# Towards Efficient Chain-of-Thought Reasoning via Adaptive-Budgeting based Policy Optimization

**Anonymous ACL submission**


## Abstract

*Chain-of-Thought* (CoT) reasoning has become a key problem-solving capability for advanced *large language models* (LLMs), but the "*over-thinking*" issue still plagues LLMs. In this paper, we propose a novel learning scheme termed *Adaptive-Budgeting based Policy Optimization* (ABPO) to balance the performance and efficiency of CoT reasoning. Our ABPO defines the RL training as an *adaptive curriculum learning* process, where example pools are curated to categorize training examples into three types, namely the *mastered*, *learning* and *hard* ones, respectively. As the training progresses, ABPO will adaptively schedule the examples with proper length budgets, and the example pools will also be dynamically updated based on the model status, thereby achieving a good balance between efficiency and performance of CoTs. The experimental results not only show the substantial efficiency improvements by ABPO, *e.g.*, reducing token length by 78.3% while improving 2.0% performance of DeepSeek-R1-Distill-Qwen-1.5B on average, but also show its obvious advantages over the compared methods, *e.g.*, reducing 59.4% length and increasing 8.3% performance on average than HAPO, respectively. Our code is anonymously released at `https://anonymous.4open.science/r/AnonymizeABPO-5380/`


## 1 Introduction

Recently, *test time scaling* (Muennighoff et al., 2025; Snell et al., 2024; OpenAI et al., 2024) has become the *de facto* paradigm of advanced *large language models* (LLMs) (Grattafiori et al., 2024; Yang et al., 2024; Bai et al., 2023; Achiam et al., 2023). Following this paradigm, advanced LLMs (DeepSeek-AI, 2025; Team, 2025) are trained in reinforced or supervised manners to generate long *chain-of-thoughts* (CoTs), thereby handling a set of difficult tasks, such as mathematical problem solving (MAA, 2024; Gao et al., 2024) and logical reasoning (Xie et al., 2025).

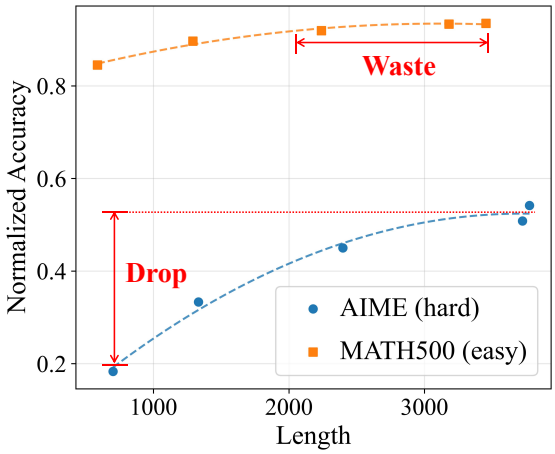


Figure 1: The performance–length curve of L1-Exact 1.5B (Aggarwal and Welleck, 2025) on two datasets. For easy questions, LLMs will show notable *over-thinking* issues, leading to a waste of computation. In contrast, for the hard ones, the limited token budgets of CoT will result in obvious performance drop. Thus, how to assign proper token budgets to different questions is the key challenge of CoT reasoning.

Despite the great success, the case of "*over-thinking*" (Chen et al., 2025) becomes another notable issue for these LLMs. In particular, over-thinking refers to that LLMs excessively engaging in *reflection*, *backtracking* and *self-verification* when answering questions (Min et al., 2024). For instance, given a simple question, such as *"The product of integers 240 and $k$ is a perfect cube. What is the smallest possible positive value of $k$?"*, the advanced LLMs like *Qwen-QwQ* (Team, 2025) will produce over $4k$ tokens to form its CoTs and consume about 250 seconds. In stark contrast, Qwen-QwQ only takes about 20 seconds to answer the question without thinking, which uses less than 400 tokens. This issue becomes more prominent in smaller LLMs due to the inferior reasoning capability, such as DeepSeek-R1-Distill-Qwen-1.5B (DeepSeek-AI, 2025) and DeepScaleR-1.5B-Preview (Luo et al., 2025b), which strongly contra-

dicts their goals towards efficient applications.

To address this problem, recent efforts are devoted to the research of efficient CoT reasoning (Muennighoff et al., 2025; Aggarwal and Welleck, 2025; Ma et al., 2025; Luo et al., 2025a). Among these progresses, one plausible strategy is to imply length-related rewards to the RL training of LLMs (Aggarwal and Welleck, 2025; Yeo et al., 2025; Hou et al., 2025; Hammoud et al., 2025). In particular, LLMs are assigned by a fixed-length token budget during its training. Thus, when the output CoT exceeds the token budget, a negative reward will be returned to penalize the prediction. In this way, LLMs are forced to learn to solve the tasks using limited-length CoTs.

However, this fixed-budget based solution still encounters several limitations. Above all, the example difficulties for LLMs vary greatly, even for the mathematical tasks (Hendrycks et al., 2021; He et al., 2024; MAA, 2024), and fixing the token budget is hard to achieve the trade-off between efficiency and performance. As shown in Fig.1, when the token budget is small, LLMs are prone to undercutting the hard examples while over-supplying the easy ones. For instance, under a budget of $2k$ tokens, DeepSeek-R1-Distill-Qwen-1.5B can perform well on the simple tasks like MATH500 but receives obvious performance drops on the more difficult ones, *e.g.*, -25% on AIME (MAA, 2024). Moreover, simply increasing the token budget will make the efficiency benefit become limited, *e.g.*, $4k$ tokens, and the tuning cost also increases substantially. In this case, we argue that allocating appropriate token budgets to different examples is the key to alleviate the *over-thinking* issue.

However, achieving the adaptive budgeting is still challenging in existing RL training schemes. Although the example difficulty for LLMs can be measured via *accuracy*, *entropy* or *prediction confidence* (Su and Cardie, 2025; AlKhuzaey et al., 2024), this assessment by the LLM is a dynamic process that changes as training progresses, and how to assign proper token budgets is intractable. Besides, the training scheduler also requires in-depth exploration. Gradually increasing the token budget is a natural solution, but it also causes the LLM to continuously traverse the simple examples, resulting additional issues about training efficiency and expenditure (Luo et al., 2025b). Therefore, how to effectively and dynamically adjust proper token budgets still remains an open problem.

To address these issues, we propose a novel training scheme termed *Adaptive-Budgeting based Policy Optimization* (ABPO) following the principle of *curriculum learning* (Bengio et al., 2009). Concretely, ABPO will categorize the training examples into three pools, namely the *mastered*, the *learning* and the *hard* ones, respectively. And these example pools will be dynamically updated based on the model status. In practice, ABPO starts from a low token budget, which is then gradually increased. During training, the hard examples are not directly used, but they will be adaptively updated to the *learning* pool according to the learning status. Meanwhile, we also introduce an efficient review mechanism for the *mastered* examples, ensuring the knowledge retention and avoiding catastrophic forgetting. This dynamic scheduling can help to assign model-aware token budgets for different examples and avoid the redundant training of examples. In this case, ABPO can achieve a good trade-off between CoT efficiency and performance as well as training expenditure.

To validate ABPO, we apply it to a set of advanced LLMs, including DeepSeek-R1-Distill-Qwen-1.5B and 7B (DeepSeek-AI, 2025) and DeepScaleR-1.5B-Preview (Luo et al., 2025b), on four representative reasoning benchmarks of varying difficulties, which are MATH500 (Hendrycks et al., 2021), AIME (MAA, 2024), AMC (MAA, 2023) and OlympiadBench (He et al., 2024). The experiment results show that ABPO can significantly reduce the token cost of CoT reasoning while maintaining competitive performance across various benchmarks. For instance, ABPO reduces the average CoT length of DeepSeek-R1-Distill-Qwen-1.5B by 78.3% while improving 2.0% of performance on average. Compared with prior CoT optimization methods, our ABPO can also obtain better CoT efficiency while achieving overall performance improvement on all benchmarks, *e.g.*, 2,217 *v.s.*, 5,466 length and 60.2 *v.s.* 55.6 accuracy compared to HAPO (Huang et al., 2025). These results well confirm the effectiveness of our motivation and designs towards efficient CoT.

Overall, our contributions are three-fold:

- We study the issue of *over-thinking* of LLMs from the perspective of adaptive CoT reasoning, and also identify the key challenges of balancing CoT performance and efficiency.
- We propose a novel training scheme termed *Adaptive-Budgeting based Policy Optimization* (ABPO), which can adaptively assign

token budgets and dynamically schedule the training examples for efficient CoT reasoning.

- On a set of LLMs and benchmarks, the proposed ABPO can help LLMs cultivate a better balance between CoT efficiency and performance than the default LLMs and the compared methods.

## 2 Related work

*Chain-of-Thought* (CoT) refers to generating stepwise reasoning traces before answering the given question (Wei et al., 2022), which has become a critical capability of advanced LLMs (DeepSeek-AI, 2025; Luo et al., 2025b; Team, 2025; Team et al., 2025). However, these reasoning LLMs often produce excessively long CoTs, which commonly involves too many branching, verification, and backtracking. To address this issue, several works have recently been proposed for efficient CoT (Hou et al., 2025; Yeo et al., 2025; Hammoud et al., 2025; Qi et al., 2025; Xia et al., 2025; Zhang et al., 2025; Dumitru et al., 2025). Specifically, one popular solution is using fixed-budget rewards in RL training (Aggarwal and Welleck, 2025; Yeo et al., 2025; Hou et al., 2025; Hammoud et al., 2025). For example, L1 (Aggarwal and Welleck, 2025) introduces a penalty term to enforce length budgets specified in the prompt. ThinkPrune (Hou et al., 2025) apply a uniform budget limit to all training samples. Curriculum GRPO (Hammoud et al., 2025) gradually reduce the number of tokens that is allowed to use. However, these fixed-budget based methods often struggle to achieve a good trade-off between efficiency and performance. In this case, recent works (Luo et al., 2025a; Huang et al., 2025; Arora and Zanette, 2025; Su and Cardie, 2025; Yi et al., 2025) focus on dynamic-length reward. O1-Pruner (Luo et al., 2025a) and TrainingEfficiently (Arora and Zanette, 2025) apply a length penalty based on the within-group comparison. ShorterBetter (Yi et al., 2025) ties its reward function to the length of the shortest and correct response for each sample. Although these methods incorporate dynamic reward signals, they still need to repeatedly train on the mastered samples, which significantly reduces training efficiency. In contrast, our approach considers the budget and training schedule for different samples, achieving a balance between performance and efficiency.

## 3 Method

### 3.1 Preliminary

*Chain-of-thought* (CoT) (Wei et al., 2022) is a key capability of advanced LLMs for addressing complex and difficult tasks. The inference process of an LLM with CoT can be defined by

$$\hat{y}_i = LLM(o_i, q_i), \quad where\ o_i = LLM(q_i). \tag{1}$$

Here $q_i$ denotes the input question, $o_i$ is the generated reasoning trace, *i.e.*, CoT, and $\hat{y}_i$ is the predicted answer. Under the popular *Group Relative Policy Optimization* (GRPO) (Shao et al., 2024), its objective can be defined by

$$\begin{aligned}
&\mathcal{J}_{GRPO}(\theta) = \mathbb{E}[q \sim D, \{o_i\}_{i=1}^{G} \sim \pi_{\theta_{old}}(O|q)] \\
&\frac{1}{G}\sum_{i=1}^{G}\frac{1}{|o_i|}\sum_{t=1}^{|o_i|}\left[w_{i,t}(\theta)\hat{A}_{i,t} - \beta L_{KL}(\pi_\theta, \pi_{ref})\right], \\
&\text{where } w_{i,t}(\theta) = \frac{\pi_\theta(o_{i,t}|q, o_{i,<t})}{\pi_{\theta_{old}}(o_{i,t}|q, o_{i,<t})}, \\
&\hat{A}_{i,t} = \frac{\mathbb{I}[\hat{y}_i = y_i] - mean(R)}{std(R)}.
\end{aligned} \tag{2}$$

Here, $q$ denotes the sampled query from the dataset $D$, $o_i$ is the $i$-th generated output sampled from the old policy $\pi_{\theta_{old}}$, $\pi_\theta$ is the updated policy for optimization, and $\pi_{ref}$ is the reference model used for KL-regularization. $\hat{A}_{i,t}$ is the standardized reward advantage for token $t$ with $R$ denoting the set of rewards. $\beta$ controls the strength of the *KL-regularization* $L_{KL}(\pi_\theta, \pi_{ref})$.

Recent research (Liu et al., 2025) show that models trained with GRPO, such as DeepSeek-R1 (DeepSeek-AI, 2025), are prone to *over-thinking* (Chen et al., 2025). And this issue will become more obvious in smaller distilled LLMs (Cuadron et al., 2025). In this case, some recent works (Hou et al., 2025; Aggarwal and Welleck, 2025; Yeo et al., 2025) implement length-related reward to RL training. As discussed in Sec. 2, the length-aware reward can encourage the model to learn to answer the questions with limited length of CoTs, but they will also make the model undercutting the hard examples while over-supplying the easy ones. In this case, we aim to achieve dynamic RL training with adaptive token budget scheduling.

### 3.2 Adaptive-Budgeting based Policy Optimization

In this paper, we propose a novel training scheme to balance the efficiency and performance of LLM's

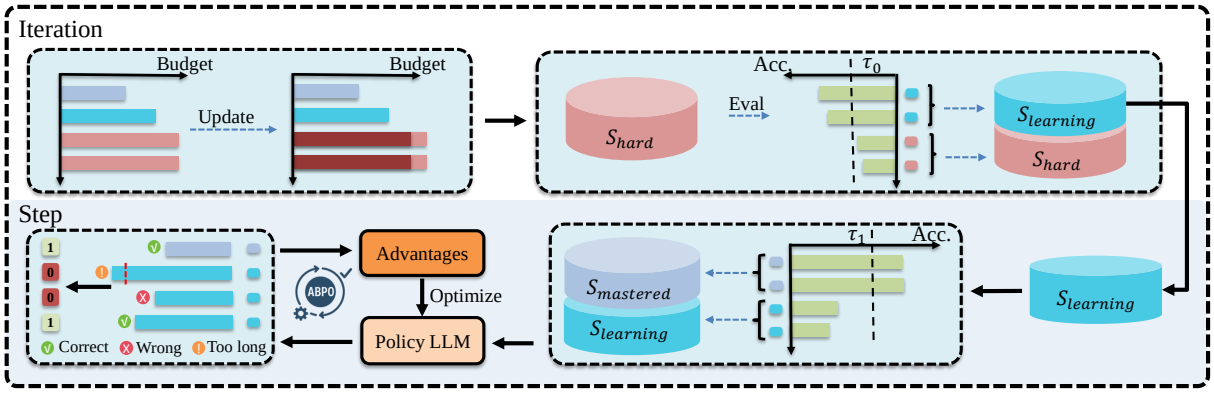


Figure 2: Illustration of *Adaptive Budgeting based Policy Optimization* (ABPO). During training, ABPO follows the principle of *curriculum learning* and groups the training examples into three pools, *i.e.*, the *mastered*, *learning* and *hard* ones, respectively. As the training progresses, ABPO will adaptively allocate proper token budgets to the learning examples, and also actively review the mastered ones to avoid knowledge forgetting. In addition, these example pools will be dynamically updated based on the LLM's status during training, thereby balancing efficiency, performance and training costs.

CoT reasoning, termed *Adaptive-Budgeting based Policy Optimization* (ABPO), of which illustration is depicted in Fig. 2. Concretely, the objective of ABPO can be defined by

$$\begin{aligned}
\mathcal{J}_{\text{ABPO}}(\theta) &= \mathbb{E}[q \sim D, \{o_i\}_{i=1}^{G} \sim \pi_{\theta_{\text{old}}}(O|q)] \\
&\frac{1}{G}\sum_{i=1}^{G}\frac{1}{T_i}\sum_{t=1}^{T_i}\left[w_{i,t}\hat{A}_{i,t} - \beta L_{\text{KL}}(\pi_\theta, \pi_{\text{ref}})\right], \\
\text{where} \quad T_i &= \min(|o_i|, b(q_i)), \\
w_{i,t} &= \frac{\pi_\theta(o_{i,t}|q, o_{i,<t})}{\pi_{\theta_{\text{old}}}(o_{i,t}|q, o_{i,<t})}, \\
\hat{A}_{i,t} &= \frac{r(o_i, \hat{y}_i) - \text{mean}(R)}{\text{std}(R)}.
\end{aligned} \tag{3}$$

Here $b(q_i)$ is the dynamic token budget that LLM can use to answer the question $q_i$. And the budget-aware reward function $r$ can be defined by

$$r(o_i, \hat{y}_i) = \begin{cases} 1, & \text{if } \hat{y}_i = y_i \text{ and } |o_i| \leq b(q_i), \\ 0, & \text{otherwise.} \end{cases} \tag{4}$$

where $o_i$ and $\hat{y}_i$ are the reasoning trace and the predicted answer for the input question $q_i$. Notably, $b(q_i)$ will be adaptively adjusted according to the difficulties of examples to the LLM.

Before training, ABPO first considers that all examples are unknown (hard) to the model, and evaluates their accuracies for the current model $\pi_\theta$. In practice, we will first generate $n$ CoT traces $O_i$ for each question $q_i$:

$$O_i = \{(o_{i,j}, \hat{y}_{i,j}) \mid (o_{i,j}, \hat{y}_{i,j}) = \pi_{\theta,j}(q_i)\}, \tag{5}$$

where $\pi_{\theta,j}$ represents the $j$-th sampling based on the policy model $\pi_\theta$. Notably, these traces can also be used as $\{o_i\}_{i=1}^{G}$ in Eq.3, avoiding the waste of computation. Then, the average accuracy of the model $\pi_{\theta_{old}}$ for the question $q_i$ under the length budget $b(q_i)$ can be obtained by

$$Acc(q_i) = \frac{1}{n}\sum_{j=1}^{n}\mathbb{I}[\,\hat{y}_{ij} = y_i \;\wedge\; |o_{i,j}| \leq b(q_i)\,], \tag{6}$$

where $\mathbb{I}[\cdot]$ is an indicator function. Initially, all samples are given the same budget $b(q_i) = t_0$. Via Eq. 6, we can categorize these examples into three pools, defined by

$$\mathcal{S}_{\text{mastered}} = \{q_i \mid Acc(q_i) \geq \tau_1\}, \tag{7}$$

$$\mathcal{S}_{\text{learning}} = \{q_i \mid \tau_1 > Acc(q_i) > \tau_0\}, \tag{8}$$

$$\mathcal{S}_{\text{hard}} = \{q_i \mid \tau_0 \geq Acc(q_i)\}. \tag{9}$$

Here,$\tau_0$ and the $\tau_1$ are the lower and upper thresholds, respectively. Thus, the mastered pool $\mathcal{S}_{\text{mastered}}$ contains examples that have been solved under existing token budget $b(q_i)$. The learning pool $\mathcal{S}_{\text{learning}}$ have examples with a certain probability of being learned. In contrast, the hard pool $\mathcal{S}_{\text{hard}}$ contains difficult examples that are not suitable to learn by LLMs, and they will not be used.

During ABPO training, the examples used are mainly composed of the *learning* ones, but we will still review a certain number of the *mastered* examples to avoid knowledge forgetting, with a review rate denoted by $\alpha$. Crucially, the reasoning traces generated during difficulty evaluation are reused as the rollouts for the policy update. After optimizing

the LLM with the current batch, we calculate their accuracies according to Eq.6. Based on the accuracies, we can update the examples pools $S_{learning}$ and $S_{mastered}$. When $|\mathcal{S}_{\text{learning}}|$ is reduced to a certain proportion $\lambda$, we will increase the length budgets $b(q_i)$ for the samples in $\mathcal{S}_{\text{hard}}$, and then start a new difficulty evaluation for these examples, which will accordingly update the pools of both hard and learning.

Specifically, the new budget for the questions $q_i \in \mathcal{S}_{\text{hard}}$ can be calculated by

$$b(q_i) = t_0 + (k-1) \cdot d, \quad where\ q_i \in \mathcal{S}_{\text{hard}}. \quad (10)$$

Here $k$ is the number of the current iteration, $t_0$ is the initial token budget, and $d$ denotes the length increment. Then we evaluate the accuracy of samples from $\mathcal{S}_{\text{hard}}$ under the new budget and move some examples into $\mathcal{S}_{\text{learning}}$ according to Eq.8 and Eq.9. Notably, this evaluation introduces only minimal additional time because the traces for evaluation will be reused for optimization. In this way, easy examples will be added to the learning and mastered pools, costing only a small budget, and the more difficult examples will receive a larger budget. During inference, ABPO behaves like standard models without additional mechanisms. The algorithm of ABPO is given in Appendix A.4.

**Discussion:** ABPO follows the principle of *curriculum learning* to cultivate CoT reasoning of LLMs from simple to difficult. Besides, its dynamic and adaptive example scheduling can also help the LLMs learn CoT examples of appropriate difficulty with a proper token budget, which can not only improve the CoT efficiency but also alleviate the instability issues associated with RL training (Yu et al., 2025). Moreover, ABPO is also robust to its hyper-parameter selections, as shown in our Appendix A.5.

# 4 Experiments

## 4.1 Dataset and Metric

We conduct experiments on four core benchmarks, including MATH500 (Hendrycks et al., 2021), AIME (MAA, 2024), AMC (MAA, 2023), and Olympiad-Bench (He et al., 2024). Among these benchmarks, MATH500 is often regarded as the easiest one, while AIME is regarded as the most difficult one. Additionally, evaluations on out-of-domain (OOD) tasks are detailed in Appendix A.2. Following the previous works (Arora and Zanette, 2025), the accuracy of an example is defined by the

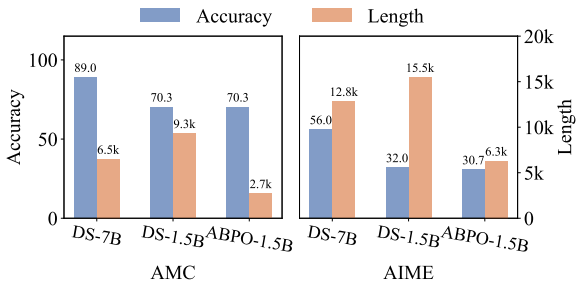


Figure 3: CoT length and accuracy on AMC and AIME. Regardless of the task difficulty, existing LLMs are prone to generating excessive CoT tokens, leading to the issue of *over-thinking*.

average performance across $K$ runs. During the evaluation, we set $K = 3$ for MATH500, $K = 5$ for Olympiad-Bench and $K = 10$ for both AIME and AMC. *Accuracy–Efficiency Score* **(AES)** (Luo et al., 2025a) is to evaluate the trade-off between accuracy and computation. The detailed formulation of AES is provided in Appendix A.1.

## 4.2 Implementation details

The main experiments are conducted on $2.4k$ training samples from the *AIME-AMC* subset of the Prime dataset (Cui et al., 2025). To further validate the effectiveness of ABPO, we additionally train DeepSeek-R1-Distill-Qwen-1.5B (hereafter referred to as DeepSeek-Distill-1.5B) on $32k$ samples from the AM dataset (Zhao et al., 2025). The detailed hyperparameters of ABPO are consistently applied across all models, which are set as follows: the initial budget $t_0 = 2000$, the budget increment $d = 500$, the difficulty thresholds $\tau_0 = 0.2$ and $\tau_1 = 0.85$, the learning pool update ratio $\lambda = 0.5$, and the review rate $\alpha = 0.1$. All experiments are conducted on 8 NVIDIA A800 GPUs.

## 4.3 Quantitative Analysis

**The issue of over-thinking.** Fig. 3 shows the average token lengths and accuracies of two LLMs on AMC and AIME, namely DeepSeek-Distill-7B and 1.5B. It can be seen that on the easy AMC benchmark, DeepSeek-Distill-7B are still prone to generating excessively long CoT traces, and the token costs become more expensive on the more difficult AIME benchmark. Moreover, this case will become more prominent on the smaller DeepSeek-Distill-1.5B. To explain, with inferior base model capability, smaller LLMs require more token length to scale up CoT reasoning, meeting the law of *test time scaling* (Snell et al., 2024). These results well confirm our motivation that *over-thinking* is a key

Table 1: Comparison between fixed-budget baselines and ABPO on DeepSeek-R1-Distill-Qwen-1.5B. "*Acc.*", "*Len.*" and "*AES*" denote accuracy (*higher is better*), length (*lower is better*) and Accuracy–Efficiency Score (*higher is better*), respectively. The best and second best results are marked in **bold** and underline, respectively.

| | AMC | | | AIME | | | **Average** | | | Training Time (h) |
|---|---|---|---|---|---|---|---|---|---|---|
| | Acc. ↑ | Len.↓ | AES | Acc.↑ | Len.↓ | AES | Acc. ↑ | Len.↓ | AES | |
| Base | 70.3 | 9314 | – | 32.0 | 15476 | – | 51.1 | 12395 | – | – |
| Fixed-Budget 2000 | 66.3 | **2320** | 0.47 | 24.0 | **5576** | -0.61 | 45.1 | **3948** | 0.09 | **15.5** |
| Fixed-Budget 3000 | 68.8 | 2850 | 0.59 | 23.3 | 6309 | -0.76 | 46.0 | 4980 | 0.13 | 22.2 |
| Fixed-Budget 4000 | **71.8** | 3600 | 0.68 | 25.7 | 6376 | -0.40 | 48.7 | 4988 | 0.36 | 28.0 |
| **ABPO** (Ours) | 70.3 | 2715 | **0.71** | **30.7** | 6290 | **0.39** | **50.5** | 4502 | **0.54** | 21.3 |

Table 2: Comparison with existing efficient CoT methods on three LLMs and four mathematical benchmarks.

| | Train Data | MATH500 | | | AMC | | | Olympiad | | | AIME | | | **Average** | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | Acc. ↑ | Len. ↓ | AES ↑ | Acc. ↑ | Len. ↓ | AES ↑ | Acc. ↑ | Len. ↓ | AES ↑ | Acc. ↑ | Len. ↓ | AES ↑ | Acc. ↑ | Len. ↓ | AES ↑ |
| *Base model: DeepSeek-R1-Distill-Qwen-1.5B* | | | | | | | | | | | | | | | | |
| Baseline | – | 83.3 | 5278 | – | 70.3 | 9314 | – | 50.3 | 10820 | – | 32.0 | 15476 | – | 59.0 | 10222 | – |
| ShorterBetter | 40$k$ | 68.9 | **864** | -0.03 | 48.3 | **1194** | -0.69 | 36.2 | **2072** | -0.60 | 19.7 | 4286 | -1.20 | 43.2 | **2104** | -0.54 |
| TrainingEfficiently | 3.2$k$ | 81.9 | 2536 | 0.44 | 70.3 | 5183 | 0.44 | 49.2 | 6818 | 0.27 | 29.0 | 10552 | -0.15 | 57.6 | 6272 | 0.27 |
| HAPO | 2$k$ | 81.5 | 2633 | 0.39 | 69.0 | 4507 | 0.43 | 49.2 | 5458 | 0.38 | 22.7 | 9268 | -1.06 | 55.6 | 5466 | 0.18 |
| **ABPO** (Ours) | 2.4$k$ | 81.3 | 1741 | 0.55 | 70.3 | 2715 | 0.71 | 50.1 | 3280 | 0.68 | **30.7** | 6290 | **0.39** | 58.1 | 3506 | 0.61 |
| **ABPO** (Ours) | 32$k$ | **85.2** | 1434 | **0.80** | **74.3** | 2068 | **0.95** | **52.7** | 2141 | **0.95** | 28.7 | **3223** | 0.28 | **60.2** | 2217 | **0.85** |
| *Base model: DeepSeek-R1-Distill-Qwen-7B* | | | | | | | | | | | | | | | | |
| Baseline | – | 93.2 | 4086 | – | 89.0 | 6484 | – | 66.1 | 8686 | – | 56.3 | 12837 | – | 76.1 | 8083 | – |
| ShorterBetter | 40$k$ | 85.6 | **1233** | 0.29 | 82.3 | **2443** | 0.24 | 56.7 | **2990** | -0.06 | 49.0 | **5673** | **-0.09** | 68.4 | **3084** | 0.11 |
| TrainingEfficiently | 3.2$k$ | 89.1 | 2121 | 0.26 | **87.5** | 4300 | 0.25 | 61.9 | 5626 | 0.04 | **52.3** | 9410 | **-0.09** | **72.7** | 5364 | 0.11 |
| **ABPO** (Ours) | 2.4$k$ | **91.5** | 1617 | **0.51** | 87.0 | 2983 | **0.40** | **63.5** | 4330 | **0.31** | 48.7 | 8258 | -0.32 | **72.7** | 4297 | **0.24** |
| *Base model: DeepScaleR-1.5B-Preview* | | | | | | | | | | | | | | | | |
| Baseline | – | 88.2 | 3155 | – | 81.3 | 5067 | – | 58.3 | 5718 | – | 40.0 | 9041 | – | 66.9 | 5745 | – |
| L1-Exact | 40$k$ | 82.1 | 3189 | -0.35 | 71.0 | 3067 | -0.24 | 50.7 | 3196 | -0.22 | 22.3 | 3715 | -1.62 | 56.5 | 3292 | -0.35 |
| L1-Max | 40$k$ | 84.5 | **1527** | **0.31** | 75.5 | **1768** | 0.30 | 51.1 | **1733** | 0.07 | 27.0 | **2290** | -0.88 | 59.5 | **1830** | 0.13 |
| HAPO | 2$k$ | 81.9 | 2537 | -0.16 | 70.5 | 4119 | -0.47 | 53.6 | 3976 | -0.10 | 31.3 | 6974 | -0.86 | 59.3 | 4401 | -0.33 |
| ThinkPrune | 2.4$k$ | 85.5 | 2029 | 0.21 | 78.5 | 3125 | 0.21 | **55.0** | 3532 | 0.10 | 34.7 | 5743 | -0.30 | 63.4 | 3607 | 0.11 |
| **ABPO** (Ours) | 2.4$k$ | **85.6** | 1874 | 0.26 | **79.8** | 2845 | **0.35** | **55.0** | 3302 | **0.14** | **37.3** | 5736 | **0.03** | **64.4** | 3439 | **0.21** |

issue that plagues the efficient use of LLMs.

**Comparison with fixed-budget baselines.** In Tab. 1, we first compare ABPO with the fixed-budget baselines. We can observe that although well reducing the cost of CoT tokens, these fixed-budget baselines still impose a negative impact on difficult examples. Specifically, the training with a budget of $3,000$ tokens on AIME reduces the output length by 59.2%, but leads to 27.2% performance drop. In comparison, ABPO requires 4.1% less training time and reduces the output length by 59.4%, while obtaining a 4.1% performance drop. Another observation is that for medium-difficulty examples, ABPO achieves a better balance between accuracy and length. Although its accuracy is close to the baseline of a higher fixed budget, the output length is much shorter. Overall, these results demonstrate that fixed-budget methods are hard to achieve a good trade-off between efficiency and performance, while our adaptive-budgeting based method can better handle this issue.

**Comparison with efficient CoT methods.** From Tab. 2, we can first observe that ABPO and existing efficient CoT methods can well reduce token costs. However, the compared methods still have significant performance compromises, *e.g.*, -38.4% and -29.1% by *ShorterBetter* and *HAPO* on the difficult AIME. In addition, the efficiency benefits of some methods become marginal on the difficult tasks. For instance, *TrainingEfficiently* requires about 10$k$ and 6.8$k$ token budgets on AIME and Olympiad for DeepSeek-Distill-1.5B, respectively, resulting in worse AES. In terms of fixed-budget methods, *e.g.*, *ThinkPrune*, they can maintain the short CoT outputs, but obviously drops the performance on the most difficult AIME. In contrast, ABPO can achieve a good balance between efficiency and performance. These results well confirm the effectiveness of our ABPO.

In Fig. 4, we further compare ABPO with L1-Max and DeepScaleR-1.5B-Preview under different token budgets during inference. From this fig-

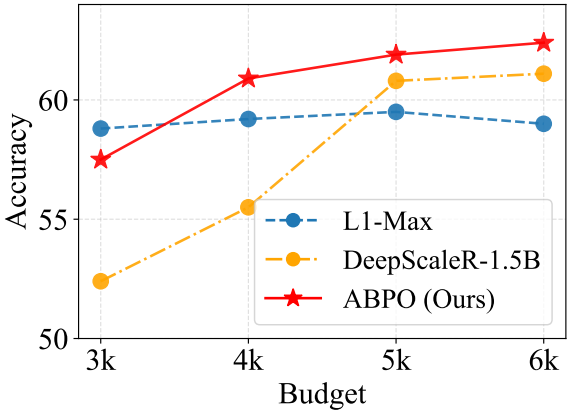


Figure 4: Comparison between ABPO, DeepScaleR-1.5B-Preview and L1-Max under different token budgets during inference. The accuracy is averaged over AMC, MATH500, AIME and Olympiad.

ure, we can first observe that L1-Max performs best under the low-budget settings, *e.g.*, $3k$. To explain, L1-Max is trained with a prompt that controls the output length uniformly between 100 and 4000 tokens without considering question difficulty. As a result, it tends to answer under lower budgets, which is more close to its training objective. However, as the token budget increases, *e.g.*, $4k$-$6k$, the advantages of ABPO become more obvious compared to L1-Max. Moreover, the performance gains of L1-Max become very limited after the $4k$ budget. In contrast, ABPO demonstrates consistent performance improvements as the budget increases. In terms of DeepScaleR-1.5B, its capability is greatly constrained by the maximum token budgets. Furthermore, our method also achieves better performance than the soft reward desgin, and detailed in A.9. Overall, these results demonstrate that the proposed ABPO achieves a better balance between efficiency and effectiveness.

**Ablation study** In Tab. 3, we also examine the roles of three example pools in ABPO. Here, $Mix(\mathcal{S}_{learning}, \mathcal{S}_{hard})$ combines the examples of $\mathcal{S}_{learning}$ with $\mathcal{S}_{hard}$ for sampling, *i.e.*, not considering the difficulty of examples for training. $Mix(\mathcal{S}_{learning}, \mathcal{S}_{mastered})$ denotes the mixing of $\mathcal{S}_{learning}$ and $\mathcal{S}_{mastered}$, *i.e.,* all examples are either under-learning or too hard for training. In contrast, ABPO will assess examples via three types while reviewing partial examples from $S_{mastered}$, as described in Sec. 3.2. From these results, we can observe that $Mix(\mathcal{S}_{learning}, \mathcal{S}_{hard})$ significantly reduces the CoT length of the LLM, *e.g.*, -65.4% on AIME. However, under this scheme, the longer CoTs for difficult examples always give the LLM a penalty signal, causing the LLM tend to output shorter CoT. On the other hand, $Mix(\mathcal{S}_{learning}, \mathcal{S}_{mastered})$ significantly reduces the CoT length while obtaining better performance. For example, it reduces the inference length by 55.1% with 2.1% performance drop on AMC. To explain, after identifying the hard examples, this scheme can help the LLM achieve better optimizations during training. However, its performance is still limited due to the excessive use of easy examples, especially on the difficult ones. In contrast, via categorizing examples into three pools with a review design, ABPO can help the LLM better compensate the performance drops while keeping low CoT budgets. Overall, these ablation studies well confirm the designs of the example pools and the review operation in ABPO.

To verify the robustness of ABPO, we conduct sensitivity studies on four key hyper-parameters: $\tau_0$, $\tau_1$, $d$, and $t_0$ in Fig. 5. As described in Sec. 3.2, $\tau_0$ and $\tau_1$ denote the thresholds of moving the hard and moderate examples to the learning and mastered pools in ABPO, while $d$ refers to the budget increment when promoting samples from the hard pool, and $t_0$ represents the initial budget. As can be seen, ABPO is relatively more sensitive to $\tau_0$ than $\tau_1$. To explain, when $\tau_0$ is too small, *e.g.*, 0.1, ABPO will put more overly difficult examples to the learning pools, which are beyond the learning range of LLM under the low token budget. Similarly, when the threshold is set too high, LLM will fail to achieve the expected average accuracy of $\tau_1$, making it hard to learn more difficult examples.

In terms of $d$ and $t_0$, ABPO is more likely to be affected by the setting of the budget increment $d$. For instance, when $d$ is set to a small value, *e.g.*, 250, LLM will suffer obvious performance drop. To explain, an insufficient $d$ fails to allocate adequate budgets to the newly incoming hard examples, leading to inferior optimization. However, when given proper budgets, *e.g.*, 500 or larger, the performance is greatly improved, also suggesting the robustness of our ABPO. Besides, ablation studies confirm that curriculum learning, adaptive budgeting, and the review mechanism are all essential, as detailed in Appendix A.8.

### 4.4 Qualitative Analysis

To gain insights into the proposed ABPO, we visualize the prediction correctness and output length on both easy and hard examples, and compare them with DeepScaleR-1.5B and L1-Max, as shown in

Table 3: Comparison between ABPO and the alternative data scheduling strategies on DeepSeek-R1-Distill-Qwen-1.5B. $Mix(\cdot)$ denotes the mixing of different types of examples for sampling.

| | AMC | | | AIME | | | **Average** | | |
|---|---|---|---|---|---|---|---|---|---|
| | Acc.↑ | Len.↓ | AES↑ | Acc.↑ | Len.↓ | AES↑ | Acc.↑ | Len.↓ | AES↑ |
| Base | 70.3 | 9314 | – | 32.0 | 15476 | – | 51.1 | 12395 | – |
| $Mix(\mathcal{S}_{learning}, \mathcal{S}_{hard})$ | 65.8 | **2515** | 0.41 | 23.7 | **5353** | -0.64 | 44.8 | **3934** | 0.06 |
| $Mix(\mathcal{S}_{learning}, \mathcal{S}_{mastered})$ | 68.8 | 2937 | 0.58 | 29.0 | 6956 | 0.08 | 48.9 | 4946 | 0.38 |
| **ABPO** (Ours) | **70.3** | 2715 | **0.71** | **30.7** | 6290 | **0.39** | **50.5** | 4502 | **0.58** |

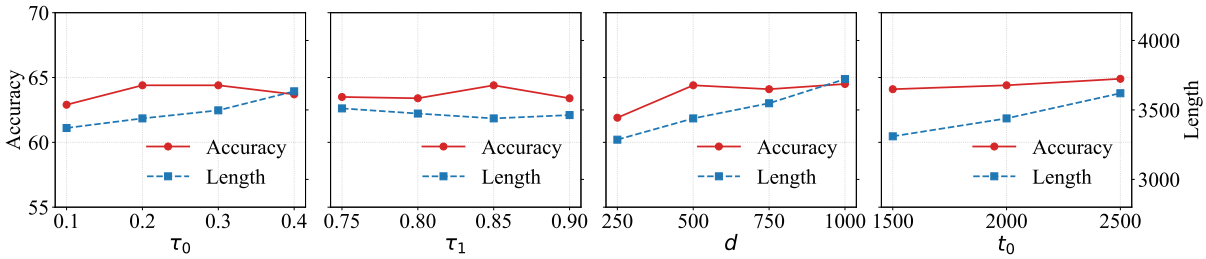


Figure 5: Hyper-parameter selection of ABPO. We report the average accuracy and length over AMC, MATH500, AIME and Olympiad under the different settings of the difficulty thresholds $(\tau_0, \tau_1)$, budget increment $(d)$, and the initial token budget $(t_0)$. In most cases, ABPO is robust to hyperparameter settings, with minimal variation in accuracy and length.

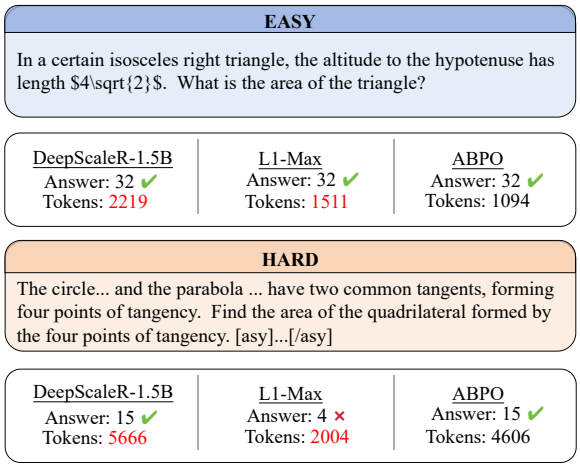


Figure 6: Visualization of prediction correctness and output length on both easy and hard examples. These examples show that ABPO can help the LLM generate more concise CoTs for simple questions and longer CoTs for the hard ones, thereby balancing the efficiency and performance of CoT reasoning.

Fig. 6. The details of these examples can be found in Appendix A.10. From these statistics, we can observe that ABPO produces CoTs with much fewer tokens on easy examples, helping the LLM quickly arrive at the correct answer. In contrast, the default DeepScaleR-1.5B tends to consume excessive tokens even for easy examples, leading to inefficient CoT reasoning. For more difficult examples, ABPO adaptively increases the token usage to ensure the reasoning depth required for the correct answer. Meanwhile, for the fixed-budget method L1-Max, the reasoning process for the harder examples is too tight to correctly answer the question. In other words, producing shorter CoTs will fail to contain the necessary reasoning steps for complex problems. This case demonstrates that ABPO is capable of adaptively performing test time scaling according to example difficulty, validating its ability to achieve difficulty-aware and efficient reasoning. Overall, these findings demonstrate the effectiveness of ABPO in adjusting token budgets according to example difficulty and its adaptive curriculum learning principle.

## 5 Conclusion

In this paper, we propose a novel reinforcement learning scheme termed ***A**daptive-**B**udgeting based **P**olicy **O**ptimization* (**ABPO**) to achieve the optimal balance between the efficiency and performance of CoT reasoning. ABPO introduces dynamic example pools to categorize training examples into three types based on the model's status, and then adopts an adaptive scheduling mechanism to assign proper token budgets for different examples. Extensive experiments on three representative LLMs and a bunch of CoT reasoning benchmarks demonstrate that our ABPO method can achieve substantial efficiency improvements, and also show obvious advantages over the compared methods in balancing the efficiency and performance. These results well confirm the effectiveness of ABPO towards efficient CoT reasoning.

## Limitations

Despite the well-motivated designs and strong empirical performance, the proposed ABPO has two main limitations. *i.* ABPO involves several hyperparameters, which may increase the difficulty of training. However, as shown in the paper and Appendix A.5, ABPO is robust to their settings at most cases, and the optimal selection of hyperparameters is empirically easy to set. *ii.* Similar to the compared methods, our ABPO is mainly validated on the mathematical tasks, which reflects the main functionality of CoT reasoning. Other types of tasks can be attempted in our near future work.

## References


Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, and 1 others. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

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

Samah AlKhuzaey, Floriana Grasso, Terry R Payne, and Valentina Tamma. 2024. Text-based question difficulty prediction: A systematic review of automatic approaches. *International Journal of Artificial Intelligence in Education*, 34(3):862–914.

Daman Arora and Andrea Zanette. 2025. Training language models to reason efficiently. *Preprint*, arXiv:2502.04463.

Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, and 1 others. 2023. Qwen technical report. *arXiv preprint arXiv:2309.16609*.

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. 2009. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pages 41–48.

Xingyu Chen, Jiahao Xu, Tian Liang, Zhiwei He, Jianhui Pang, Dian Yu, Linfeng Song, Qiuzhi Liu, Mengfei Zhou, Zhuosheng Zhang, Rui Wang, Zhaopeng Tu, Haitao Mi, and Dong Yu. 2025. Do not think that much for 2+3=? on the overthinking of o1-like llms. *Preprint*, arXiv:2412.21187.

Alejandro Cuadron, Dacheng Li, Wenjie Ma, Xingyao Wang, Yichuan Wang, Siyuan Zhuang, Shu Liu, Luis Gaspar Schroeder, Tian Xia, Huanzhi Mao, and 1 others. 2025. The danger of overthinking: Examining the reasoning-action dilemma in agentic tasks. *arXiv preprint arXiv:2502.08235*.

Ganqu Cui, Lifan Yuan, Zefan Wang, Hanbin Wang, Wendi Li, Bingxiang He, Yuchen Fan, Tianyu Yu, Qixin Xu, Weize Chen, and 1 others. 2025. Process reinforcement through implicit rewards. *arXiv preprint arXiv:2502.01456*.

DeepSeek-AI. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *Preprint*, arXiv:2501.12948.

Razvan-Gabriel Dumitru, Darius Peteleaza, Vikas Yadav, and Liangming Pan. 2025. ConciseRL: Conciseness-guided reinforcement learning for efficient reasoning models. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 17099–17123, Suzhou, China. Association for Computational Linguistics.

Bofei Gao, Feifan Song, Zhe Yang, Zefan Cai, Yibo Miao, Qingxiu Dong, Lei Li, Chenghao Ma, Liang Chen, Runxin Xu, Zhengyang Tang, Benyou Wang, Daoguang Zan, Shanghaoran Quan, Ge Zhang, Lei Sha, Yichang Zhang, Xuancheng Ren, Tianyu Liu, and Baobao Chang. 2024. Omni-math: A universal olympiad level mathematic benchmark for large language models. *Preprint*, arXiv:2410.07985.

Mor Geva, Daniel Khashabi, Elad Segal, Tushar Khot, Dan Roth, and Jonathan Berant. 2021. Did Aristotle Use a Laptop? A Question Answering Benchmark with Implicit Reasoning Strategies. *Transactions of the Association for Computational Linguistics (TACL)*.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, and 1 others. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Hasan Abed Al Kader Hammoud, Kumail Alhamoud, Abed Hammoud, Elie Bou-Zeid, Marzyeh Ghassemi, and Bernard Ghanem. 2025. Train long, think short: Curriculum learning for efficient reasoning. *Preprint*, arXiv:2508.08940.

Chaoqun He, Renjie Luo, Yuzhuo Bai, Shengding Hu, Zhen Leng Thai, Junhao Shen, Jinyi Hu, Xu Han, Yujie Huang, Yuxiang Zhang, and 1 others. 2024. Olympiadbench: A challenging benchmark for promoting agi with olympiad-level bilingual multimodal scientific problems. *arXiv preprint arXiv:2402.14008*.

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. *arXiv preprint arXiv:2103.03874*.

Bairu Hou, Yang Zhang, Jiabao Ji, Yujian Liu, Kaizhi Qian, Jacob Andreas, and Shiyu Chang. 2025. Thinkprune: Pruning long chain-of-thought of llms via reinforcement learning. *Preprint*, arXiv:2504.01296.

Chengyu Huang, Zhengxin Zhang, and Claire Cardie. 2025. Hapo: Training language models to reason concisely via history-aware policy optimization. *Preprint*, arXiv:2505.11225.

Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. 2024. Livecodebench: Holistic and contamination free evaluation of large language models for code. *Preprint*, arXiv:2403.07974.

Zichen Liu, Changyu Chen, Wenjun Li, Penghui Qi, Tianyu Pang, Chao Du, Wee Sun Lee, and Min Lin. 2025. Understanding r1-zero-like training: A critical perspective. In *Conference on Language Modeling (COLM)*.

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

Michael Luo, Sijun Tan, Justin Wong, Xiaoxiang Shi, William Y. Tang, Manan Roongta, Colin Cai, Jeffrey Luo, Tianjun Zhang, Li Erran Li, Raluca Ada Popa, and Ion Stoica. 2025b. Deepscaler: Surpassing o1-preview with a 1.5b model by scaling rl.

Xinyin Ma, Guangnian Wan, Runpeng Yu, Gongfan Fang, and Xinchao Wang. 2025. Cot-valve: Length-compressible chain-of-thought tuning. *Preprint*, arXiv:2502.09601.

MAA. 2023. American mathematics competitions - amc. `https://maa.org/student-programs/amc/`.

MAA. 2024. American invitational mathematics examination - aime.

Yingqian Min, Zhipeng Chen, Jinhao Jiang, Jie Chen, Jia Deng, Yiwen Hu, Yiru Tang, Jiapeng Wang, Xiaoxue Cheng, Huatong Song, Wayne Xin Zhao, Zheng Liu, Zhongyuan Wang, and Ji-Rong Wen. 2024. Imitate, explore, and self-improve: A reproduction report on slow-thinking reasoning systems. *Preprint*, arXiv:2412.09413.

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. *Preprint*, arXiv:2501.19393.

OpenAI, :, Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, Alex Iftimie, Alex Karpenko, Alex Tachard Passos, Alexander Neitz, Alexander Prokofiev, Alexander Wei, Allison Tam, and 244 others. 2024. Openai o1 system card. *Preprint*, arXiv:2412.16720.

Penghui Qi, Zichen Liu, Tianyu Pang, Chao Du, Wee Sun Lee, and Min Lin. 2025. Optimizing anytime reasoning via budget relative policy optimization. *Preprint*, arXiv:2505.13438.

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. *Preprint*, arXiv:2408.03314.

Jinyan Su and Claire Cardie. 2025. Thinking fast and right: Balancing accuracy and reasoning length with adaptive rewards. *Preprint*, arXiv:2505.18298.

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

Qwen Team. 2025. Qwq-32b-preview. *https://qwenlm.github.io/blog/qwq-32b-preview/*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, and 1 others. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.

Heming Xia, Chak Tou Leong, Wenjie Wang, Yongqi Li, and Wenjie Li. 2025. TokenSkip: Controllable chain-of-thought compression in LLMs. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 3351–3363, Suzhou, China. Association for Computational Linguistics.

Chulin Xie, Yangsibo Huang, Chiyuan Zhang, Da Yu, Xinyun Chen, Bill Yuchen Lin, Bo Li, Badih Ghazi, and Ravi Kumar. 2025. On memorization of large language models in logical reasoning. *Preprint*, arXiv:2410.23123.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, and 1 others. 2024. Qwen2. 5 technical report. *arXiv preprint arXiv:2412.15115*.

Edward Yeo, Yuxuan Tong, Morry Niu, Graham Neubig, and Xiang Yue. 2025. Demystifying long chain-of-thought reasoning in llms. *Preprint*, arXiv:2502.03373.

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

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, Xin Liu, Haibin Lin, Zhiqi Lin, Bole Ma, Guangming Sheng, Yuxuan Tong, Chi Zhang, Mofan Zhang, Wang Zhang, and 16 others. 2025. Dapo: An open-source llm reinforcement learning system at scale. *Preprint*, arXiv:2503.14476.

Xiaoyun Zhang, Jingqing Ruan, Xing Ma, Yawen Zhu, Haodong Zhao, Hao Li, Jiansong Chen, Ke Zeng, and Xunliang Cai. 2025. When to continue thinking: Adaptive thinking mode switching for efficient reasoning. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 5808–5828, Suzhou, China. Association for Computational Linguistics.

Han Zhao, Haotian Wang, Yiping Peng, Sitong Zhao, Xiaoyu Tian, Shuaiting Chen, Yunjie Ji, and Xiangang Li. 2025. 1.4 million open-source distilled reasoning dataset to empower large language model training. *Preprint*, arXiv:2503.19633.

# A Appendix

## A.1 Accuracy-Efficiency Score (AES)

To measure the trade-off between reasoning accuracy and efficiency, we utilize the *Accuracy-Efficiency Score* (AES) (Luo et al., 2025a), which jointly considers the relative changes in solution length and accuracy. Specifically, the relative length reduction $\Delta\text{Length}$ and the relative accuracy change $\Delta\text{Acc}$ are defined as follows:

$$\Delta\text{Length} = \frac{\text{Length}_{\text{baseline}} - \text{Length}_{\text{model}}}{\text{Length}_{\text{baseline}}} \quad (11)$$

$$\Delta\text{Acc} = \frac{\text{Acc}_{\text{model}} - \text{Acc}_{\text{baseline}}}{\text{Acc}_{\text{baseline}}} \quad (12)$$

Based on these relative changes, the AES is computed using a piecewise function to properly reward length reductions while penalizing accuracy drops:

$$\text{AES} = \begin{cases} \alpha \cdot \Delta\text{Length} + \beta \cdot |\Delta\text{Acc}|, & \text{if } \Delta\text{Acc} \geq 0 \\ \alpha \cdot \Delta\text{Length} - \gamma \cdot |\Delta\text{Acc}|, & \text{if } \Delta\text{Acc} < 0 \end{cases} \quad (13)$$

where $\alpha > 0$, $\beta > 0$, and $\gamma > 0$ are weighting coefficients. Following previous works (Yi et al., 2025; Luo et al., 2025a), we set $\alpha = 1$, $\beta = 3$, and $\gamma = 5$ by default in our experiments to emphasize the importance of maintaining performance while optimizing for efficiency.

## A.2 Generalization to Out-of-Domain Tasks

To verify the generalizability of ABPO beyond mathematical problem-solving, we extend our evaluation to out-of-domain (OOD) reasoning benchmarks. Specifically, we evaluate the models on StrategyQA (Geva et al., 2021) for logical and commonsense reasoning, and LiveCodeBench (Jain et al., 2024) for code generation tasks.

As shown in Tab. 4, we can first observe that ABPO still exhibits a strong balance between reasoning length and performance on out-of-domain (OOD) tasks. For example, on the LiveCodeBench dataset, ABPO reduces the reasoning length by 68.5% compared to the baseline, while maintaining and even slightly improving the performance. Moreover, ABPO still maintains distinct advantages over the compared methods. For instance, on the LiveCodeBench dataset, ABPO's reasoning length is 62.2% shorter than *Training Efficiently*, while its accuracy is 5.7% higher.

Overall, these results demonstrate that ABPO exhibits strong generalizability.

Table 4: Comparison of different efficient CoT methods on Out-of-Domain (OOD) benchmarks. "Acc." and "Len." denote accuracy and average token length, respectively.

| Method | StrategyQA | | LiveCodeBench | |
|---|---|---|---|---|
| | Acc. ↑ | Len. ↓ | Acc. ↑ | Len. ↓ |
| *Base model: DeepSeek-R1-Distill-Qwen-1.5B* | | | | |
| Baseline | 39.0 | 775 | 20.8 | 10091 |
| Training Efficiently | 44.6 | 484 | 20.9 | 8395 |
| ABPO (Ours) | **52.9** | **454** | **22.1** | **3174** |
| *Base model: DeepScaleR-1.5B-Preview* | | | | |
| Baseline | 47.2 | 749 | 24.6 | 7852 |
| HAPO | 49.9 | **416** | 23.8 | 6263 |
| ABPO (Ours) | **50.3** | 509 | **23.9** | **5953** |

Table 5: Comparison with TokenSkip using DeepScaleR-1.5B as the base model. "Acc." and "Len." denote accuracy and average token length, respectively.

| Method | GSM8K | | MATH500 | |
|---|---|---|---|---|
| | Acc. ↑ | Len. ↓ | Acc. ↑ | Len. ↓ |
| DeepScaleR-1.5B | 83.9 | 1381 | 88.2 | 3155 |
| TokenSkip | 67.2 | 895 | 53.3 | **1654** |
| ABPO (Ours) | **86.7** | **821** | **85.6** | 1874 |

### A.3 Comparison with SFT-based Efficient CoT Methods

To further evaluate the effectiveness of our approach, we compare ABPO with TokenSkip, a representative Supervised Fine-Tuning (SFT) based efficient CoT method that trains models on explicitly compressed reasoning sequences.

As shown in Tab. 5, we observe that while SFT-based methods utilizing compressed sequences can effectively reduce the reasoning length, they often suffer from severe performance degradation, whereas ABPO maintains a much stronger balance between performance and efficiency. For example, on the MATH500 dataset, TokenSkip achieves the shortest CoT length but causes a massive accuracy drop from 88.2% to 53.3%. In contrast, ABPO significantly reduces the length (from 3155 to 1874 tokens) while only experiencing a minor accuracy fluctuation (85.6%). Furthermore, on the GSM8K dataset, ABPO even surpasses the base model in accuracy (86.7% v.s. 83.9%) while achieving the shortest reasoning length (821 tokens), vastly outperforming TokenSkip in both metrics. In conclusion, compared to static SFT on compressed sequences, the dynamic curriculum learning paradigm of ABPO is significantly more effective at preserving the intrinsic reasoning capabilities of LLMs while compressing CoT lengths.

### A.4 Pseudo-code of ABPO

The pseudo code of ABPO is given in Algorithm 1.

**Algorithm 1:** Adaptive-Budgeting based Policy Optimization (ABPO)

**Input:** $D=\{(q_i, y_i)\}_{i=1}^{N}$
**Output:** LLM $M$ with compact CoT
**Init:**
$\mathcal{S}_{\text{hard}} \leftarrow D,\ \mathcal{S}_{\text{learning}} \leftarrow \emptyset,\ \mathcal{S}_{\text{mastered}} \leftarrow \emptyset;$
$\texttt{budget}[q] \leftarrow 0;\ t \leftarrow 0;\ k \leftarrow 1.$
**while** $t < S_{\max}$ ***and*** $\mathcal{S}_{mastered} \neq D$ **do**
  **foreach** $q \in \mathcal{S}_{hard}$ **do**
    $\texttt{budget}[q] \leftarrow t_0 + (k-1)d;$
    **if** $Acc(q;n) > \tau_0$ **then** move $q$ to tail of $\mathcal{S}_{\text{learning}}$ ;
  $L_0 \leftarrow |\mathcal{S}_{\text{learning}}|;$
  **while** $|\mathcal{S}_{learning}| \geq \lambda L_0$ ***and*** $t < S_{\max}$ **do**
    Sample batch $\mathcal{B}{:=}\text{mix}(\mathcal{S}_{\text{mastered}}, \mathcal{S}_{\text{learning}}; \alpha)$ ;
    GRPO_UPDATE$(M, \mathcal{B}, n)$;
    **foreach** $q \in \mathcal{B}$ **do**
      **if** $Acc(q;n) \geq \tau_1$ **then** move $q$ to tail of $\mathcal{S}_{\text{mastered}}$;
      **else** move $q$ to tail of $\mathcal{S}_{\text{learning}}$;
    $t \leftarrow t+1;$
  $k \leftarrow k+1;$

### A.5 Hyper-parameter Sensitivity Analysis

We conducted quick yet comprehensive sensitivity studies for these hyper-parameters. All results are reported in Tables 6–11, where the default settings used in our main experiments are marked with †.

**Thresholds $\tau_0$ and $\tau_1$.** $\tau_0$ and $\tau_1$ are two thresholds used for moving the hard and moderate examples to the learning and mastered pools in ABPO, respectively. Form these Tab. 6 and Tab. 7, we can first see that ABPO is robust to these two hyper-parameters in most cases. The only exception is the most difficult benchmark AIME, where most existing methods perform poorly.

Specifically, $\tau_0$ is more sensitive on this benchmark than $\tau_1$. To explain, when $\tau_0$ is too small, *e.g.*, 0.1, ABPO will put more overly difficult examples to the learning pools, which are beyond the learning range of LLM under the low token budget

Table 6: Sensitivity to $\tau_0$ (default marked with †). $\tau_0$ is the lower accuracy threshold in ABPO for promoting samples from the hard pool to the learning pool. "Acc." and "Len." denote accuracy and average token length, respectively.

| $\tau_0$ | MATH500 | | AIME | | AMC | | Olympiad | | Avg | |
|---|---|---|---|---|---|---|---|---|---|---|
| | Acc. ↑ | Len. ↓ | Acc. ↑ | Len. ↓ | Acc. ↑ | Len. ↓ | Acc. ↑ | Len. ↓ | Acc. ↑ | Len. ↓ |
| 0.10 | 86.0 | **1813** | 32.7 | **5645** | 78.3 | **2778** | 54.7 | **3245** | 62.9 | **3370** |
| **0.20**† | 85.6 | 1874 | **37.3** | 5736 | 79.8 | 2845 | 55.0 | 3302 | **64.4** | 3439 |
| 0.30 | 85.6 | 1893 | 35.7 | 5697 | **80.5** | 3096 | **55.7** | 3303 | 64.4 | 3497 |
| 0.40 | **87.0** | 1945 | 33.3 | 6109 | 79.2 | 3078 | 55.0 | 3459 | 63.7 | 3634 |

thereby impeding optimization. Similarly, when the threshold is set too high, LLM will consistently fail to achieve the expected average accuracy of $\tau_0$, making it unable to learn more difficult examples.

**Budget increment $d$ and initial budget $t_0$.** $d$ controls the increment of token budgets when re-evaluating and promoting samples from the hard pool, and $t_0$ is the initial budget as described in Eq. 10.

Tab. 8 and Tab. 9 show that ABPO is less effected by $d$ and $t_0$ in terms of *avg.* accuracy, but $t_0$ will slightly affect the final length of CoTs, but still marginal. When $d$ is set to a small value, *e.g.*, 250, LLM will suffer obvious performance drop, *e.g.*, 61.9 *avg.* accuracy. To explain, when $d$ is too small, ABPO will allocate insufficient budgets to the newly income hard examples, also leading to inferior optimization. However, when given proper budgets, *e.g.*, 500 or larger, the performance of LLM is greatly improved, also suggesting the robustness of our ABPO.

**Learning pool ratio $\lambda$ and review rate $\alpha$.** $\lambda$ denotes the proportion of the learning examples, which also determine the update of both learning and hard pools. As shown in Tab. 10, its effects to ABPO is limited. $\alpha$ denotes the review rate of the mastered examples, where a smaller value is more beneficial to the final training results, but it is still not very sensitive, as shown in Tab. 11.

Overall, the careful designs of ABPO makes it robust to these hyper-parameters in most cases. And the exceptions are akin to the motivation of ABPO and have be well tackled by our previous quick validations.

### A.6 Generalization to Different Model Families

Our method is not limited to a specific model family and can be seamlessly applied to different architectures. In the main experiments, we primarily evaluated the Qwen family because most prior works on efficient CoT reasoning are based on Qwen, which enables fair and direct comparisons with existing methods.

To further validate the generalizability of Adaptive-Budgeting based Policy Optimization (ABPO), we additionally conducted experiments on the LLaMA family. Specifically, we applied our method to DeepSeek-Distill-Llama-8B and compared it against the baseline and a fixed-budget approach. The results across four mathematical benchmarks are presented in Table 12.

From Table 12, we observe that ABPO consistently achieves comparable accuracy to the baseline while significantly reducing reasoning length across all tasks. For instance, ABPO reduces the average reasoning length of DeepSeek-Distill-Llama-8B by 60.6% (from 8029 to 3161) while maintaining a highly competitive average accuracy of 68.9%. Furthermore, compared to the fixed-budget baseline, ABPO achieves superior efficiency without compromising performance. These results demonstrate that the core adaptive budgeting mechanism of ABPO generalizes well beyond the Qwen family to other prevailing architectures.

### A.7 Effect of Rollouts in Evaluation

In this section, we analyze the effect of the number of rollouts $K$ used during evaluation. In the main experiments, we adopt $K = (3, 5, 10)$ rollouts following common practice in recent efficient CoT training works. For example, *Training Efficiently* (Arora and Zanette, 2025) evaluates MATH500 with $K = 3$ and AIME with $K = 10$ rollouts. Similarly, HAPO (Huang et al., 2025) uses $K = 4$ rollouts on MATH500, while Shorter-Better (Yi et al., 2025) evaluates all datasets with $K = 1$. To ensure fair comparisons, all methods in our main results are evaluated under the same rollout settings.

We further report results under a larger rollout

Table 7: Sensitivity to $\tau_1$ (default marked with †). $\tau_1$ is the upper accuracy threshold in ABPO for promoting samples from the learning pool to the mastered pool. "Acc." and "Len." denote accuracy and average token length, respectively.

| $\tau_1$ | MATH500 | | AIME | | AMC | | Olympiad | | Avg | |
|---|---|---|---|---|---|---|---|---|---|---|
| | Acc. ↑ | Len. ↓ | Acc. ↑ | Len. ↓ | Acc. ↑ | Len. ↓ | Acc. ↑ | Len. ↓ | Acc. ↑ | Len. ↓ |
| 0.90 | **86.8** | **1837** | 33.3 | **5587** | 78.0 | 3133 | **55.5** | **3295** | 63.4 | 3463 |
| **0.85**† | 85.6 | 1874 | **37.3** | 5736 | **79.8** | **2845** | 55.0 | 3302 | **64.4** | **3439** |
| 0.80 | 86.3 | 1898 | 35.0 | 5628 | 77.3 | 2949 | 55.2 | 3304 | 63.4 | 3474 |
| 0.75 | 86.3 | 1899 | 35.0 | 5704 | 77.0 | 3048 | **55.5** | 3393 | 63.5 | 3511 |

Table 8: Sensitivity to budget increment $d$ (default marked with †). $d$ controls the increase of token budgets when re-evaluating and promoting samples from the hard pool. "Acc." and "Len." denote accuracy and average token length, respectively.

| $d$ | MATH500 | | AIME | | AMC | | Olympiad | | Avg | |
|---|---|---|---|---|---|---|---|---|---|---|
| | Acc. ↑ | Len. ↓ | Acc. ↑ | Len. ↓ | Acc. ↑ | Len. ↓ | Acc. ↑ | Len. ↓ | Acc. ↑ | Len. ↓ |
| 250 | 85.8 | **1785** | 29.7 | **5325** | 76.3 | **2747** | 55.8 | **3088** | 61.9 | **3286** |
| **500**† | 85.6 | 1874 | **37.3** | 5736 | **79.8** | 2845 | 55.0 | 3302 | 64.4 | 3439 |
| 750 | 85.6 | 1942 | 36.7 | 5505 | 78.0 | 3037 | 55.4 | 3365 | 64.1 | 3548 |
| 1000 | **86.2** | 2012 | 36.7 | 6158 | 79.0 | 3165 | **55.9** | 3552 | **64.5** | 3722 |

setting of $K = 16$ on DeepScaleR-1.5B-Preview to examine whether increasing the number of rollouts affects the relative performance of different methods. Table 13 summarizes the comparison between $K = 16$ and the default setting used in our paper (denoted by †).

Across different rollout settings, similar observations and conclusions can be drawn. For instance, our ABPO still achieve the best *avg.* accuracy and AES compared to other efficient CoT methods. Besides, ABPO can still allocate proper token budgets for different tasks of varying difficulties, *e.g.*, 1874 for MATH500 and 3302 for Olympiad. Moreover, the performance gaps of ABPO between two settings are also marginal, *e.g.*, 64.4 v.s. 64.3 *avg.* accuracy and 3439 v.s. 3434 *avg.* length.

### A.8 Ablation of ABPO Components

In Tab. 14, we ablate the key designs of ABPO on DeepSeek-R1-Distill-Qwen-1.5B. In this table, "+ *Curriculum Learning*" denotes the training with curriculum scheduling, and the training batches are constructed by the examples from $\mathcal{S}_{\text{learning}}$ with CoT length limited by 4,000 tokens. "+ *Adaptive Budget*" further introduces dynamic budget scheduling, where their initial budget is set to $t_0$=2,000 and gradually expanded with $d$=500 per iteration. Finally, "+ *Review*" corresponds to the full settings of ABPO, which additionally adds the review mechanism by mixing a fraction of the mastered samples. From Tab. 14, we can first observe that using curriculum learning can reduce the CoT length substantially. For instance, on AMC benchmark,"+ *Curriculum Learning*" reduces the reasoning trace by 64.2% with 1.1% performance drop. After adding adaptive budgeting, *i.e.*, "+ *Adaptive Budget*", the output length is further shortened while improving accuracy. For instance, the length of CoT is shorten by 58.8%, while the performance drop is reduced to only 7.7% on AIME benchmark. This improvement demonstrates that allocating distinct token budgets according to examples difficulty (*e.g.*, assigning lower budgets to simple examples and higher budgets to hard ones) effectively drives the model to further reduce its overall reasoning length. With "+ *Review*", ABPO achieves the best trade-off, and it restores the accuracy close to the original LLMs while significantly reducing CoT token length. For instance, on AMC, the proposed ABPO method not only reduces the CoT length by 70.9%, but also keep the same performance. These results well confirm the designs of the adaptive budget assignments and the example review.

### A.9 Analysis of Reward Design

In this section, we analyze the effect of reward design in ABPO by comparing our default hard budget-aware reward with a soft length penalty based on L1-Max (Aggarwal and Welleck, 2025).

Tab. 15 reports the results on DeepScaleR-1.5B-

Table 9: Sensitivity to initial token budget $t_0$ (default marked with †). $t_0$ denotes the initial token budget assigned to all samples at the start of training. "Acc." and "Len." denote accuracy and average token length, respectively.

| $t_0$ | MATH500 | | AIME | | AMC | | Olympiad | | Avg | |
|---|---|---|---|---|---|---|---|---|---|---|
| | Acc. ↑ | Len. ↓ | Acc. ↑ | Len. ↓ | Acc. ↑ | Len. ↓ | Acc. ↑ | Len. ↓ | Acc. ↑ | Len. ↓ |
| 1500 | 85.9 | **1787** | 36.3 | **5351** | 78.8 | 2902 | 55.4 | **3199** | 64.1 | **3310** |
| **2000**† | 85.6 | 1874 | **37.3** | 5736 | 79.8 | **2845** | 55.0 | 3302 | 64.4 | 3439 |
| 2500 | **87.0** | 1966 | 36.0 | 5928 | **80.5** | 3062 | **56.1** | 3522 | **64.9** | 3620 |

Table 10: Sensitivity to learning pool ratio $\lambda$ (default marked with †). $\lambda$ specifies the proportion of remaining learning samples that triggers budget and pool updates. "Acc." and "Len." denote accuracy and average token length, respectively.

| $\lambda$ | MATH500 | | AIME | | AMC | | Olympiad | | Avg | |
|---|---|---|---|---|---|---|---|---|---|---|
| | Acc. ↑ | Len. ↓ | Acc. ↑ | Len. ↓ | Acc. ↑ | Len. ↓ | Acc. ↑ | Len. ↓ | Acc. ↑ | Len. ↓ |
| 0.6 | 85.7 | 1857 | 35.7 | 5612 | 79.3 | 2914 | 55.9 | 3340 | 64.2 | 3431 |
| **0.5**† | 85.6 | 1874 | **37.3** | 5736 | 79.8 | **2845** | 55.0 | 3302 | 64.4 | 3439 |
| 0.4 | **86.2** | **1806** | 34.3 | **5538** | 80.8 | 2855 | 55.3 | 3238 | 64.2 | **3359** |
| 0.3 | 86.1 | 1881 | 36.0 | 5588 | **81.5** | 3029 | **56.0** | **3232** | **64.9** | 3433 |

Preview. Compared with the soft-penalty variant, the default ABPO achieves a 5.6% improvement in average accuracy while only increasing 348 *avg.* length. Notably, on the most difficult task AIME, our performance gain can be up to 19.2%, and the *avg.* CoT length is only 5736, about 63% of baseline. In terms of these superior results, our analysis is as follow:

- Our curriculum learning designs in ABPO well consider the difficulties of examples to LLM, *i.e.*, the construction and dynamic scheduling of the mastered, learning and hard pools, thus the entire training process is in-fact not a hard-cut optimization but a smooth and progressive control over reasoning length.

- Under the curriculum learning scheme, the learning examples are within a suitable range for the LLM's difficulty and the token budget, e.g., about half of CoTs in a group are correct. In this case, it is more important to enhance the penalty signals for the incorrect CoTs, and using the soft penalty will instead make the rewards hard to distinguish, thereby undermining the optimization.

- Moreover, compared to the default reward of our ABPO, using the soft reward function will include inappropriated hard examples to the learning pools. To explain, the use of soft reward will obviously increase the *avg.* reward value of the learning examples. In this case, some hard examples may be added to the learning pool prematurely, which does not fully meet the principle of curriculum learning and impede the optimization instead.

Overall, our ABPO uses the curriculum learning principle to build a smooth and progressive training process for the control of CoT length, and our hard reward design is more suitable in this training shceme.

### A.10 Additional Case Studies of ABPO

We provide more detailed outputs on both easy and hard examples in Fig.7.

Table 11: Sensitivity to review rate $\alpha$ (default marked with †). $\alpha$ denotes the sampling rate of mastered samples for review during training. "Acc." and "Len." denote accuracy and average token length, respectively.

| $\alpha$ | MATH500 | | AIME | | AMC | | Olympiad | | Avg | |
|---|---|---|---|---|---|---|---|---|---|---|
| | Acc. ↑ | Len. ↓ | Acc. ↑ | Len. ↓ | Acc. ↑ | Len. ↓ | Acc. ↑ | Len. ↓ | Acc. ↑ | Len. ↓ |
| **0.1**† | 85.6 | **1874** | **37.3** | 5736 | **79.8** | **2845** | 55.0 | **3302** | **64.4** | **3439** |
| 0.2 | **86.9** | 1931 | 32.7 | 5718 | 79.3 | 2983 | **56.0** | 3363 | 63.7 | 3499 |
| 0.3 | 86.3 | 1938 | 35.7 | **5625** | 78.3 | 2905 | 55.2 | 3337 | 63.9 | 3451 |

Table 12: Comparison of different efficient CoT methods on the LLaMA model family. We use DeepSeek-Distill-Llama-8B as the base model. "Acc." and "Len." denote accuracy and average token length, respectively.

| Method | MATH500 | | AIME | | AMC | | Olympiad | | Average | |
|---|---|---|---|---|---|---|---|---|---|---|
| | Acc. ↑ | Len. ↓ | Acc. ↑ | Len. ↓ | Acc. ↑ | Len. ↓ | Acc. ↑ | Len. ↓ | Acc. ↑ | Len. ↓ |
| DeepSeek-Distill-Llama-8B | 88.1 | 4042 | 48.3 | 13349 | 90.3 | 6363 | 62.9 | 8363 | 72.4 | 8029 |
| Fixed-Budget 3000 | **86.9** | 2375 | 43.7 | 8311 | 86.0 | 3577 | **59.2** | 4754 | **68.9** | 4754 |
| ABPO (Ours) | 86.7 | **1425** | **44.0** | **5872** | **86.5** | **2306** | 58.4 | **3039** | **68.9** | **3161** |

Table 13: Comparison under different rollout settings. † denotes the default evaluation setting used in the paper. "Acc.", "Len." and "AES" denote accuracy, average token length and Accuracy–Efficiency Score, respectively.

| Method | Evaluation Setting | MATH500 | | | AIME | | | AMC | | | Olympiad | | | Avg | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | Acc | Len | AES | Acc | Len | AES | Acc | Len | AES | Acc | Len | AES | Acc | Len | AES |
| DeepScaleR-1.5B | $K = 16$ | 88.3 | 3148 | – | 41.0 | 9200 | – | 81.4 | 5189 | – | 58.6 | 5678 | – | 67.3 | 5804 | – |
| DeepScaleR-1.5B | $K = (3, 10, 10, 5)^\dagger$ | 88.2 | 3155 | – | 40.0 | 9041 | – | 81.3 | 5067 | – | 58.3 | 5718 | – | 66.9 | 5745 | – |
| L1-Max | $K = 16$ | 83.2 | 1331 | 0.29 | 23.1 | 2064 | -1.40 | 74.7 | 1587 | 0.28 | 50.6 | 1545 | 0.04 | 57.9 | 1632 | 0.02 |
| L1-Max | $K = (3, 10, 10, 5)^\dagger$ | 84.5 | 1527 | 0.31 | 27.0 | 2290 | -0.88 | 75.5 | 1768 | 0.30 | 51.1 | 1733 | 0.07 | 59.5 | 1830 | 0.13 |
| ThinkPrune | $K = 16$ | 86.5 | 2034 | 0.25 | 34.4 | 5720 | -0.43 | 79.5 | 3078 | 0.29 | 56.1 | 3542 | 0.16 | 64.1 | 3594 | 0.14 |
| ThinkPrune | $K = (3, 10, 10, 5)^\dagger$ | 85.5 | 2029 | 0.21 | 34.7 | 5743 | -0.30 | 78.5 | 3125 | 0.21 | 55.0 | 3532 | 0.10 | 63.4 | 3607 | 0.11 |
| ABPO | $K = 16$ | 85.6 | 1874 | 0.26 | 37.3 | 5736 | 0.03 | 79.8 | 2845 | 0.35 | 55.0 | 3302 | 0.14 | 64.4 | 3439 | 0.21 |
| ABPO | $K = (3, 10, 10, 5)^\dagger$ | 85.9 | 1872 | 0.27 | 36.0 | 5566 | -0.21 | 79.8 | 2990 | 0.33 | 55.6 | 3308 | 0.16 | 64.3 | 3434 | 0.19 |

Table 14: Ablation study of ABPO on DeepSeek-R1-Distill-Qwen-1.5B. We progressively add Curriculum Learning, Adaptive Budget and Example Review to ABPO. $\star$ denotes the full setting of ABPO. "Acc.", "Len." and "AES" denote accuracy, average token length and Accuracy–Efficiency Score, respectively.

| | AMC | | | AIME | | | **Average** | | |
|---|---|---|---|---|---|---|---|---|---|
| | Acc.↑ | Len.↓ | AES↑ | Acc.↑ | Len.↓ | AES↑ | Acc.↑ | Len.↓ | AES↑ |
| Base | 70.3 | 9314 | – | 32.0 | 15476 | – | 51.1 | 12395 | – |
| + Curriculum Learning | 69.5 | 3337 | 0.59 | 28.0 | 6500 | -0.05 | 48.8 | 4918 | 0.37 |
| + Adaptive Budget | 69.0 | 2976 | 0.59 | 29.7 | 6383 | 0.23 | 49.4 | 4680 | 0.45 |
| + Review$^\star$ | **70.3** | **2715** | **0.71** | **30.7** | **6290** | **0.39** | **50.5** | **4502** | **0.58** |

Table 15: Comparison of different reward designs on DeepScaleR-1.5B-Preview. "ABPO w/ L1-Max Reward" replaces the hard budget-aware reward in ABPO with the soft length penalty of L1-Max while keeping the same curriculum learning framework. "Acc." and "Len." denote accuracy and average token length, respectively.

| Method | MATH500 | | AIME | | AMC | | Olympiad | | Avg | |
|---|---|---|---|---|---|---|---|---|---|---|
| | Acc. ↑ | Len. ↓ | Acc. ↑ | Len. ↓ | Acc. ↑ | Len. ↓ | Acc. ↑ | Len. ↓ | Acc. ↑ | Len. ↓ |
| DeepScaleR-1.5B-Preview | 88.2 | 3155 | 40.0 | 9041 | 81.3 | 5067 | 58.3 | 5718 | 66.9 | 5745 |
| L1-Max 1.5B | 84.5 | **1527** | 27.0 | **2290** | 75.5 | **1768** | 51.1 | **1733** | 59.5 | **1830** |
| ABPO w/ L1-Max Reward | 84.5 | 1655 | 31.3 | 5039 | 74.3 | 2712 | 53.8 | 2959 | 61.0 | 3091 |
| **ABPO (Ours)** | **85.6** | 1874 | **37.3** | 5736 | **79.8** | 2845 | **55.0** | 3302 | **64.4** | 3439 |

**EASY**

In a certain isosceles right triangle, the altitude to the hypotenuse has length $4\sqrt{2}$. What is the area of the triangle?

DeepScaleR-1.5B

Okay, **so** I have this problem here about an isosceles right triangle. Let me read it **again**: "In a certain isosceles right triangle, the altitude to the hypotenuse has length \(4\sqrt{2}\). What is the area of the triangle?" Hmm, I need to find the area, which is something I remember is (base * height)/2. **But** since it's an isosceles right triangle, maybe there's a simpler way.

First, let's recall what an isosceles right triangle is. It's a triangle with two legs equal in length and a right angle between them. **So**, the two legs are equal, and the angles are 45°, 45°, and 90°. That might help later.

Now, the problem says the altitude to the hypotenuse is \(4\sqrt{2}\). **Wait**, the altitude in a triangle is a line segment from a vertex perpendicular to the opposite side. **But** in this case, it's the altitude to the hypotenuse. Hmm, **so** in an isosceles right triangle, the altitude to the hypotenuse is also a median and an angle bisector, right? Because in a right isosceles triangle, the altitude from the right angle to the hypotenuse will split the hypotenuse into two equal parts. That might be useful.

**So**, let's denote the triangle as ABC, where angle C is the right angle, and sides AC and BC are equal. Let me draw that in my mind. **So**, triangle ABC with right angle at C, legs AC and BC equal, and hypotenuse AB.

The altitude to the hypotenuse would then be from point C to hypotenuse AB, and that length is given as \(4\sqrt{2}\). **Wait**, **but** in an isosceles right triangle, the altitude from the right angle to the hypotenuse is equal to half the hypotenuse? **Wait**, no, because in a right triangle, the legs multiplied by the hypotenuse and divided by the hypotenuse is the altitude. Maybe I should use some formula.

I remember that in any triangle, the area can be expressed as (base * height)/2. Since this is a triangle, maybe the area can be calculated in two ways: once using the legs as base and height, and once using the hypotenuse and the altitude to it.

**But wait**, in this case, the triangle is isosceles right, **so** let me denote the legs as length 'a', **so** the hypotenuse will be \(a\sqrt{2}\) since in a right triangle, the hypotenuse is \(a\sqrt{2}\) when legs are equal to 'a'.

**So**, the area of the triangle can be calculated as (a * a)/2 = \(a^2/2\). **Alternatively**, the area can be calculated as (hypotenuse * altitude)/2. The hypotenuse is \(a\sqrt{2}\), and the altitude is given as \(4\sqrt{2}\). **Therefore**, area is also ( \(a\sqrt{2}\) * \(4\sqrt{2}\) ) /2.

Let me compute that: first, multiply \(a\sqrt{2}\) and \(4\sqrt{2}\). That would be \(a * 4 * (\sqrt{2} * \sqrt{2})\) which is \(4a * 2\) because \(\sqrt{2} * \sqrt{2}\) is 2. **So**, that gives 8a. Then divide by 2, **so** the area is (8a)/2 = 4a.

**But** we also have the area as \(a^2/2\). **So**, equate these two expressions for the area: \(a^2 /2 = 4a\). Let me write that equation:

\(a^2 / 2 = 4a\)

**So**, a=8, same as before. Then area is (a^2)/2=64/2=32. **So**, same answer.

**Therefore**, area is 32.

**But wait**, the problem says "the altitude to the hypotenuse has length \(4\sqrt{2}\)", which we used to compute 'a' as 8 and area as 32. That seems consistent.

**Alternatively**, maybe think of the altitude in terms of similar triangles.

In an isosceles right triangle, when you draw the altitude from the right angle to the hypotenuse, it also bisects the hypotenuse. **So**, the hypotenuse is split into two equal segments, each of length \( (a\sqrt{2}) /2 \).

**But** in this case, the length of the altitude is \(4\sqrt{2}\). **Wait**, **perhaps** using similar triangles.

In the original triangle ABC, with right angle at C, and altitude CD to hypotenuse AB at point D. Then triangles ACD, CBD, and ABC are all similar.

**So**, triangle ACD is similar to triangle ABC, **so** their sides are proportional.

**So**, in triangle ACD, side AC is equal to 'a', side CD is the altitude \(4\sqrt{2}\), and hypotenuse AD is half of AB, which is \( (a\sqrt{2}) /2 \).

**Therefore**, triangle ACD is similar to triangle ABC, **so** the ratio of sides should be equal.

**So**, AC / AB = AD / AC.

Thus, AC is 'a', AB is \(a\sqrt{2}\), AD is \( (a\sqrt{2}) /2 \).

Thus, the ratio is a/(a√2) = (√2 / 2 ). On the other side, AD / AC is ( (a√2)/2 ) / a = (√2 / 2 ). **So**, that holds.

**Alternatively**, maybe consider that in similar triangles, the altitude is the geometric mean of the segments. **But** maybe that's complicating.

**Alternatively**, **perhaps** directly compute the area as (hypotenuse * altitude)/2.

We know altitude is 4√2, hypotenuse is a√2. **So**, area is (a√2 * 4√2)/2.

Let me compute that:

Multiply a√2 and 4√2: a * √2 *4√2 = a*4*(√2)^2 = a*4*2=8a.

Then divide by 2: 8a /2 =4a.

**But** we also know the area is (a^2)/2. **So**, equate 4a = a^2 /2.

Multiply both sides by 2: 8a =a^2.

Thus, same as before: a^2 -8a=0, **so** a=8.

**So**, area is 32. Same result.

**Therefore**, I think the area is 32.

**Final Answer**
The area of the triangle is \boxed{32}.
</think>

Given an isosceles right triangle with an altitude to the hypotenuse of length \(4\sqrt{2}\), we need to find the area of the triangle.

1. **Identify the properties of the isosceles right triangle**:
   - The legs are equal, denoted as \(a\).
   - The hypotenuse is \(a\sqrt{2}\).

2. **Calculate the area using the altitude**:
   - The area can be expressed as \(\frac{1}{2} \times \text{hypotenuse} \times \text{altitude}\).
   - The hypotenuse is \(a\sqrt{2}\) and the altitude is \(4\sqrt{2}\).
   - **Therefore**, the area is \(\frac{1}{2} \times a\sqrt{2} \times 4\sqrt{2}\).

3. **Simplify the area calculation**:
   - Multiply the terms: \(a\sqrt{2} \times 4\sqrt{2} = 4a \times 2 = 8a\).
   - Divide by 2: \(\frac{8a}{2} = 4a\).

4. **Calculate the area using the legs**:
   - The area is also \(\frac{1}{2} \times a \times a = \frac{a^2}{2}\).

5. **Equate the two expressions for the area**:
   - \(\frac{a^2}{2} = 4a\).
   - Solve for \(a\): \(a^2 = 8a\) leads to \(a(a - 8) = 0\), **so** \(a = 8\).

6. **Calculate the area**:
   - Substitute \(a = 8\) into the area formula: \(\frac{8^2}{2} = \frac{64}{2} = 32\).

Thus, the area of the triangle is \(\boxed{32}\).

L1-Max

Okay, **so** I have this problem here: In a certain isosceles right triangle, the altitude to the hypotenuse has length $4\sqrt{2}$. I need to find the area of the triangle. Hmm, let me try to figure this out step by step.

First, let me recall some properties of isosceles right triangles. An isosceles right triangle has two legs that are equal in length, and the angles are 45°, 45°, and 90°. **So**, if the legs are of length 'a', then the hypotenuse will be $a\sqrt{2}$. That's a standard result from the Pythagorean theorem.

Now, the problem is about the altitude to the hypotenuse. I remember that in a right triangle, the altitude to the hypotenuse has a special relationship with the legs. Let me try to visualize this. In an isosceles right triangle, the altitude to the hypotenuse will split the hypotenuse into two equal parts because the triangle is symmetric. **So**, if the hypotenuse is $a\sqrt{2}$, then each half will be $(a\sqrt{2})/2 = a/\sqrt{2}$.

I also recall that the area of the triangle can be expressed in two ways: one using the legs, and another using the hypotenuse and the altitude. **So**, the area is $(1/2)*a*a = (a^2)/2$. On the other hand, the area can also be expressed as (1/2)*hypotenuse*altitude. The hypotenuse is $a\sqrt{2}$, and the altitude is given as $4\sqrt{2}$. **So**, the area is also $(1/2)*(a\sqrt{2})*(4\sqrt{2})$.

Let me write that down:

Area = $(a^2)/2 = (1/2)*(a\sqrt{2})*(4\sqrt{2})$

Let me simplify the right-hand side:

$(1/2)*(a\sqrt{2})*(4\sqrt{2}) = (1/2)*a*\sqrt{2}*4*\sqrt{2} = (1/2)*a*4*(\sqrt{2}*\sqrt{2}) = (1/2)*a*4*2 = (1/2)*a*8 = 4a$

**So**, we have:

$(a^2)/2 = 4a$

Let me solve for 'a':

Multiply both sides by 2:

$a^2 = 8a$

Then, bring all terms to one side:

$a^2 - 8a = 0$

Factor out 'a':

$a(a - 8) = 0$

**So**, a = 0 or a = 8

Since a cannot be zero (it's a triangle), we have a = 8.

**Therefore**, the legs are each of length 8. **So**, the area of the triangle is $(1/2)*a^2 = (1/2)*64 = 32$.

**Wait**, let me double-check that. If a = 8, then the legs are 8 each, **so** the area is (8*8)/2 = 64/2 = 32. That seems correct.

**But** let me verify using the other expression for the area. We had area = 4a. If a = 8, then area = 4*8 = 32. Yes, that matches. **So**, that's consistent.

**Alternatively**, let me think about the altitude. In an isosceles right triangle, the altitude to the hypotenuse is equal to half the length of the hypotenuse. Is that true? **Wait**, no. Let me think.

In a right triangle, the length of the altitude to the hypotenuse is given by (product of the legs)/hypotenuse. **So**, in this case, since it's an isosceles right triangle, the legs are both 'a', **so** the altitude 'h' is (a*a)/(a√2) = a/√2.

**But** the problem says the altitude is 4√2. **So**, set that equal:

a/√2 = 4√2

Solve for 'a':

Multiply both sides by √2:

a = 4√2 * √2 = 4*2 = 8

**So**, **again**, we get a = 8. **Therefore**, the legs are 8 each, and the area is (8*8)/2 = 32.

**So**, both methods lead to the same result. **Therefore**, the area of the triangle is 32.

Just to ensure I didn't make any mistakes, let me go through the steps **again**.

1. Recognize it's an isosceles right triangle, **so** legs are equal, let's call them 'a'.
2. Hypotenuse is a√2.
3. The altitude to the hypotenuse in a right triangle is (product of legs)/hypotenuse = (a*a)/(a√2) = a/√2.
4. Given that this altitude is 4√2, **so** a/√2 = 4√2.
5. Solving for 'a': a = 4√2 * √2 = 4*2 = 8.
6. **Therefore**, legs are 8, area is (8*8)/2 = 32.

**Alternatively**, using the area in terms of hypotenuse and altitude:

Area = (1/2)*hypotenuse*altitude = (1/2)*(a√2)*(4√2) = (1/2)*a*8 = 4a.
Since area is also $(a^2)/2$, set equal: $(a^2)/2 = 4a \Rightarrow a^2 = 8a \Rightarrow a=8$.

Thus, area is 32.

Another way: In an isosceles right triangle, the altitude to the hypotenuse is equal to half the length of the leg. **Wait**, is that correct? Let me think.

No, actually, in a right triangle, the altitude to the hypotenuse is h = (a*b)/c, where a and b are legs, c is hypotenuse. Since it's isosceles, a = b, **so** $h = (a^2)/c$. Here, c = a√2, **so** $h = (a^2)/(a\sqrt{2}) = a/\sqrt{2}$, which is consistent with what I had before.

Given h = 4√2, **so** a/√2 = 4√2 => a = 8.

**Therefore**, all methods lead to the conclusion that the area is 32.

I think that's solid. I don't see any mistakes in the logic. The key was recognizing the relationship between the altitude and the legs, and then solving for 'a' which gave us the side length, and then computing the area.

Just to ensure, let me compute the hypotenuse with a=8: hypotenuse is 8√2. Then, the altitude is 4√2, which is half of the hypotenuse (since 8√2 / 2 = 4√2). **So**, that checks out. **Therefore**, the area is indeed 32.
</think>

The area of the isosceles right triangle is \boxed{32}.

ABPO

Okay, **so** I have this problem here: In an isosceles right triangle, the altitude to the hypotenuse has length $4\sqrt{2}$. I need to find the area of the triangle. Hmm, let's break this down step by step.

First, I remember that an isosceles right triangle has two legs of equal length and a hypotenuse. The two legs are the ones forming the right angle, and the hypotenuse is the side opposite the right angle. Since it's isosceles, both legs are equal in length.

Let me denote the length of each leg as 'a'. Then, the hypotenuse 'c' can be found using the Pythagorean theorem. Since both legs are equal, $c = \sqrt{(a^2 + a^2)} = \sqrt{(2a^2)} = a\sqrt{2}$.

Now, the problem mentions the altitude to the hypotenuse. In a right triangle, the altitude to the hypotenuse has a special property. I recall that this altitude is related to the legs and the hypotenuse. Specifically, the length of the altitude (let's denote it as 'h') can be found by the formula:

h = (product of the legs) / hypotenuse

**So**, $h = (a * a) / c = a^2 / c$

We know h is given as $4\sqrt{2}$, and c is $a\sqrt{2}$, **so** let's plug in:

$4\sqrt{2} = a^2 / (a\sqrt{2})$

Simplify the right side:

$a^2 / (a\sqrt{2}) = a / \sqrt{2}$

**So**, we have:

$4\sqrt{2} = a / \sqrt{2}$

Multiply both sides by $\sqrt{2}$ to solve for 'a':

$4\sqrt{2} * \sqrt{2} = a$

Since $\sqrt{2} * \sqrt{2} = 2$, this becomes:

$4 * 2 = a$

**So**, $a = 8$

**Therefore**, each leg is 8 units long.

Now, to find the area of the triangle, we can use the formula:

Area = (base * height) / 2

Here, the base and height are the legs of the triangle, both equal to 8.

**So**, Area = $(8 * 8) / 2 = 64 / 2 = 32$

**So**, Area = (8 * 8) / 2 = 64 / 2 = 32

**Alternatively**, another way to find the area is using the hypotenuse. Since it's an isosceles right triangle, the area can also be calculated as (hypotenuse²) / (2 * √2). **Wait**, let's check:

We know that hypotenuse c = a√2 = 8√2.

**So**, using the area formula with hypotenuse:

Area = (c * h) / 2 = (8√2 * 4√2) / 2

Calculate numerator:

8√2 * 4√2 = (8*4)*(√2*√2) = 32 * 2 = 64

Then, divide by 2: 64 / 2 = 32

Same result. **So**, the area is 32.

I think that's correct. Let me just verify once more.

Given h = 4√2. The legs are 8 each, hypotenuse is 8√2.

Area by legs: (8*8)/2 = 32.

Area by hypotenuse and altitude: (8√2 * 4√2)/2 = (32*2)/2 = 64/2 = 32.

Both methods give the same answer, **so** I'm confident.

**Final Answer**
The area of the triangle is \boxed{32}.
</think>

Given an isosceles right triangle with an altitude to the hypotenuse of length \(4\sqrt{2}\), we need to find the area of the triangle.

1. Let the length of each leg be \(a\). The hypotenuse \(c\) is \(a\sqrt{2}\).
2. The altitude \(h\) to the hypotenuse is given by \(h = \frac{a^2}{c}\).
3. Substituting the given altitude \(h = 4\sqrt{2}\) and \(c = a\sqrt{2}\):
   \[
   4\sqrt{2} = \frac{a^2}{a\sqrt{2}}
   \]
4. Simplifying the right side:
   \[
   4\sqrt{2} = \frac{a}{\sqrt{2}}
   \]
5. Solving for \(a\):
   \[
   4\sqrt{2} \cdot \sqrt{2} = a \implies 4 \cdot 2 = a \implies a = 8
   \]

6. The area of the triangle can be calculated using the legs:

$$
\text{Area} = \frac{1}{2} \times a \times a = \frac{1}{2} \times 8 \times 8 = \frac{64}{2} = 32
$$

Thus, the area of the triangle is $\boxed{32}$.

**HARD**

The circle $x^2 + y^2 = 2$ and the parabola $y^2 = 8x$ have two common tangents, forming four points of tangency. Find the area of the quadrilateral formed by the four points of tangency.

[asy] unitsize(0.8 cm);
real upperparab (real x) {return (sqrt(8*x));} real lowerparab (real x) {return (-sqrt(8*x));}
pair A, B, C, D; A = (-1,1); B = (2,4); C = (-1,-1); D = (2,-4);
draw(graph(upperparab,0,3)); draw(graph(lowerparab,0,3)); draw(Circle((0,0),sqrt(2)));
draw(interp(A,B,-0.2)--interp(A,B,1.2)); draw(interp(C,D,-0.2)--interp(C,D,1.2));
draw(A--C); draw(B--D); dot(A); dot(B); dot(C); dot(D); [/asy]

DeepScaleR-1.5B

Okay, **so** I have this problem where I need to find the area of a quadrilateral formed by the points of tangency of two curves: a circle and a parabola. The circle is given by the equation \( x^2 + y^2 = 2 \) and the parabola is \( y^2 = 8x \). The problem states that these two curves have two common tangents, which form four points of tangency. I need to find the area of the quadrilateral formed by these four points.

First, I should recall what common tangents are. A common tangent is a line that touches both curves at exactly one point without crossing them. Since the circle and the parabola are both symmetric with respect to the x-axis, I think their common tangents will also have some symmetry. **So perhaps** the four points of tangency are symmetric in some way as well.

Let me try to visualize this. The circle is centered at the origin with radius \( \sqrt{2} \). The parabola \( y^2 = 8x \) opens to the right and has its vertex at the origin. **So** the parabola is wider than the circle, which has a smaller radius. Their points of tangency should lie somewhere on the right side of the circle and the parabola.

I remember that the equation of a tangent line to a parabola can be written in the form \( y = mx + c \), where \( m \) is the slope and \( c \) is the y-intercept. For the parabola \( y^2 = 8x \), the condition for a line \( y = mx + c \) to be tangent is that the discriminant of the quadratic equation obtained by substituting \( y \) into the parabola's equation is zero.

Let me write that out. Substitute \( y = mx + c \) into \( y^2 = 8x \):

\( (mx + c)^2 = 8x \)
Expanding this:
\( m^2x^2 + 2mcx + c^2 - 8x = 0 \)
This is a quadratic in \( x \): \( m^2x^2 + (2mc - 8)x + c^2 = 0 \)

For this quadratic to have exactly one solution, the discriminant must be zero. The discriminant \( D \) is given by \( D = (2mc - 8)^2 - 4 * m^2 * c^2 \).

**So**, let's compute that:

\( D = (2mc - 8)^2 - 4m^2c^2 \)
Expanding \( (2mc - 8)^2 \):
\( 4m^2c^2 - 32mc + 64 \)
**So**, plug back into D:
\( D = (4m^2c^2 - 32mc + 64) - 4m^2c^2 \)

Simplify:
$D = -32mc + 64$
Set D = 0:
$-32mc + 64 = 0$
Solving for c:
$-32mc = -64$
$mc = 2$
**So**, $c = \frac{2}{m}$

**Therefore**, the condition for the line $y = mx + c$ to be tangent to the parabola is $c = \frac{2}{m}$.

Now, since this line is also tangent to the circle $x^2 + y^2 = 2$, let's find the condition for this line to be tangent to the circle.

The condition for a line $y = mx + c$ to be tangent to a circle $x^2 + y^2 = r^2$ is that the distance from the center (which is (0,0)) to the line is equal to the radius.

The distance from the center (0,0) to the line $mx - y + c = 0$ is given by $\frac{|0 - 0 + c|}{\sqrt{m^2 + 1}} = \frac{|c|}{\sqrt{m^2 + 1}}$.

Since this distance must be equal to the radius of the circle, which is $\sqrt{2}$, we have:

$\frac{|c|}{\sqrt{m^2 + 1}} = \sqrt{2}$
Squaring both sides:
$\frac{c^2}{m^2 + 1} = 2$
**So**, $c^2 = 2(m^2 + 1)$

**But** earlier, we found that $c = \frac{2}{m}$, **so** let's substitute that into this equation.

First, compute $c^2$:

$c^2 = \left( \frac{2}{m} \right)^2 = \frac{4}{m^2}$

Set this equal to $2(m^2 + 1)$:

$\frac{4}{m^2} = 2(m^2 + 1)$

Multiply both sides by $m^2$:

$4 = 2m^4 + 2m^2$

Divide both sides by 2:

$2 = m^4 + m^2$

Rewriting:

$m^4 + m^2 - 2 = 0$

Let me set $z = m^2$, **so** the equation becomes:

$z^2 + z - 2 = 0$

Solving this quadratic equation:

$z = \frac{ -1 \pm \sqrt{1 + 8} }{2} = \frac{ -1 \pm 3 }{2}$

**So**, $z = \frac{ -1 + 3 }{2 } = 1$ or $z = \frac{ -1 - 3 }{2 } = -2$. Since $z = m^2 \geq 0$, we discard the negative solution.

Thus, $z = 1$, **so** $m^2 = 1$, which means $m = 1$ or $m = -1$.

**Therefore**, the slopes of the common tangents are $m = 1$ and $m = -1$. Let's find the corresponding c values.

Since $c = \frac{2}{m}$:

- If $m = 1$, then $c = 2$.
- If $m = -1$, then $c = -2$.

Thus, the equations of the tangent lines are:

- For $m = 1$: $y = x + 2$
- For $m = -1$: $y = -x - 2$

**Wait**, hold on. Let me check that. If $m = -1$, then $c = 2 / (-1) = -2$, **so** the line is $y = -x - 2$. Correct.

Now, we need to find the points where these tangent lines touch the parabola and the circle. Since we have the equations of the lines, we can find their points of tangency.

Let's start with the parabola $y^2 = 8x$.

Take the line $y = x + 2$. To find the point of tangency, substitute $y = x + 2$ into $y^2 = 8x$:

$(x + 2)^2 = 8x$
Expanding:
$x^2 + 4x + 4 = 8x$
Simplify:
$x^2 - 4x + 4 = 0$
Which factors as:
$(x - 2)^2 = 0$
Thus, $x = 2$. Then $y = 2 + 2 = 4$. **So** the point of tangency is (2, 4).

Similarly, for the line $y = -x - 2$:

Substitute into $y^2 = 8x$:

$(-x - 2)^2 = 8x$
Expanding:
$x^2 + 4x + 4 = 8x$

Simplify:
$x^2 - 4x + 4 = 0$
**Again**, factors as $(x - 2)^2 = 0$, **so** $x = 2$. Then $y = -2 - 2 = -4$. **So** the point of tangency is (2, -4).

Now, let's find the points of tangency on the circle $x^2 + y^2 = 2$.

First, let's take the line $y = x + 2$. Substitute into the circle's equation:

$x^2 + (x + 2)^2 = 2$
Expanding:
$x^2 + x^2 + 4x + 4 = 2$
Combine like terms:
$2x^2 + 4x + 4 - 2 = 0$
Simplify:
$2x^2 + 4x + 2 = 0$
Divide by 2:
$x^2 + 2x + 1 = 0$
Which factors as:
$(x + 1)^2 = 0$
**So**, $x = -1$. Then $y = (-1) + 2 = 1$. **So** the point of tangency is (-1, 1).

Similarly, for the line $y = -x - 2$:

Substitute into circle's equation:

$x^2 + (-x - 2)^2 = 2$
Expanding:
$x^2 + x^2 + 4x + 4 = 2$
Combine like terms:
$2x^2 + 4x + 4 - 2 = 0$
Simplify:
$2x^2 + 4x + 2 = 0$
Divide by 2:
$x^2 + 2x + 1 = 0$
Factor:
$(x + 1)^2 = 0$
Thus, $x = -1$. Then $y = -(-1) - 2 = 1 - 2 = -1$. **So** the point of tangency is (-1, -1).

**Therefore**, the four points of tangency are:

- From the line $y = x + 2$: (2, 4)
- From the line $y = -x - 2$: (2, -4)
- From the line $y = x + 2$: (-1, 1)
- From the line $y = -x - 2$: (-1, -1)

**Wait**, hold on. **Wait**, actually, each tangent line touches the circle at one point, **so** each line corresponds to one point. **But** since there are two lines, each tangent line touches the circle at one point, **so** actually, we have four points? **Wait**, no, each line is tangent to the circle once, **so** two lines give two points. **Wait**, **perhaps** I made a mistake here.

**Wait**, **perhaps** not. Let's clarify.

Each of the two lines (with slopes 1 and -1) is tangent to both the parabola and the circle. Each line is tangent to the parabola at one point and tangent to the circle at one point. **Therefore**, each line gives two points of tangency: one on the parabola and one on the circle.

**But** in the problem statement, it says that there are two common tangents. Each common tangent touches both curves at one point each. **Therefore**, each tangent line touches the parabola at one point and the circle at another point. **Therefore**, each of the two tangent lines gives two points of tangency. However, since there are two tangent lines, each with their own points, **so** in total, four points of tangency on the parabola and circle.

**But wait**, looking back at the Asymptote code provided, it defines four points A, B, C, D as (-1,1), (2,4), (-1,-1), (2,-4). **So** these four points are on both the circle and the parabola? **Wait**, hold on.

**Wait**, no. The Asymptote code seems to define points A, B as on the parabola and points C, D as on the circle? Or is it the other way around?

**Wait**, **perhaps** let's see. The Asymptote code defines:

```
pair A, B, C, D;

A = (-1,1);
B = (2,4);
C = (-1,-1);
D = (2,-4);
```

Then draws the circle and the parabola, and then the lines connecting A-C and B-D.

**So perhaps** the four points of tangency are A, B on the parabola and C, D on the circle? **Wait**, no, because A is (-1,1) on the circle, B is (2,4) on the parabola, C is (-1,-1) on the circle, and D is (2,-4) on the parabola. Thus, the four points are A, B, C, D, which are ( -1,1 ), (2,4), (-1,-1), (2,-4). **So** the quadrilateral is formed by connecting these four points.

**But wait**, **perhaps** it's connected as A to B, B to D, D to C, and C to A? Or **perhaps** a convex quadrilateral formed by these four points.

**But perhaps** it's better to think that each tangent line touches the parabola at one point and the circle at another point, **so** since there are two tangent lines, each giving two points, **but perhaps** in reality, each tangent line is tangent to the parabola at one point and the circle at another point. Thus, in total, four points: two on the parabola and two on the circle.

**But** the Asymptote code seems to have four points: two on the parabola (B and D) and two on the circle (A and C). **Therefore**, **perhaps** the quadrilateral is formed by connecting these four points in some order.

**But** maybe it's better to actually calculate the coordinates first.

From earlier, we have four points:

From line $y = x + 2$: tangent to parabola at (2,4) and tangent to circle at (-1,1).

From line y = -x -2: tangent to parabola at (2,-4) and tangent to circle at (-1,-1).

**Therefore**, the four points of tangency are:

- On the parabola: (2,4) and (2,-4)
- On the circle: (-1,1) and (-1,-1)

Thus, these four points are (2,4), (2,-4), (-1,1), (-1,-1). **Wait**, **but** in the Asymptote code, the points are given as:

A = (-1,1);
B = (2,4);
C = (-1,-1);
D = (2,-4);

**So perhaps** the four points are A(-1,1), B(2,4), C(-1,-1), D(2,-4). **Therefore**, the quadrilateral is formed by connecting these four points in order.

**So** to find the area of quadrilateral ABCD with vertices at (-1,1), (2,4), (-1,-1), (2,-4).

**But** let's check: **Wait**, **perhaps** the quadrilateral is formed by connecting the points as A to B to D to C to A? Or **perhaps** A to C to B to D?

**Wait**, **perhaps** it's better to plot these points.

Point A: (-1,1)
Point B: (2,4)
Point C: (-1,-1)
Point D: (2,-4)

Plotting these points:

- (-1,1) is in the second quadrant.
- (2,4) is in the first quadrant.
- (-1,-1) is in the third quadrant.
- (2,-4) is in the fourth quadrant.

**So** the quadrilateral would have vertices in the second, first, third, and fourth quadrants. Connecting them in order: A to B to D to C to A. Hmm, maybe.

**Alternatively**, **perhaps** it's A(-1,1) connected to B(2,4), then to D(2,-4), then to C(-1,-1), then back to A(-1,1). That would form a quadrilateral.

**But perhaps** it's better to compute the area using coordinates.

Given four points, we can use the shoelace formula to compute the area.

**But** first, let's make sure of the order of the points. Since the quadrilateral is symmetric about the y-axis, **perhaps** the order should be A(-1,1), B(2,4), D(2,-4), C(-1,-1), and back to A.

**Alternatively**, **perhaps** just take the points as given and compute the area.

**Alternatively**, **perhaps** notice that the quadrilateral is symmetric about the x-axis. Because points A and C are reflections over the x-axis, as are points B and D.

**Therefore**, **perhaps** the quadrilateral is a kite or something similar.

**Alternatively**, **perhaps** it's a trapezoid or a parallelogram.

**But** let's proceed step by step.

First, let's list the coordinates:

A: (-1,1)
B: (2,4)
C: (-1,-1)
D: (2,-4)

**So** the four points are (-1,1), (2,4), (-1,-1), (2,-4). Let's arrange them in order.

**But perhaps** the correct order is A(-1,1), B(2,4), D(2,-4), C(-1,-1), back to A(-1,1).

Thus, the coordinates in order are:

1. (-1,1)
2. (2,4)
3. (2,-4)
4. (-1,-1)

Now, let's apply the shoelace formula.

The shoelace formula is given by:

Area = 1/2 |sum_{i=1 to n} (x_i y_{i+1} - x_{i+1} y_i)|

where (x_{n+1}, y_{n+1}) = (x_1, y_1).

**So** let's compute each term.

First, list the coordinates in order:

1. A: (-1,1)
2. B: (2,4)
3. D: (2,-4)
4. C: (-1,-1)
5. Back to A: (-1,1)

Compute the terms:

Term 1: x1 y2 - x2 y1 = (-1)(4) - (2)(1) = -4 - 2 = -6

Term 2: x2 y3 - x3 y2 = (2)(-4) - (2)(4) = -8 - 8 = -16

Term 3: x3 y4 - x4 y3 = (2)(-1) - (-1)(-4) = -2 - 4 = -6

Term 4: x4 y5 - x5 y4 = (-1)(1) - (-1)(-1) = -1 - 1 = -2

Sum all terms:

-6 + (-16) + (-6) + (-2) = -6 -16 -6 -2 = -30

Take absolute value and multiply by 1/2:

Area = (1/2)|-30| = (1/2)(30) = 15

**Therefore**, the area is 15.

**But** let me verify this because sometimes the order of points affects the calculation.

**Alternatively**, **perhaps** arrange the points in a different order.

**Alternatively**, **perhaps** arrange them in such a way that the polygon is traversed correctly without crossing.

**Alternatively**, **perhaps** use vectors or coordinate geometry.

**Alternatively**, **perhaps** compute the area using vectors or by dividing the quadrilateral into triangles.

**But** since we have four points, **perhaps** it's better to compute using shoelace.

**Wait**, **perhaps** let's write the coordinates in order, ensuring that the polygon is simple and the points are ordered correctly.

**Alternatively**, **perhaps** it's better to note the symmetry.

The quadrilateral has points (-1,1), (2,4), (-1,-1), (2,-4). **So perhaps** the quadrilateral can be split into two triangles: one above the x-axis and one below.

**But perhaps** it's more straightforward to compute using shoelace.

**Alternatively**, **perhaps** note that the quadrilateral is symmetric about the y-axis.

**Wait**, looking at the coordinates:

A(-1,1), B(2,4), D(2,-4), C(-1,-1)

If we plot these, it seems that the quadrilateral is symmetric about the line x = 0.5, **perhaps**.

**But perhaps** let's just proceed with the shoelace formula step by step.

**So** let's list the points in order: A, B, D, C, A.

Compute the shoelace sum.

Compute (x1 y2 - x2 y1) + (x2 y3 - x3 y2) + (x3 y4 - x4 y3) + (x4 y5 - x5 y4)

Where:

x1 = -1, y1 = 1

x2 = 2, y2 = 4

x3 = 2, y3 = -4

x4 = -1, y4 = -1

x5 = x1 = -1, y5 = y1 = 1

Compute each term:

1. x1 y2 - x2 y1 = (-1)(4) - (2)(1) = -4 - 2 = -6

2. x2 y3 - x3 y2 = (2)(-4) - (2)(4) = -8 - 8 = -16

3. x3 y4 - x4 y3 = (2)(-1) - (-1)(-4) = -2 - 4 = -6

4. x4 y5 - x5 y4 = (-1)(1) - (-1)(-1) = -1 - 1 = -2

Sum these: -6 -16 -6 -2 = (-6 -16) + (-6 -2) = (-22) + (-8) = -30

Take absolute value and multiply by 1/2: | -30 | * 1/2 = 30 * 1/2 = 15

**Therefore**, the area is 15.

**Alternatively**, **perhaps** it's better to use vectors or divide the quadrilateral into triangles.

Let's try dividing the quadrilateral into two triangles: ABD and BCD.

**Wait**, **perhaps** not. **Alternatively**, **perhaps** divide it into triangles ABC and ACD, **but perhaps** it's more complicated.

**Alternatively**, **perhaps** compute the area using coordinates.

**Alternatively**, **perhaps** note that the quadrilateral is a trapezoid. Let's check if any sides are parallel.

Compute the slopes:

Slope AB: (4 -1)/(2 - (-1)) = 3/3 = 1

Slope BC: (-4 -4)/(2 -2) = (-8)/0, undefined (vertical line)

Slope CD: (-1 - (-4))/( -1 -2 ) = (3)/(-3) = -1

Slope DA: (1 - (-1))/( -1 - (-1)) = 2/0, undefined (vertical line)

**Wait**, **perhaps** the quadrilateral is symmetric across the y-axis.

**Alternatively**, **perhaps** compute the area as two congruent trapezoids or something.

**But perhaps** since we've already computed using shoelace formula and got 15, **perhaps** that's the answer.

**Therefore**, the area of the quadrilateral formed by the four points of tangency is 15.

**Final Answer**
The area of the quadrilateral is \boxed{15}.
</think>

To find the area of the quadrilateral formed by the four points of tangency of the circle \(x^2 + y^2 = 2\) and the parabola \(y^2 = 8x\), we start by determining the points of tangency.

1. **Equation of the tangent line to the parabola**:
   - The condition for a line \(y = mx + c\) to be tangent to the parabola \(y^2 = 8x\) is \(c = \frac{2}{m}\).

2. **Equation of the tangent line to the circle**:
   - The condition for a line \(y = mx + c\) to be tangent to the circle \(x^2 + y^2 = 2\) is \(\frac{|c|}{\sqrt{m^2 + 1}} = \sqrt{2}\). This simplifies to \(c^2 = 2(m^2 + 1)\).

3. **Finding the slopes and intercepts**:
   - Solving these conditions, we find the slopes \(m = 1\) and \(m = -1\), and the corresponding intercepts \(c = 2\) and \(c = -2\).

4. **Points of tangency**:
   - For the line \(y = x + 2\), the points of tangency on the parabola and the circle are \((2, 4)\) and \((-1, 1)\).
   - For the line \(y = -x - 2\), the points of tangency on the parabola and the circle are \((2, -4)\) and \((-1, -1)\).

5. **Coordinates of the points**:
   - The four points of tangency are \((-1, 1)\), \((2, 4)\), \((-1, -1)\), and \((2, -4)\).

6. **Area calculation using the shoelace formula**:
   - Using the shoelace formula for the points \((-1, 1)\), \((2, 4)\), \((2, -4)\), and \((-1, -1)\), we find the area to be 15.

Thus, the area of the quadrilateral is \(\boxed{15}\).

L1-Max

Okay, **so** I have this problem here: there's a circle defined by the equation $x^2 + y^2 = 2$ and a parabola defined by $y^2 = 8x$. They have two common tangents, and these tangents touch the circle at four points. I need to find the area of the quadrilateral formed by these four points of tangency.

First, let me understand the problem. We have a circle centered at the origin with radius sqrt(2), which is approximately 1.414. The parabola $y^2 = 8x$ is a right-opening parabola with vertex at the origin. Its focus is at (2,0) and the directrix is the line x = -2.

The problem mentions that there are two common tangents to both the circle and the parabola. These tangents touch the circle at four points, and these points form a quadrilateral. I need to find the area of this quadrilateral.

Looking at the Asymptote code, it seems like the points of tangency are given as A, B, C, D with coordinates (-1,1), (2,4), (-1,-1), and (2,-4). **So**, **perhaps** these are the four points. **But** I shouldn't rely on that; I need to derive the solution.

I remember that the equation of a tangent to a parabola $y^2 = 4ax$ is given by $y = mx + a/m$, where m is the slope. **But** in this case, the parabola is $y^2 = 8x$, **so** $4a = 8$, which means $a = 2$. **So**, the tangent to this parabola would be $y = mx + 2/m$.

Similarly, the circle $x^2 + y^2 = 2$ has the equation of tangent lines. The general equation of a tangent to a circle $x^2 + y^2 = r^2$ is $x \cos\theta + y \sin\theta = r$. Here, r = sqrt(2), **so** the tangent would be $x \cos\theta + y \sin\theta$ = sqrt(2).

Since these are common tangents, the equations of the tangents to both the circle and the parabola must be the same. **So**, **perhaps** I can set the equations equal to each other.

**So**, let's suppose that the tangent line to the parabola is $y = mx + 2/m$, and this is also a tangent to the circle. **Therefore**, the distance from the center of the circle (which is at (0,0)) to this line must be equal to the radius sqrt(2).

The distance from (0,0) to the line $y = mx + 2/m$ is $|0 - mx - 2/m| / \text{sqrt}(m^2 + 1) = |mx + 2/m| / \text{sqrt}(m^2 + 1)$. **But** since it's a tangent, this distance should be equal to the radius sqrt(2).

**Therefore**, $|mx + 2/m| / \text{sqrt}(m^2 + 1) = \text{sqrt}(2)$. **But wait**, actually, the line is $y = mx + c$, where $c = 2/m$. **So**, the distance is $|c| / \text{sqrt}(1 + m^2) = |2/m| / \text{sqrt}(1 + m^2) = \text{sqrt}(2)$.

**Therefore**, $|2/m| / \text{sqrt}(1 + m^2) = \text{sqrt}(2)$. Let's solve for m.

Square both sides: $(4/m^2) / (1 + m^2) = 2$.

**So**, $4 / (m^2(1 + m^2)) = 2$.

Multiply both sides by $m^2(1 + m^2)$: $4 = 2m^2(1 + m^2)$.

Simplify: $4 = 2m^2 + 2m^4$.

Divide both sides by 2: $2 = m^2 + m^4$.

Rearrange: $m^4 + m^2 - 2 = 0$.

Let me set $z = m^2$, then the equation becomes $z^2 + z - 2 = 0$.

Solve for z: $z = [-1 \pm \text{sqrt}(1 + 8)] / 2 = [-1 \pm 3]/2$.

**So**, $z = (-1 + 3)/2 = 1$ or $z = (-1 -3)/2 = -2$. Since $z = m^2$ can't be negative, $z = 1$.

**Therefore**, $m^2 = 1$, **so** $m = 1$ or $m = -1$.

**Therefore**, the slopes are $m = 1$ and $m = -1$.

**Therefore**, the equations of the tangent lines are $y = x + 2/1 = x + 2$ and $y = -x + 2/(-1) = -x -2$.

**So**, the tangent lines are $y = x + 2$ and $y = -x -2$.

Now, let's find the points of tangency on the circle $x^2 + y^2 = 2$.

For the line $y = x + 2$, substitute into the circle equation:

$x^2 + (x + 2)^2 = 2$.

Expand: $x^2 + x^2 + 4x + 4 = 2$.

Combine like terms: $2x^2 + 4x + 4 - 2 = 0 \Rightarrow 2x^2 + 4x + 2 = 0$.

Divide by 2: $x^2 + 2x + 1 = 0$.

This factors as $(x + 1)^2 = 0$, **so** $x = -1$. **Therefore**, $y = (-1) + 2 = 1$. **So**, the point is (-1, 1).

Similarly, for the line $y = -x -2$, substitute into the circle:

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

Expand: $x^2 + x^2 + 4x + 4 = 2$.

Combine like terms: $2x^2 + 4x + 4 - 2 = 0 \Rightarrow 2x^2 + 4x + 2 = 0$.

Divide by 2: $x^2 + 2x + 1 = 0$.

Which factors as $(x + 1)^2 = 0$, **so** $x = -1$. Then $y = -(-1) -2 = 1 -2 = -1$. **So**, the point is (-1, -1).

**Therefore**, the four points of tangency are (-1,1), (-1,-1), (1,1), and (1,-1). **Wait**, **but** in the Asymptote code, it shows points as (-1,1), (2,4), (-1,-1), (2,-4). Hmm, **perhaps** the Asymptote code is using a different coordinate system or scaling.

**But** based on our calculations, the points are (-1,1), (-1,-1), (1,1), and (1,-1). **So**, these are the four points.

**Therefore**, the quadrilateral formed by these points is a rectangle because the points are symmetric with respect to both x and y axes.

To find the area, we can find the lengths of the sides.

The points are (-1,1), (-1,-1), (1,1), (1,-1).

**So**, the distance between (-1,1) and (-1,-1) is 2 units (along y-axis).

Similarly, the distance between (-1,-1) and (1,-1) is 2 units (along x-axis).

**Therefore**, the quadrilateral is a rectangle with side lengths 2 and 2. **Therefore**, the area is 2*2=4.

**But wait**, looking back, the points are (-1,1), (-1,-1), (1,1), (1,-1). **So**, plotting these, they form a rectangle centered at the origin with vertices at (±1, ±1). **Therefore**, the length and width are both 2 units.

Thus, the area is 2*2=4.

**Therefore**, the area of the quadrilateral is 4.

**Final Answer**
The area of the quadrilateral is \boxed{4}.
</think>

To find the area of the quadrilateral formed by the four points of tangency of the common tangents to the circle \(x^2 + y^2 = 2\) and the parabola \(y^2 = 8x\), we start by determining the equations of the tangents.

The tangents to the parabola \(y^2 = 8x\) are given by \(y = mx + \frac{2}{m}\). These lines must also be tangent to the circle \(x^2 + y^2 = 2\). The distance from the center of the circle to this line must be equal to the radius \(\sqrt{2}\).

Solving for the slopes \(m\), we find that \(m = 1\) and \(m = -1\). The corresponding tangent lines are \(y = x + 2\) and \(y = -x - 2\).

Finding the points of tangency on the circle, we get the points \((-1, 1)\), \((-1, -1)\), \((1, 1)\), and \((1, -1)\). These points form a rectangle.

The area of this rectangle is calculated by finding the side lengths, which are both 2 units. **Therefore**, the area is \(2 \times 2 = 4\).

Thus, the area of the quadrilateral is \(\boxed{4}\).

ABPO

Okay, **so** I need to find the area of the quadrilateral formed by the four points of tangency between the circle $x^2 + y^2 = 2$ and the parabola $y^2 = 8x$. Hmm, let's break this down step by step.

First, I know that both the circle and the parabola are symmetric about the x-axis because the equation of the parabola has $y^2$, which is even in y, and the circle equation is symmetric in both x and y. **But wait**, the circle equation is $x^2 + y^2 = 2$, which is symmetric about both axes, while the parabola $y^2 = 8x$ is symmetric about the x-axis.

**Therefore**, the four points of tangency should form a symmetric quadrilateral, probably a kite or something else. **But** let's think more carefully.

Each tangent line to both the circle and the parabola will touch the circle at one point and the parabola at another. Since the parabola is symmetric about the x-axis and the circle is symmetric about both, the points of tangency should be symmetric as well.

Let me recall that the equation of a tangent to a parabola $y^2 = 4ax$ is $yy_1 = 2a(x + x_1)$. Comparing to our parabola $y^2 = 8x$, which is similar to $y^2 = 4a x$ with $4a = 8$, **so** $a = 2$. **Therefore**, the tangent to the parabola at point $(x_1, y_1)$ is $yy_1 = 8(x + x_1)/2$, **wait** no, let me check.

Actually, for $y^2 = 4a x$, the tangent is $yy_1 = 2a(x + x_1)$. **Wait**, no, let's derive it.

Let me take the parabola $y^2 = 8x$. Let's find the tangent at a general point $(x_1, y_1)$ on the parabola. The derivative will give the slope.

Differentiating both sides: $2y \frac{dy}{dx} = 8$, **so** $\frac{dy}{dx} = \frac{4}{y}$. **Therefore**, the slope of the tangent at $(x_1, y_1)$ is $\frac{4}{y_1}$.

Using the point-slope form: $y - y_1 = \frac{4}{y_1}(x - x_1)$. **But** since $y_1^2 = 8x_1$, we can write $x_1 = \frac{y_1^2}{8}$. Substitute into the equation:

$y - y_1 = \frac{4}{y_1}\left(x - \frac{y_1^2}{8}\right)$.

Multiply both sides by $y_1$:

$y y_1 - y_1^2 = 4\left(x - \frac{y_1^2}{8}\right)$.

Simplify:

$y y_1 - y_1^2 = 4x - \frac{y_1^2}{2}$.

Bring all terms to one side:

$y y_1 - y_1^2 - 4x + \frac{y_1^2}{2} = 0$.

Simplify:

$y y_1 - 4x - \frac{y_1^2}{2} = 0$.

$D = [\pm 2m\sqrt{(2(1 + m^2))} - 8]^2 - 4 * m^2 * 2(1 + m^2) = 0$

This seems a bit complicated, **but** let's compute it step by step.

First, compute the discriminant:

$D = [\pm 2m\sqrt{(2(1 + m^2))} - 8]^2 - 8m^2(1 + m^2) = 0$

Let me compute the first square term:

$[\pm 2m\sqrt{(2(1 + m^2))} - 8]^2 = (\pm 2m\sqrt{(2(1 + m^2))})^2 - 2 * 2m\sqrt{(2(1 + m^2))} * 8 + 8^2$

**Wait**, actually, expanding $(a - b)^2 = a^2 - 2ab + b^2$.

**So**,

$= (2m\sqrt{(2(1 + m^2))})^2 - 2 * 2m\sqrt{(2(1 + m^2))} * 8 + 8^2$

$= 4m^2 * 2(1 + m^2) - 32m\sqrt{(2(1 + m^2))} + 64$

$= 8m^2(1 + m^2) - 32m\sqrt{(2(1 + m^2))} + 64$

Then subtract $8m^2(1 + m^2)$:

**So**,

$D = [8m^2(1 + m^2) - 32m\sqrt{(2(1 + m^2))} + 64] - 8m^2(1 + m^2) = 0$

Simplify:

$8m^2(1 + m^2)$ cancels with $-8m^2(1 + m^2)$, leaving:

$-32m\sqrt{(2(1 + m^2))} + 64 = 0$

**So**,

$-32m\sqrt{(2(1 + m^2))} + 64 = 0$

Move 64 to the other side:

$-32m\sqrt{(2(1 + m^2))} = -64$

Divide both sides by -32:

$m\sqrt{(2(1 + m^2))} = 2$

Square both sides to eliminate the square root:

$m^2 * 2(1 + m^2) = 4$

Simplify:

**But** since $y_1^2 = 8x_1$, substitute:

$y y_1 - 4x - 4x_1 = 0$.

**So**, the tangent equation is $y y_1 = 4x + 4x_1$.

**But** since $x_1 = \frac{y_1^2}{8}$, substitute:

$y y_1 = 4x + 4 \cdot \frac{y_1^2}{8}$.

Simplify:

$y y_1 = 4x + \frac{y_1^2}{2}$.

**But** since $y_1^2 = 8x_1$, then $\frac{y_1^2}{2} = 4x_1$. **Therefore**, $y y_1 = 4x + 4x_1$.

**But perhaps** another way is better. **Alternatively**, since I know the circle is $x^2 + y^2 = 2$, and the tangent lines to both the circle and parabola will have some relation.

**But** maybe instead of dealing with tangents, let's consider the system where the tangent lines satisfy both equations.

Let me assume that a line is tangent to both the circle and the parabola. Let's write the general equation of a line.

Since both the circle and parabola are symmetric about the x-axis, **perhaps** the tangent points are symmetric as well.

Let me denote the tangent points on the parabola as $(a, b)$ and on the circle as $(c, d)$. Since it's a tangent, the line connecting these two points must satisfy the condition for tangency.

**But** maybe a better approach is to find the equations of the common tangents.

For a parabola $y^2 = 8x$, the equation of the tangent can be written as $y = mx + \frac{2}{m}$, where m is the slope. **Wait**, let's verify.

The standard form for a tangent to $y^2 = 4ax$ is $y = mx + \frac{a}{m}$. Since here, $4a = 8$, **so** $a = 2$. **Therefore**, the tangent equation is $y = mx + \frac{2}{m}$.

Similarly, for the circle $x^2 + y^2 = 2$, the equation of the tangent can be written as $xx_1 + yy_1 = 2$, where $(x_1, y_1)$ is the point of tangency. Since the point is on the circle, $x_1^2 + y_1^2 = 2$.

**Therefore**, the tangent line is $x x_1 + y y_1 = 2$.

Since the tangent lines are same for both the parabola and the circle, their equations must be equal. **Therefore**, we can set the equations equal:

For the parabola: $y = mx + \frac{2}{m}$.

For the circle: $x x_1 + y y_1 = 2$.

**So**, let's express y from the parabola's tangent equation: $y = mx + \frac{2}{m}$.

Plugging into the circle's tangent equation:

$x x_1 + \left(mx + \frac{2}{m}\right) y_1 = 2$.

Simplify:

$x x_1 + m x y_1 + \frac{2 y_1}{m} = 2$.

This equation must hold for all x, **so** the coefficients of x and the constant term must satisfy certain conditions. Let's write it as:

$(x_1 + m y_1) x + \frac{2 y_1}{m} - 2 = 0$.

Since this is a tangent to both, the coefficients must satisfy:

1. Coefficient of x is zero: $x_1 + m y_1 = 0$.
2. Constant term is zero: $\frac{2 y_1}{m} - 2 = 0$.

From condition 2: $\frac{2 y_1}{m} = 2$ => $y_1 = m$.

From condition 1: $x_1 + m y_1 = 0$ => $x_1 + m^2 = 0$ => $x_1 = -m^2$.

Now, since the point $(x_1, y_1)$ is on the circle, we have $x_1^2 + y_1^2 = 2$.

Substituting $x_1 = -m^2$ and $y_1 = m$:

$(-m^2)^2 + m^2 = 2$ => $m^4 + m^2 - 2 = 0$.

Let me solve for m:

Let $u = m^2$, then equation becomes $u^2 + u - 2 = 0$.

Solving quadratic: $u = \frac{-1 \pm \sqrt{1 + 8}}{2} = \frac{-1 \pm 3}{2}$.

Thus, $u = 1$ or $u = -2$. Since $u = m^2$ must be non-negative, we take $u = 1$, **so** $m^2 = 1$ => $m = 1$ or $m = -1$.

**Therefore**, possible slopes are m = 1 and m = -1.

Thus, for each m, we can find the tangent points.

First, for m = 1:

From $y_1 = m = 1$ and $x_1 = -m^2 = -1$.

**Therefore**, the tangent point on the parabola is (-1, 1).

Similarly, the tangent point on the circle is (x1, y1) = (-1, 1). **Wait**, **but** the circle equation is $x^2 + y^2 = 2$, **so** plug in (-1,1): $(-1)^2 + 1^2 = 1 + 1 = 2$, which is correct.

Similarly, for m = -1:

From \(y_1 = m = -1\) and \(x_1 = -m^2 = -1\).

**Therefore**, the tangent point on the parabola is (-1, -1).

**So**, the four points of tangency are: (-1,1), (-1,-1), (1,1), (1,-1). **Wait**, **but** hold on, are these four points?

**But** let's think: for m =1, we get (-1,1); for m=-1, we get (-1,-1). **But perhaps** for each m, there are two lines, **but** since we're dealing with common tangents, **perhaps** the upper and lower ones. **Wait**, maybe I need to consider both positive and negative slopes.

**Alternatively**, **perhaps** each slope m gives two points? No, **perhaps** each slope gives one point.

**Wait**, **perhaps** I need to check.

From the condition above, we have m =1 and m=-1. **Therefore**, the tangent lines are y = x + 2 and y = -x -2? **Wait**, let's compute the tangent equations.

When m =1: y = x + 2/1 = x + 2.

When m = -1: y = -x + 2/(-1) = -x -2.

**So** the lines are y = x + 2 and y = -x -2.

Now, let's find the points of tangency on the parabola.

For y = x + 2: substituting into parabola \(y^2 = 8x\):

\((x + 2)^2 = 8x\).

Expand: \(x^2 + 4x +4 = 8x\).

Simplify: \(x^2 -4x +4 =0\).

Solve: \(x = [4 ± sqrt(16 -16)] / 2 = [4]/2=2\). **So** x=2, then y=2+2=4. **So** the point is (2,4).

Similarly, for y = -x -2: substituting into parabola:

\((-x -2)^2 =8x\).

Expand: \(x^2 +4x +4 =8x\).

Simplify: \(x^2 -4x +4=0\).

Solve: x=2, then y= -2 -2= -4. **So** the point is (2,-4).

Now, find the points of tangency on the circle.

Figure 7: Detailed visualization of model outputs on both easy and hard examples.