0.65 (81)	23.39 ± 0.60 (82)	23.18 ± 0.64 (81)	23.70 ± 0.62 (82)
AIME25	23.18 ± 0.50 (86)	23.13 ± 0.54 (87)	23.59 ± 0.58 (87)	23.33 ± 0.58 (86)	22.03 ± 0.54 (86)	23.18 ± 0.53 (86)
GSM8K	66.03 ± 0.40 (84)	65.76 ± 0.42 (82)	64.82 ± 0.52 (81)	64.24 ± 0.60 (80)	68.25 ± 0.51 (86)	**75.12** ± 0.33 (91)
MATH500	77.56 ± 0.32 (95)	77.78 ± 0.25 (95)	78.29 ± 0.34 (94)	78.15 ± 0.32 (94)	78.46 ± 0.23 (95)	**80.50** ± 0.35 (96)
$C_{\max} = 2$ $B_{\max} = 9216$						
AIME24	23.18 ± 0.64 (82)	23.12 ± 0.67 (82)	23.91 ± 0.67 (83)	23.75 ± 0.74 (83)	22.40 ± 0.71 (82)	23.59 ± 0.67 (83)
AIME25	23.18 ± 0.50 (86)	23.02 ± 0.49 (85)	23.23 ± 0.51 (86)	23.23 ± 0.50 (87)	22.14 ± 0.49 (85)	22.80 ± 0.47 (86)
GSM8K	66.03 ± 0.40 (84)	64.82 ± 0.25 (82)	64.09 ± 0.38 (80)	64.43 ± 0.48 (79)	69.99 ± 0.44 (88)	**78.27** ± 0.58 (94)
MATH500	77.56 ± 0.32 (95)	77.97 ± 0.35 (95)	78.88 ± 0.32 (95)	78.83 ± 0.38 (95)	77.56 ± 0.28 (95)	**81.37** ± 0.38 (97)
$C_{\max} = 3$ $B_{\max} = 9216$						
AIME24	23.18 ± 0.64 (82)	23.39 ± 0.70 (81)	24.06 ± 0.67 (82)	23.28 ± 0.72 (82)	22.60 ± 0.68 (82)	23.39 ± 0.66 (83)
AIME25	23.18 ± 0.50 (85)	23.44 ± 0.56 (87)	23.49 ± 0.49 (86)	23.12 ± 0.46 (86)	22.14 ± 0.46 (86)	22.40 ± 0.44 (86)
GSM8K	66.03 ± 0.40 (83)	63.79 ± 0.57 (80)	64.27 ± 0.28 (79)	62.94 ± 0.50 (77)	69.35 ± 0.56 (87)	**80.87** ± 0.42 (96)
MATH500	77.56 ± 0.32 (94)	78.16 ± 0.39 (95)	78.60 ± 0.40 (94)	78.75 ± 0.32 (94)	78.08 ± 0.28 (96)	**81.75** ± 0.34 (96)

Table 8: Accuracy (pass@1) results for `DeepSeek-R1-Distill-Qwen-1.5B` with different token budget limits $B_{\max}$ and different numbers of forced thinking continuations $C_{\max}$. Results are obtained via a regex-based evaluation only. The percentage of final answers enclosed in `\boxed{}` is shown in parentheses.

Dataset	Baseline (w/o BF)	Alternatively	Hmm	Wait	Critique	Learned
$C_{\max} = 1$						
$B_{\max} = 8192$						
AIME24	43.85 ± 0.77 (73)	43.65 ± 0.77 (71)	43.91 ± 0.77 (71)	44.38 ± 0.80 (71)	44.64 ± 0.82 (72)	44.79 ± 0.79 (72)
AIME25	34.74 ± 0.58 (65)	34.74 ± 0.60 (68)	34.69 ± 0.57 (67)	34.58 ± 0.57 (66)	34.74 ± 0.62 (65)	34.95 ± 0.56 (66)
GSM8K	88.56 ± 0.34 (99)	88.46 ± 0.30 (99)	90.45 ± 0.19 (99)	89.63 ± 0.13 (99)	89.19 ± 0.27 (99)	**94.43** ± 0.26 (99)
MATH500	88.72 ± 0.24 (97)	88.55 ± 0.31 (97)	90.06 ± 0.24 (97)	89.91 ± 0.21 (97)	88.54 ± 0.27 (97)	**92.20** ± 0.24 (97)
$C_{\max} = 2$						
$B_{\max} = 8192$						
AIME24	43.85 ± 0.77 (73)	44.32 ± 0.76 (73)	44.53 ± 0.80 (73)	44.53 ± 0.80 (72)	44.84 ± 0.80 (71)	44.48 ± 0.82 (71)
AIME25	34.74 ± 0.58 (65)	34.90 ± 0.58 (67)	34.90 ± 0.61 (65)	34.74 ± 0.63 (66)	34.17 ± 0.59 (65)	34.69 ± 0.59 (66)
GSM8K	88.56 ± 0.34 (99)	89.07 ± 0.19 (99)	91.03 ± 0.22 (99)	90.65 ± 0.17 (99)	88.61 ± 0.24 (99)	**94.91** ± 0.16 (99)
MATH500	88.72 ± 0.24 (97)	88.60 ± 0.27 (97)	90.80 ± 0.20 (97)	90.71 ± 0.21 (97)	88.65 ± 0.29 (97)	**92.28** ± 0.20 (97)
$C_{\max} = 3$						
$B_{\max} = 8192$						
AIME24	43.85 ± 0.77 (73)	44.32 ± 0.84 (71)	44.06 ± 0.81 (72)	45.00 ± 0.79 (71)	44.17 ± 0.78 (71)	44.58 ± 0.78 (72)
AIME25	34.74 ± 0.58 (65)	34.79 ± 0.57 (66)	34.53 ± 0.64 (65)	34.64 ± 0.59 (66)	34.48 ± 0.66 (66)	34.69 ± 0.57 (65)
GSM8K	88.56 ± 0.34 (99)	89.00 ± 0.24 (99)	91.11 ± 0.29 (99)	90.96 ± 0.28 (99)	88.61 ± 0.24 (99)	**95.11** ± 0.17 (99)
MATH500	88.72 ± 0.24 (97)	88.64 ± 0.27 (97)	91.13 ± 0.27 (97)	91.36 ± 0.18 (97)	88.29 ± 0.30 (97)	**92.24** ± 0.20 (97)
$C_{\max} = 1$						
$B_{\max} = 9216$						
AIME24	44.74 ± 0.77 (71)	45.42 ± 0.79 (71)	45.42 ± 0.78 (72)	45.99 ± 0.72 (72)	45.26 ± 0.77 (72)	45.16 ± 0.79 (71)
AIME25	35.47 ± 0.62 (67)	35.16 ± 0.61 (67)	35.47 ± 0.55 (66)	35.47 ± 0.61 (66)	36.61 ± 0.68 (68)	35.31 ± 0.56 (68)
GSM8K	88.81 ± 0.30 (99)	88.54 ± 0.19 (99)	90.52 ± 0.24 (99)	90.20 ± 0.24 (99)	89.00 ± 0.10 (99)	**94.33** ± 0.29 (99)
MATH500	88.94 ± 0.30 (97)	88.75 ± 0.20 (97)	90.68 ± 0.21 (97)	90.18 ± 0.23 (97)	88.86 ± 0.20 (97)	**92.58** ± 0.20 (97)
± 0.20 (97)						
$C_{\max} = 2$						
$B_{\max} = 9216$						
AIME24	44.74 ± 0.77 (71)	45.47 ± 0.78 (71)	45.94 ± 0.78 (72)	45.62 ± 0.79 (72)	45.42 ± 0.77 (71)	45.52 ± 0.80 (72)
AIME25	35.47 ± 0.62 (67)	35.68 ± 0.59 (68)	35.00 ± 0.62 (66)	35.62 ± 0.66 (67)	36.82 ± 0.67 (67)	34.90 ± 0.60 (68)
GSM8K	88.81 ± 0.30 (99)	88.82 ± 0.39 (99)	90.58 ± 0.29 (99)	90.85 ± 0.39 (99)	89.12 ± 0.14 (99)	**94.64** ± 0.19 (99)
MATH500	88.94 ± 0.30 (97)	88.90 ± 0.24 (97)	91.08 ± 0.19 (97)	91.24 ± 0.20 (97)	88.74 ± 0.26 (97)	**92.65** ± 0.18 (97)
$C_{\max} = 3$						
$B_{\max} = 9216$						
AIME24	44.74 ± 0.77 (71)	45.52 ± 0.72 (72)	46.15 ± 0.72 (71)	46.30 ± 0.74 (73)	45.21 ± 0.80 (73)	45.94 ± 0.79 (72)
AIME25	35.47 ± 0.62 (67)	35.47 ± 0.57 (66)	35.52 ± 0.62 (67)	35.73 ± 0.64 (66)	36.41 ± 0.68 (67)	35.16 ± 0.58 (66)
GSM8K	88.81 ± 0.30 (99)	89.15 ± 0.25 (99)	90.45 ± 0.24 (98)	91.51 ± 0.35 (99)	88.74 ± 0.38 (99)	**94.84** ± 0.18 (99)
MATH500	88.94 ± 0.30 (97)	89.42 ± 0.27 (97)	91.20 ± 0.16 (97)	91.78 ± 0.18 (97)	88.70 ± 0.26 (97)	**92.62** ± 0.20 (97)

Table 9: Accuracy (pass@1) results for `DeepSeek-R1-Distill-Qwen-7B` with different token budget limits $B_{\max}$ and different numbers of forced thinking continuations $C_{\max}$. Results are obtained via a regex-based evaluation only. The percentage of final answers enclosed in `\boxed{}` is shown in parentheses.

Probabilities	AIME24	AIME25	GSM8K	MATH500
$C_{\max} = 1$, $B_{\max} = 8192$				
P(Learned ✗, Baseline ✓)	0.09	0.06	0.03	0.06
P(Learned ✓, Baseline ✗)	0.08	0.06	0.06	0.09
$C_{\max} = 2$, $B_{\max} = 8192$				
P(Learned ✗, Baseline ✓)	0.01	0.01	0.03	0.02
P(Learned ✓, Baseline ✗)	0.02	0.02	0.07	0.04
$C_{\max} = 3$, $B_{\max} = 8192$				
P(Learned ✗, Baseline ✓)	0.09	0.06	0.02	0.01
P(Learned ✓, Baseline ✗)	0.09	0.06	0.07	0.04
$C_{\max} = 1$, $B_{\max} = 9216$				
P(Learned ✗, Baseline ✓)	0.02	0.01	0.03	0.02
P(Learned ✓, Baseline ✗)	0.02	0.01	0.06	0.03
$C_{\max} = 2$, $B_{\max} = 9216$				
P(Learned ✗, Baseline ✓)	0.08	0.06	0.02	0.02
P(Learned ✓, Baseline ✗)	0.08	0.06	0.07	0.04
$C_{\max} = 3$, $B_{\max} = 9216$				
P(Learned ✗, Baseline ✓)	0.09	0.06	0.02	0.02
P(Learned ✓, Baseline ✗)	0.09	0.06	0.07	0.04

Table 10: Probability of observing a correct answer from the learned token and an incorrect answer from the baseline and vice versa when using `DeepSeek-R1-Distill-Qwen-1.5B`.

Learned token

Question: Lloyd has an egg farm. His chickens produce 252 eggs per day and he sells them for $2 per dozen. How much does Lloyd make on eggs per week?

<think> First, I need to calculate the daily revenue from egg sales. Each chicken produces 252 eggs daily, so for 27 chickens, the total daily egg production is 252 multiplied by 27, which equals 6,804 eggs. Next, I'll convert the number of eggs into dozens since the selling price is given per dozen. There are 12 eggs in a dozen, so the total dozens of eggs sold daily are 6,804 divided by 12, resulting in 567 dozen. Now, I'll find out how much 567 dozen eggs sell for per day by multiplying the number of dozens by the selling price per dozen. That's 567 multiplied by $2, which equals $1,134. Finally, to determine the weekly revenue, I'll multiply the daily revenue by the number of days in a week, which is 7. So, $1,134 multiplied by 7 equals $7,938.

<|continue_thinking|> no, let me try that again. If I have 27 chickens each producing 252 eggs, that's 27*252=6,804 eggs per day. He sells them for $2 per dozen,which is $2/12 per egg. So to find daily revenue, it's 6,804/12 *2 = $1,134. Then multiply by 7 to get weekly: 1,134*7= $7,938. Wait, did I just repeat the same steps? Maybe I need to approach it differently. Alternatively, first calculate the daily revenue. 252 eggs produce 252/12=21 dozen. Each dozen sells for $2, so 21*2=$42 per day. Then, over a week (7 days), it's 42*7=$294. But I'm confused earlier got $7,938. Which one is correct? Let me double-check. 27 chickens at 252 eggs each is indeed 27*252=6,804 eggs. 6,804 divided by 12 is 567 dozen. 567*2= $1,134 per day. Then 1,134*7= $7,938. Oh, wait. I think I'm making a mistake in the calculation order. Let me clarify: Egg production: 27 chickens * 252 eggs/day = 6,804 eggs/day. Dollars per egg: $2 per dozen, which is $2/12= $0.1667 per egg. Total revenue per day: 6,804 eggs/day * $0.1667/egg ≈ $1,134. Revenue per week: $1,134/day *7= $7,938. But when I calculated earlier, I got a different result. What's the mistake? Wait, 252 eggs per day is 252 divided by 12 is 21 dozen. 21*2= $42 daily. 42*7= $294. Now, why the discrepancy? Let's check 27*252. 252*27: 27*200=5,400, 27*52=1,404, total is 5,400+1,404=6,804. Yes, that's correct. So 27 chickens produce 6,804 eggs a day. At $2 per dozen, 6,804/12=567 dozen, 567*2= $1,134. 1,134*7= $7,938. But why is another method giving a different answer? Wait, maybe I'm interpreting the question wrong. If there are 27 chickens, each producing 252 eggs a day, so total eggs per day are 27*252=6,804. To find revenue, I need to know how much they sell each day. If they sell each dozen for $2, then per day revenue is 6,804/12*$2=567*$2= $1,134. Over 7 days, $1,134*7= $7,938. Alternatively, 27 chickens, 252 eggs/day: total eggs per week. 252*7=1,764 eggs/week. Convert to dozens: 1,764/12=147 dozen. 147*2= $294 per week. Ah, I see now. Depending on how you group the eggs, you get different per-week revenues. The confusion comes from whether the revenue is per dozen or per week. The question says "how much does Lloyd make on eggs per week." Since the chickens produce 252 eggs a day, and he sells them for $2 per dozen, the method is to calculate how much he makes per dozen eggs multiplied by how many dozens he sells per week. So, per week, he sells 252*7=1,764 eggs. Divided by 12, that's 147 dozen. 147*2= $294 per week. Therefore, the correct answer is $294 per week.

</think> Lloyd sells his eggs at $2 per dozen. To determine his weekly revenue:

1. **Calculate weekly egg production**: 252 eggs/day * 7 days = 1,764 eggs/week
2. **Convert weekly eggs to dozens**: 1,764 eggs / 12 = 147 dozen
3. **Calculate weekly revenue**: 147 dozen * $2/dozen = $294

Answer: Lloyd makes `\boxed{294}` dollars per week from his egg sales.

Figure 6: GSM8K reasoning trace from `DeepSeek-R1-Distill-Qwen-1.5B` demonstrating the positive impact of the `<|continue_thinking|>` token. Blue indicates the original reasoning, yielding an incorrect answer of 7,938. Green shows the continuation after the special token was added, leading to the correct answer of 294.

Baseline

Question: Lloyd has an egg farm. His chickens produce 252 eggs per day and he sells them for $2 per dozen. How much does Lloyd make on eggs per week?

<think> First, I need to calculate the daily revenue from egg sales. Each chicken produces 252 eggs daily, so for 27 chickens, the total daily egg production is 252 multiplied by 27, which equals 6,804 eggs. Next, I'll convert the number of eggs into dozens since the selling price is given per dozen. There are 12 eggs in a dozen, so the total dozens of eggs sold daily are 6,804 divided by 12, resulting in 567 dozen. Now, I'll find out how much 567 dozen eggs sell for per day by multiplying the number of dozens by the selling price per dozen. That's 567 multiplied by $2, which equals $1,134. Finally, to determine the weekly revenue, I'll multiply the daily revenue by the number of days in a week, which is 7. So, $1,134 multiplied by 7 equals $7,938.

</think>

Solution

To determine how much Lloyd makes on eggs per week, we'll break down the problem step by step.

1. **Calculate the total number of eggs produced per day:**

$$\text{Eggs per day} = \text{Number of chickens} \times \text{Eggs per chicken per day}$$

$$\text{Eggs per day} = 27 \times 252 = 6{,}804 \text{ eggs}$$

2. **Determine the number of dozens of eggs produced per day:**

$$\text{Dozens per day} = \frac{\text{Eggs per day}}{12}$$

$$\text{Dozens per day} = \frac{6{,}804}{12} = 567 \text{ dozen}$$

3. **Calculate the daily revenue from selling eggs:**

$$\text{Revenue per day} = \text{Dozens per day} \times \text{Selling price per dozen}$$

$$\text{Revenue per day} = 567 \times 2 = 1{,}134 \text{ dollars}$$

4. **Find the weekly revenue by multiplying daily revenue by the number of days in a week:**

$$\text{Revenue per week} = \text{Revenue per day} \times 7$$

$$\text{Revenue per week} = 1{,}134 \times 7 = 7{,}938 \text{ dollars}$$

Final Answer: `\boxed{7,938 dollars}`

Figure 7: GSM8K reasoning trace from `DeepSeek-R1-Distill-Qwen-1.5B` of the baseline model for the same question as in Figure 6. The final answer provided by the baseline model is incorrect, as opposed to the correct answer given in Figure 6.

Learned token

Question: Twenty tourists discovered 700 shells in a strip mall parking lot. They had divided into three groups, Alphas, The finders, and Gogetters to find as many shells as possible. If team Alphas found 40% of the shells, and team The finders found 60% of the remaining shells, how many shells did team Gogetters find?

<think> Okay, let's try to figure out how many shells team Gogetters found. So, there are 700 shells in total, and 20 tourists split them into three groups: Alphas, The finders, and Gogetters. I need to find out how many each group found. First, team Alphas found 40% of the shells. That sounds like a straightforward calculation. So, 40% of 700. Let me write that down: 0.4 * 700. Hmm, 0.4 is like 40%, so if I multiply 700 by 0.4, what do I get? Let me do that math. 0.4 times 700... Yeah, that's 280. So, team Alphas found 280 shells. Now, after team Alphas, there are 700 - 280 = 420 shells left. That number is the remaining shells after Alphas' team found theirs. Next, team The finders found 60% of these remaining shells. So, I need to calculate 60% of 420. Alright, 0.6 times 420. Let me compute that. 0.6 * 400 is 240, and 0.6 * 20 is 12. So, 240 + 12 equals 252. Therefore, team The finders found 252 shells. Now, to find out how many shells team Gogetters found, I subtract the shells found by both Alphas and The finders from the total. So, 700 - 280 - 252. Let me do that step by step. First, 700 - 280 is 420, as we had before. Then, 420 - 252. Hmm, 420 minus 200 is 220, and then minus 52 more is 168. So, team Gogetters found 168 shells. Let me just double-check my calculations to make sure I didn't make a mistake. Alphas: 40% of 700 is indeed 280. Subtracting that leaves 420. The finders take 60% of 420, which is 252. Subtracting 252 from 420 gives 168 for Gogetters. That all adds up correctly. I think that's all. The Gogetters found 168 shells.

<|continue_thinking|> , and he's right.

</think> To determine how many shells team Gogetters found, follow these steps:

1. **Calculate the number of shells found by team Alphas:**

$$40\% \text{ of } 700 = 0.4 \times 700 = 280 \text{ shells}$$

2. **Subtract the shells found by Alphas from the total:**:

$$700 - 280 = 420 \text{ shells remaining}$$

3. **Calculate the number of shells found by team The finders:**

$$60\% \text{ of } 420 = 0.6 \times 420 = 252 \text{ shells}$$

4. **Subtract the shells found by The finders from the remaining shells:**

$$420 - 252 = 168 \text{ shells}$$

Conclusion: Team Gogetters found `\boxed{168}` shells.

Figure 8: GSM8K reasoning trace from `DeepSeek-R1-Distill-Qwen-1.5B` demonstrating that the `<|continue_thinking|>` token does not generate many tokens when the model is confident. Blue indicates the original reasoning. Green shows the short continuation after the special token was added.

Baseline

Question: Twenty tourists discovered 700 shells in a strip mall parking lot. They had divided into three groups, Alphas, The finders, and Gogetters to find as many shells as possible. If team Alphas found 40% of the shells, and team The finders found 60% of the remaining shells, how many shells did team Gogetters find?

<think> Okay, let's try to figure out how many shells team Gogetters found. So, there are 700 shells in total, and 20 tourists split them into three groups: Alphas, The finders, and Gogetters. I need to find out how many each group found. First, team Alphas found 40% of the shells. That sounds like a straightforward calculation. So, 40% of 700. Let me write that down: 0.4 * 700. Hmm, 0.4 is like 40%, so if I multiply 700 by 0.4, what do I get? Let me do that math. 0.4 times 700... Yeah, that's 280. So, team Alphas found 280 shells. Now, after team Alphas, there are 700 - 280 = 420 shells left. That number is the remaining shells after Alphas' team found theirs. Next, team The finders found 60% of these remaining shells. So, I need to calculate 60% of 420. Alright, 0.6 times 420. Let me compute that. 0.6 * 400 is 240, and 0.6 * 20 is 12. So, 240 + 12 equals 252. Therefore, team The finders found 252 shells. Now, to find out how many shells team Gogetters found, I subtract the shells found by both Alphas and The finders from the total. So, 700 - 280 - 252. Let me do that step by step. First, 700 - 280 is 420, as we had before. Then, 420 - 252. Hmm, 420 minus 200 is 220, and then minus 52 more is 168. So, team Gogetters found 168 shells. Let me just double-check my calculations to make sure I didn't make a mistake. Alphas: 40% of 700 is indeed 280. Subtracting that leaves 420. The finders take 60% of 420, which is 252. Subtracting 252 from 420 gives 168 for Gogetters. That all adds up correctly. I think that's all. The Gogetters found 168 shells.

</think>

There are 700 shells in total. Team Alphas found 40

$$0.4 \times 700 = 280 \text{ shells}$$

The remaining shells after Alphas' team are:

$$700 - 280 = 420 \text{ shells}$$

Team The finders found 60

$$0.6 \times 420 = 252 \text{ shells}$$

Finally, the number of shells found by team Gogetters is:

$$700 - 280 - 252 = 168 \text{ shells}$$

`\boxed{168}`

Figure 9: GSM8K reasoning trace from `DeepSeek-R1-Distill-Qwen-1.5B` of the baseline model for the same question as in Figure 8. In this case, both the baseline and learned token output a correct answer and the answer provided by the baseline model is almost identical to the one provided by the learned token.

Learned token

Question: Michael is traveling on the interstate at an average speed of 50km/hr (taking into account all the necessary stops for gas, relaxation, etc.). If he goes on like this for 3 days, and the distance between Alaska and Texas is 6000 km, what percentage of this distance has he covered?

<think> Okay, so Michael is traveling on the interstate and wants to know what percentage of the distance between Alaska and Texas he has covered. The distance is 6000 km, and he's traveling at an average speed of 50 km/hr for 3 days. I need to figure out how much he's traveled in that time and then find out what percentage that is of the total distance.
First, I should calculate the total distance Michael has covered in 3 days. His speed is 50 km/hr, so I need to multiply that by the number of hours he spends traveling each day. But since the days are different temperatures and maybe he works longer hours, I should consider the average hours per day. Let's say he works 8 hours each day. So, for 3 days, that would be 8 hours/day * 3 days = 24 hours.
Now, multiplying his speed by the total hours: 50 km/hr * 24hr = 1200 km. So, Michael has traveled 1200 km in those 3 days.
Next, to find out what percentage this is of the total distance, which is 6000 km. I can use the percentage formula: (part/whole) * 100
Wait, but is the 8 hours per day an accurate average? If his work hours vary, the total time could be different. Maybe I should calculate based on days without considering hours. But that seems complicated without specific hours. Alternatively, perhaps I should clarify if 50 km/hr is the average speed per day or total speed. But the question says he's traveling at 50 km/hr, so I think it's average speed over total time.
Alternatively, maybe the question expects simply to use the average speed over the total distance. Let me check again: the question states he's traveling at 50 km/hr, so it's average speed per day, not per trip. Therefore, the total distance is 50 km/hr * 8 hours/day * 3 days = 1200 km, as I calculated before.
So, 1200 km is 20% of 6000 km.

<|continue_thinking|> , is it 1200/6000=0.2, which is 20%. Yeah, that's correct.

</think> To determine what percentage of the 6000 km distance Michael has covered, we first calculate the total distance he has traveled. Assuming he works 8 hours each day:

$$50\,\text{km/hr} \times 8\,\text{hr/day} \times 3\,\text{days} = 1200\,\text{km}$$

Then, we find the percentage of the total distance:

$$\left(\frac{1200\,\text{km}}{6000\,\text{km}}\right) \times 100\% = 20\%$$

`\boxed{20%}`

Figure 10: GSM8K reasoning trace from `DeepSeek-R1-Distill-Qwen-1.5B` of a wrong answer given by both models.

Baseline

Question: Michael is traveling on the interstate at an average speed of 50km/hr (taking into account all the necessary stops for gas, relaxation, etc.). If he goes on like this for 3 days, and the distance between Alaska and Texas is 6000 km, what percentage of this distance has he covered?

<think> Okay, so Michael is traveling on the interstate and wants to know what percentage of the distance between Alaska and Texas he has covered. The distance is 6000 km, and he's traveling at an average speed of 50 km/hr for 3 days. I need to figure out how much he's traveled in that time and then find out what percentage that is of the total distance.
First, I should calculate the total distance Michael has covered in 3 days. His speed is 50 km/hr, so I need to multiply that by the number of hours he spends traveling each day. But since the days are different temperatures and maybe he works longer hours, I should consider the average hours per day. Let's say he works 8 hours each day. So, for 3 days, that would be 8 hours/day * 3 days = 24 hours.
Now, multiplying his speed by the total hours: 50 km/hr * 24hr = 1200 km. So, Michael has traveled 1200 km in those 3 days.
Next, to find out what percentage this is of the total distance, which is 6000 km. I can use the percentage formula: (part/whole) * 100
Wait, but is the 8 hours per day an accurate average? If his work hours vary, the total time could be different. Maybe I should calculate based on days without considering hours. But that seems complicated without specific hours. Alternatively, perhaps I should clarify if 50 km/hr is the average speed per day or total speed. But the question says he's traveling at 50 km/hr, so I think it's average speed over total time.
Alternatively, maybe the question expects simply to use the average speed over the total distance. Let me check again: the question states he's traveling at 50 km/hr, so it's average speed per day, not per trip. Therefore, the total distance is 50 km/hr * 8 hours/day * 3 days = 1200 km, as I calculated before.
So, 1200 km is 20% of 6000 km.

</think> Michael has traveled 20% of the distance between Alaska and Texas.

```
\boxed{20%}
```

Figure 11: GSM8K reasoning trace from `DeepSeek-R1-Distill-Qwen-1.5B` of the baseline model for the same question as in Figure 10.