

000 DRQA: DYNAMIC REASONING QUOTA ALLOCATION 001 FOR CONTROLLING OVERTHINKING IN REASONING 002 LARGE LANGUAGE MODELS 003

004
005
006 **Anonymous authors**
007 Paper under double-blind review
008
009
010
011
012

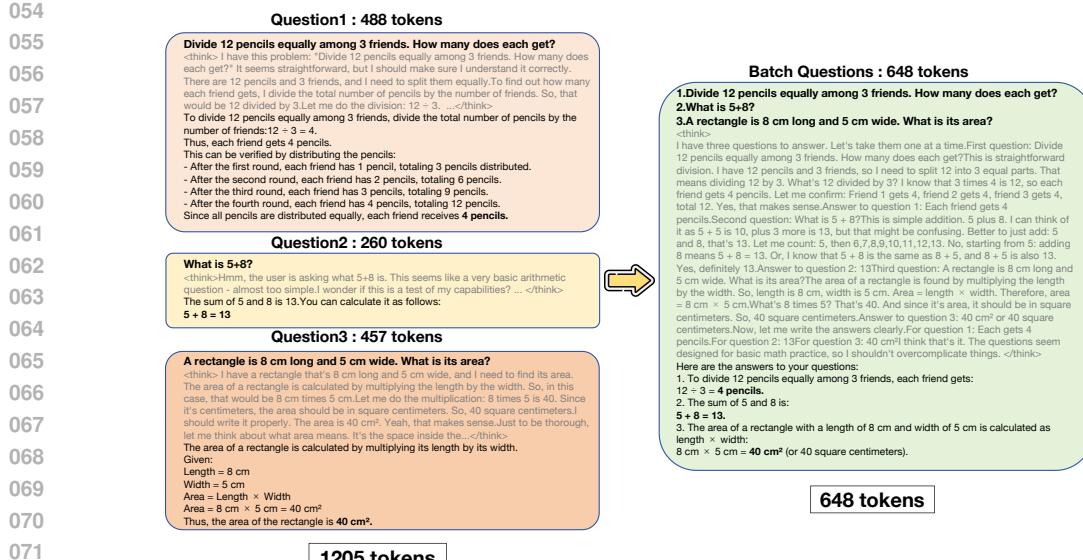
ABSTRACT

013 Reasoning large language models (RLLMs), such as OpenAI-O3 and DeepSeek-
014 R1, have recently demonstrated remarkable capabilities by performing structured
015 and multi-step reasoning. However, recent studies reveal that RLLMs often suf-
016 fer from overthinking, i.e., producing unnecessarily lengthy reasoning chains
017 even for simple questions, leading to excessive token consumption and computa-
018 tional inefficiency. Interestingly, we observe that when processing multiple ques-
019 tions in batch mode, RLLMs exhibit more resource-efficient behavior by dynam-
020 ically compressing reasoning steps for easier problems, due to implicit resource
021 competition. Inspired by this, we propose *Dynamic Reasoning Quota Alloca-
022 tion (DRQA)*, a novel method that transfers the benefits of resource competition
023 from batch processing to single-question inference. Specifically, DRQA leverages
024 batch-generated preference data and reinforcement learning to train the model to
025 allocate reasoning resources adaptively. By encouraging the model to internalize
026 a preference for responses that are both accurate and concise, DRQA enables it to
027 generate concise answers for simple questions while retaining sufficient reason-
028 ing depth for more challenging ones. Extensive experiments on a wide range of
029 mathematical and scientific reasoning benchmarks demonstrate that DRQA sig-
030 nificantly reduces token usage while maintaining, and in many cases improving,
031 answer accuracy. By effectively mitigating the overthinking problem, DRQA of-
032 fers a promising direction for more efficient and scalable deployment of RLLMs,
033 and we hope it inspires further exploration into fine-grained control of reasoning
034 behaviors.
035

1 INTRODUCTION

036 Reasoning large language models (RLLMs), such as OpenAI-O3 (OpenAI, 2025) and DeepSeek-R1
037 (DeepSeek-AI et al., 2025), have recently showcased remarkable capabilities in complex problem
038 solving and decision-making, achieving state-of-the-art performance across a wide range of tasks.
039 However, recent studies have revealed that LLMs often generate unnecessarily lengthy reasoning
040 chains, even for simple questions like “2+3=?” (Sui et al., 2025; Chen et al., 2025). While extended
041 reasoning can improve accuracy on complex tasks, this tendency to *overthink* leads to excessive
042 token usage and growing computational and economic costs, posing significant challenges for the
043 scalable and practical deployment of RLLMs in real-world scenarios.
044

045 Inspired by recent findings on instruction-tuned LLMs (Lin et al., 2024; Cheng et al., 2023), which
046 show that processing multiple inputs together during *batch inference* can reduce the total generated
047 length compared to answering them individually, we investigate whether a similar phenomenon
048 exists in RLLMs. Our study reveals that this effect in RLLMs goes beyond mere solution shortening:
049 batch inference also *compresses the chain-of-thought reasoning process* itself. For example, as
050 shown in Figure 1, answering three questions together yields only 648 tokens in total, compared to
051 1205 tokens when answered separately. This suggests that under a shared context window, questions
052 implicitly compete for a global reasoning quota, prompting the model to prioritize essential logic
053 and suppress redundancy, an effect we refer to as “*resource competition pressure*”.



even for simple questions. To investigate whether batch inference can encourage more efficient reasoning, we conduct a series of controlled experiments. Specifically, we randomly select 500 samples from the DeepScaleR dataset (Luo et al., 2025c) and evaluate several mainstream LLMs under two settings: (i) querying one question at a time (Vanilla), and (ii) querying two questions per prompt (Batch-2). As shown in Table 1, models including DeepSeek-R1 (DeepSeek-AI et al., 2025), Qwen3-32B (think) (Yang et al., 2025a), and Doubao-Seed-1.6 (Seed, 2025) consistently generate shorter outputs in the ‘Batch-2’ setting, suggesting that batch inference naturally promotes more concise reasoning and that this effect generalizes well across different model architectures.

Table 1: Comparison of average output token lengths across different models under the ‘Vanilla’ and ‘Batch-2’ settings.

Model	Vanilla	Batch-2
Deepseek-R1	5640.4	4035.2
Qwen3-32B (think)	7761.6	5274.7
Doubao-Seed-1.6	5288.1	3898.2

Scaling Up Batch Size Further Enhances Efficiency. To further analyze the effect, we vary the batch size using DeepSeek-R1 as a case study, testing batches of **2, 3, 5, 10** and **15** questions. As shown in Figure 2, increasing the batch size leads to a continuous and substantial reduction in the average output length per question. Notably, this compression is achieved with only minimal degradation in answer accuracy, **We hypothesize that this phenomenon stems from an *attention budget* mechanism under context constraints. When processing multiple queries simultaneously, the shared context window acts as a soft bottleneck. To maintain coherence across distinct logical streams, the model is implicitly forced to prioritize high-saliency tokens (core reasoning steps) while suppressing low-information tokens (redundant verbiage). This suggests that ‘resource competition’ acts as a context-driven information bottleneck, triggering the model’s latent capability to compress reasoning without losing semantic integrity.** We refer to this emergent behavior as *resource competition pressure*.

These findings provide compelling empirical evidence that RLLMs are capable of implicit reasoning compression when facing context constraints. The behavior of allocating reasoning resources based on task complexity, without any explicit instruction, points to a promising direction for mitigating the overthinking problem commonly observed in single-question inference. Building on this insight, our work is driven by a central research question: *can we transfer the benefits of resource competition from batch inference to single-question settings?* If so, models could learn to reason adaptively, producing concise answers for simple queries while maintaining sufficient reasoning depth for more complex ones. To this end, we introduce Dynamic Reasoning Quota Allocation (DRQA), detailed in the following section.

3 METHODOLOGY

Our goal is to enable RLLMs to assess question complexity and allocate reasoning resources adaptively, even when processing a single query. Ideally, the model should generate short responses for simple problems while preserving sufficient reasoning depth for more challenging ones, thereby improving inference efficiency without compromising answer accuracy. A key challenge in realizing this capability lies in how to effectively transfer “resource competition pressure” from batch inference to single-question settings. We first explore a straightforward solution via supervised fine-tuning (SFT) using batch-generated data. However, this approach revealed inherent limitations in teaching the model to internalize conciseness as a quality criterion. Inspired by recent advancements in Reinforcement Learning with Verifiable Rewards (RLVR) (Lambert et al., 2025; DeepSeek-AI et al., 2025), we introduce Dynamic Reasoning Quota Allocation (DRQA), a reinforcement learning framework that explicitly encourages reasoning that is both accurate and concise. By optimizing an intrinsic reward aligned with these dual objectives, DRQA guides models to dynamically allocate reasoning resources, enabling more efficient and adaptive inference.

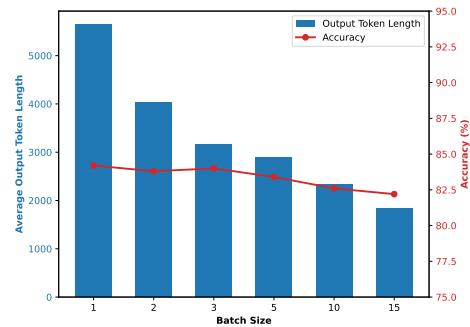


Figure 2: Impact of batch size on output length and accuracy (DeepSeek-R1).

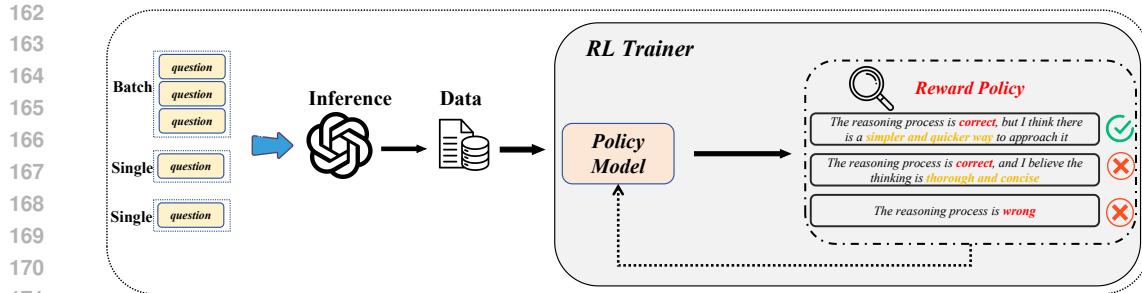


Figure 3: The pipeline of Dynamic Reasoning Quota Allocation (DRQA). Batched questions are input to LLM, producing reasoning chains labeled as A/B/C. Reinforcement learning trains the model to prefer concise and accurate reasoning for efficient resource allocation.

Table 2: Single-question evaluation results of Qwen3-8B after SFT with data generated by batch inference. Batch-X denotes fine-tuning with data from batches of X questions, and Vanilla refers to the original model without SFT.

Method	GSM8K		Math500		AIME2024		GPQA-Diamond		AMC		AIME2025		Overall	
	Acc	tokens	Acc	tokens	Acc	tokens	Acc	tokens	Acc	tokens	Acc	tokens	Acc	tokens
Vanilla	95.67	1878.55	96.00	5270.58	74.67	15468.23	66.67	8685.21	97.50	8608.85	63.33	18058.65	82.31	9661.68
Batch-2	96.67	575.64	95.00	2359.21	57.33	11100.55	53.54	6874.42	90.00	4136.58	45.33	13130.95	72.98	6362.89
Batch-3	93.33	437.23	82.67	1593.53	26.00	5685.36	55.56	3555.65	77.50	4098.10	28.00	7400.53	60.51	3795.07
Batch-5	93.33	336.81	69.67	434.50	9.33	2486.77	46.46	1190.23	42.50	922.25	7.33	2365.41	44.77	1289.33

3.1 SUPERVISED FINE-TUNING WITH BATCH DATA

Our initial approach to transferring the benefits of resource competition into single-question inference is based on imitation learning, where we apply supervised fine-tuning (SFT) to mimic the efficient reasoning patterns exhibited by models during batch inference.

Method We use DeepSeek-R1 (DeepSeek-AI et al., 2025) to perform batch inference over multiple questions sampled from DeepScaleR (Luo et al., 2025c) and collect the generated responses, which are consistently more concise than those from single-question inference. Based on these results, we construct a dataset of “question–concise answer” pairs and apply full-parameter SFT on a Qwen3-8B (Yang et al., 2025a), with the goal of teaching it to generate similarly concise responses in single-question scenarios.

Experimental Results and Analysis We evaluate the fine-tuned models on a comprehensive set of reasoning benchmarks, including GSM8K (Cobbe et al., 2021), MATH-500 (Hendrycks et al., 2021), AIME 2024 (MAA Committees), GPQA-Diamond (Rein et al., 2023), AMC (AI-MO, 2024), and AIME 2025 (MAA Committees). The results shown in Table 2 indicate that SFT does lead to substantial reductions in output length. For example, on GSM8K, the average response length drops from 1878.55 to 575.64 tokens, a 69.36% reduction, demonstrating that overthinking is mitigated to some extent.

However, the efficiency gains come at a considerable cost to accuracy, particularly on more challenging tasks. As shown in Table 2, Models fine-tuned with two-question batch data show a slight accuracy increase from 95.67% to 96.67% on GSM8K, while on MATH-500 accuracy drops from 96.00% to 95.00%, a decrease of 1.00% compared to vanilla prompting. More notably, the performance degradation becomes increasingly severe with higher batch sizes and task complexity. On AIME 2024, accuracy falls from 74.67% (Vanilla) to 57.33% (Batch-2), 26.00% (Batch-3), and just 9.33% (Batch-5). These results suggest the emergence of catastrophic forgetting (Luo et al., 2025d): in attempting to mimic the surface-level conciseness of batch responses, the model compromises its ability to perform the deeper, more nuanced reasoning necessary for solving complex problems.

In summary, while supervised fine-tuning with batch data effectively mitigates overthinking and improves inference efficiency, it comes at the cost of reasoning accuracy, especially on complex tasks,

216 highlighting its limitations for real-world deployment. These shortcomings underscore the need for
 217 a more principled solution that can balance conciseness with reasoning depth, which motivates our
 218 proposed method: Dynamic Reasoning Quota Allocation (DRQA).

220 **3.2 DYNAMIC REASONING QUOTA ALLOCATION**

222 Rather than imitating outputs from batch inference, we aim to endow the model with an intrinsic
 223 ability to evaluate and generate reasoning chains that are both accurate and concise. To this end, we
 224 propose Dynamic Reasoning Quota Allocation (DRQA), a reinforcement learning framework that
 225 enables RLLMs to dynamically allocate reasoning resources in single-question inference.

227 **Core Idea** The core idea of DRQA is to enhance the model’s intrinsic reasoning capabilities by
 228 equipping it with the ability to evaluate the quality of its own reasoning chains. Specifically, the
 229 model is trained to make two key judgments: (i) whether a given reasoning chain is logically correct,
 230 and (ii) if correct, whether it is unnecessarily verbose. *By developing this self-evaluation ability, the*
 231 *model learns to strike a balance between accuracy and conciseness during generation, effectively*
 232 *realizing adaptive resource allocation.*

234 **Preference Data Construction** To train this evaluation ability, we construct a preference dataset
 235 consisting of multiple-choice question-answering samples. Each sample contains a question, a
 236 model-generated chain of thought (CoT), and three evaluation options that reflect different levels
 237 of reasoning quality:

- 238 • **A:** The reasoning process is correct, but I think there is a simpler and quicker way to approach it.
- 239 • **B:** The reasoning process is correct, and I believe the thinking is thorough and concise.
- 240 • **C:** The reasoning process is wrong.

242 The dataset construction process involves three key steps. First, for ease of evaluation, we select
 243 all questions in the DeepScaleR (Luo et al., 2025c) dataset whose answers are numbers of various
 244 types, resulting in approximately 30,000 samples. Second, for each question, we generate two types
 245 of reasoning chains using DeepSeek-R1 (DeepSeek-AI et al., 2025): (1) *vanilla CoTs* obtained by
 246 prompting the model with individual questions, and (2) *batch CoTs* generated by prompting the
 247 model with batched questions, followed by extracting the corresponding reasoning chain for each
 248 question. Finally, we assign labels based on reasoning correctness and conciseness: for vanilla CoTs,
 249 we label **A** if the reasoning is correct, and **C** if incorrect; for batch CoTs, we label **B** if the reasoning
 250 is correct, and **C** if incorrect. This labeling scheme enables the model to learn nuanced distinctions
 251 between correct-but-verbose reasoning (option A), correct-and-concise reasoning (option B), and
 252 incorrect reasoning (option C), thereby developing a clearer understanding of what constitutes a
 253 high-quality reasoning chain.

254 **Reinforcement Learning Framework** We use Group Relative Policy Optimization
 255 (GRPO) (Shao et al., 2024) to train the model to accurately classify each reasoning chain as
 256 A, B, or C, thus encouraging concise and accurate reasoning. Formally, the GRPO objective is
 257 defined as maximizing the likelihood of selecting the correct evaluation label:

$$259 \mathcal{L}_{\text{GRPO}}(\theta) = \mathbb{E}_{\tau \sim \mathcal{D}} \left[\sum_{a \in \mathcal{G}} \log \pi_{\theta}(a | s) \hat{A}(a, s, a^*) - \beta \text{KL}(\pi_{\theta} \| \pi_{\text{old}}) \right] \quad (1)$$

262 where $\tau \sim \mathcal{D}$ denotes a sample from the dataset, with state s representing the question, reasoning
 263 chain, and multiple-choice options (A, B, C); a^* is the ground-truth label; $\mathcal{G} = \{A, B, C\}$ is the set
 264 of actions; $\hat{A}(a, s, a^*)$ is the relative advantage estimate, positive if $a = a^*$ and negative otherwise;
 265 $\text{KL}(\pi_{\theta} \| \pi_{\text{old}})$ is the KL divergence between the current and old policies, constrains the policy update;
 266 and β is a regularization coefficient balancing learning efficiency and policy stability. This training
 267 objective encourages the model to assign higher probabilities to correct judgments while mitigating
 268 the risk of catastrophic forgetting caused by over-updating, a common issue encountered in SFT. As
 269 a result, the model gradually internalizes a preference for reasoning chains that are both correct and
 concise.

270 **Summary** DRQA enables the model to move beyond surface-level imitation and develop an intrinsic, reward-driven preference for high-quality reasoning. By balancing accuracy and conciseness, 271 the model learns to allocate reasoning resources more effectively, addressing the limitations of SFT 272 and supporting more efficient and adaptive inference in single-question settings.

273
274 **4 EXPERIMENTS**

275
276 In this section, we systematically evaluate the performance of the proposed DRQA algorithm, 277 focusing on its ability to balance reasoning accuracy and efficiency. We compare DRQA against a 278 range of strong baselines and provide an in-depth analysis of the results.

279
280 **4.1 EXPERIMENTAL SETUP**

281
282 **Models** We evaluate all methods using three widely adopted distilled models: DeepSeek-R1- 283 Distill-Qwen-1.5B, DeepSeek-R1-Distill-Qwen-7B, and DeepSeek-R1-Distill-Llama-8B. All 284 models are derived from the more powerful DeepSeek-R1 (DeepSeek-AI et al., 2025) through large-scale 285 distillation, offering a favorable trade-off between computational efficiency and reasoning capability.

286
287 **Datasets** For training, we use the dataset described in Section 3.2, constructed by performing 288 batch inference with DeepSeek-R1 on the DeepScaleR (Luo et al., 2025c) training set. This process 289 yields over 50,000 multiple-choice examples annotated with reasoning quality labels.

290
291 **Baselines** To assess the effectiveness of DRQA, we compare it against a comprehensive set of 292 strong baselines approaches (refer to Appendix A for detailed descriptions of the baselines). All 293 baselines are either publicly released or carefully reproduced according to their original protocols. **We note that these baselines span different paradigms, methods like AutoL2S train base 294 models for efficient reasoning, while others and our DRQA focus on compressing or adapting 295 powerful existing RLLMs. We include this diverse set to map the full landscape of efficient 296 reasoning techniques.**

297
298 **Evaluation** We evaluate the performance of different methods across a diverse set of benchmarks. 299 For mathematical reasoning, we include GSM8K (Cobbe et al., 2021), MATH-500 (Hendrycks et al., 300 2021), AIME 2024 and 2025 (MAA Committees), and AMC 2023 (AI-MO, 2024). For domain- 301 specific scientific reasoning, we use the high-quality GPQA-diamond subset (Rein et al., 2023). 302 Detailed descriptions of these datasets are provided in Appendix B. We use both accuracy and 303 response length as evaluation metrics and report the average performance across all test sets. For the 304 AIME datasets, which contain only 30 questions each, we repeatedly sample 5 responses for each 305 case and report the average results to ensure more stable and reliable evaluation.

306
307 All models are evaluated using a unified inference configuration to ensure fair comparison. Experiments 308 are conducted with the vLLM framework on a computing cluster equipped with eight A800 309 (40GB) GPUs. The inference parameters are set to a temperature of 0.6 and a maximum generation 310 length of 32K tokens.

311
312 **Training Details** We use verl (Sheng et al., 2024) as the training framework. We set the batch 313 size to 256, the number of rollouts to 16, the learning rate to 1×10^{-6} , and the maximum response 314 length to 16K tokens. The model is trained for one epoch, consisting of 204 steps in total.

315
316 **4.2 MAIN RESULTS**

317
318 As shown in Table 3, DRQA demonstrates clear superiority in both answer accuracy and response 319 efficiency across all mathematical benchmarks. For example, on GSM8K with the 1.5B model, 320 DRQA achieves an accuracy of 86.67%, outperforming the vanilla baseline by 2 percentage points, 321 while reducing average token usage from 1928.96 to 1427.63, a 25.9% reduction. Similar patterns 322 are observed on more challenging datasets such as AIME 2024 and MATH-500, where DRQA 323 maintains high accuracy while significantly reducing output length. These results highlight DRQA’s 324 effectiveness in dynamically allocating reasoning resources, enabling it to strike a favorable balance 325 between accuracy and efficiency across tasks of varying difficulties. Moreover, DRQA demonstrates 326

Table 3: Performance of different methods using three RLLMs: DeepSeek-R1-Distill-Qwen-1.5B, DeepSeek-R1-Distill-Qwen-7B, and DeepSeek-R1-Distill-Llama-8B. DRQA achieves competitive or superior accuracy while greatly reducing token usage across all datasets and model variants, striking an excellent balance between performance and efficiency.

Method	GSMBK		MATH-500		AIME 2024		GPQA-Diamond		AMC 2023		AIME 2025		Overall	
	Acc	Tokens	Acc	Tokens	Acc	Tokens	Acc	Tokens	Acc	Tokens	Acc	Tokens	Acc _{All}	Tokens _{All}
DeepSeek-R1-Distill-Qwen-1.5B														
<i>Vanilla</i>	84.67%	1928.96	83.33%	5536.14	28.67%	14394.61	30.84%	14731.59	72.50%	8830.10	23.67%	15323.3	53.95%	10124.12
<i>O1-Pruner</i>	74.80%	458	82.20%	3212	28.90%	10361	-	-	-	-	-	-	-	-
<i>DAST</i>	77.20%	586	83.00%	2428	26.90%	7745	-	-	-	-	-	-	-	-
<i>ShortBetter</i>	63.67%	107.86	60.33%	1186.27	11.33%	2935.68	21.72%	1433.95	57.50%	1260.43	12.67%	3326.22	37.87% _{-16.08}	1708.40 _{-83.13%}
<i>AdaptiThink</i>	86.00%	324.26	83.67%	1244.98	29.33%	7044.06	29.80%	4744.23	72.50%	2441.45	24.67%	7490.79	54.33% _{-0.38}	3881.63 _{-61.66%}
<i>GRPO</i>	87.33%	1691.19	84.67%	5743.01	32.67%	15015.74	27.78%	13809.53	77.50%	9378.21	24.00%	13082.98	55.66% _{-1.71}	9787.08 _{-3.33%}
<i>GRPO+Length Penalty</i>	86.00%	722.34	84.67%	2479.14	24.67%	9011.46	26.76%	6184.50	67.50%	3130.51	22.00%	9782.34	51.93% _{-2.01}	5212.38 _{-48.52%}
<i>SFT</i>	81.67%	2296.54	80.33%	5645.95	25.33%	21337.44	27.27%	18540.94	65.00%	8806.48	19.33%	20258.82	49.82% _{-4.13}	12784.36 _{-26.28%}
<i>DRQA(our)</i>	86.67%	1427.63	84.67%	3488.08	32.00%	11008.31	31.81%	9148.83	75.00%	5355.03	24.00%	10382.12	55.69% _{-1.74}	6801.67 _{-32.82%}
DeepSeek-R1-Distill-Qwen-7B														
<i>Vanilla</i>	91.33%	1735.5	90.40%	5099.95	53.33%	13712.6	48.98%	13313.92	90.00%	6349.53	40.00%	14248.11	69.01%	9076.60
<i>DAST</i>	86.70%	459	89.60%	2162	45.60%	7578	-	-	-	-	-	-	-	-
<i>O1-Pruner</i>	87.60%	428	86.60%	2534	49.20%	9719	-	-	-	-	-	-	-	-
<i>Dynasor-CoT</i>	89.60%	1285	89.00%	2971	46.70%	12695	30.50%	7639	85.00%	5980	-	-	-	-
<i>DEER</i>	90.60%	6917	89.80%	2143	49.20%	9839	31.30%	5469	85.00%	4451	-	-	-	-
<i>ShortBetter</i>	70.00%	112.86	68.00%	623.44	41.33%	5005.96	43.43%	1811.43	57.50%	1567.50	30.67%	5593.96	51.82% _{-17.19}	2419.19 _{-73.35%}
<i>AdaptiThink</i>	89.67%	296.94	91.67%	1839.59	54.00%	9894.05	51.52%	7128.95	87.50%	3287.95	39.33%	12454.59	68.95% _{-0.06}	5817.01 _{-35.91%}
<i>Autol2S</i>	93.33%	444.8	83.33%	3113.93	40.67%	6499.32	45.39%	2553.01	85.00%	2613.05	31.33%	3669.53	63.18% _{-5.84}	3148.94 _{-65.31%}
<i>GRPO</i>	93.67%	1524.24	92.00%	4532.21	54.67%	12013.92	47.47%	12124.10	87.50%	5103.10	41.33%	12192.12	79.19% _{-4.43}	7919.45 _{-12.75%}
<i>GRPO+Length Penalty</i>	91.33%	876.25	91.33%	2751.13	52.00%	7213.11	45.96%	7124	92.50%	3256.02	39.67%	6058.40	68.80% _{-0.21}	4546.49 _{-49.91%}
<i>SFT</i>	92.33%	1317.85	92.00%	3824.43	44.67%	14903.82	46.97%	12385.43	77.50%	5515.59	32.00%	13931.80	64.25% _{-4.76}	8647.15 _{-4.73%}
<i>DRQA(our)</i>	92.67%	1324.24	91.40%	390.74	54.67%	10007.18	49.50%	8988.50	92.50%	4463.03	40.67%	9545.44	70.24% _{-1.23}	6371.85 _{-29.80%}
DeepSeek-R1-Distill-Llama-8B														
<i>Vanilla</i>	91.67%	1829.12	90.00%	5417.41	49.33%	13585.12	48.98%	11845.27	87.50%	7177.73	38.67%	14260.26	67.69%	9019.15
<i>GRPO</i>	92.33%	1605.94	91.67%	4812.02	50.67%	12897.09	46.46%	9869.20	90.00%	7600.58	39.33%	12204.58	68.41% _{-0.72}	8164.90 _{-9.47%}
<i>GRPO+Length Penalty</i>	91.67%	875.66	91.33%	2753.43	48.00%	7192.28	45.96%	7055.54	90.00%	3236.22	38.00%	8040.74	67.49% _{-0.20}	4858.98 _{-46.13%}
<i>SFT</i>	90.67%	1315.83	90.00%	3825.52	44.67%	14881.25	44.95%	10897.06	75.00%	5509.82	32.67%	13915.29	62.99% _{-4.70}	8390.80 _{-6.97%}
<i>DRQA(our)</i>	93.00%	1594.70	91.33%	4180.83	50.67%	9904.46	8986.63	92.50%	4463.43	39.33%	9542.11	69.47% _{-1.78}	6451.56 _{-28.47%}	

strong generalization on out-of-distribution (OOD) benchmarks, as evidenced by its performance on GPQA-Diamond.

We also compare DRQA with aggressive compression methods such as ShorterBetter (Yi et al., 2025) and DAST (Shen et al., 2025), which can reduce output length even further, for example, generating outputs as short as 107.86 tokens on GSM8K. However, these methods often suffer from severe accuracy degradation, with performance drops exceeding 20 percentage points in some cases. This highlights a key limitation of methods that rely solely on length-based reward signals: they tend to compromise the logical integrity of reasoning chains, limiting their practical applicability.

Notably, DRQA remains highly effective on larger models. On GSM8K with the 7B model, DRQA improves accuracy by 1.34% over the baseline while reducing token usage by 23.6%. On Llama-8B, DRQA achieves a 1.78% accuracy gain while cutting token usage by 28.47%, highlighting its ability to enhance performance and efficiency at larger model scales. Across all benchmarks, it consistently achieves the most favorable trade-off between accuracy and output efficiency. Compared to strong baselines such as DAST (Shen et al., 2025), O1-Pruner (Luo et al., 2025b), Dynasor-CoT (Fu et al., 2025), and DEER (Xia et al., 2024), DRQA not only matches or surpasses them in length reduction but, more importantly, maintains state-of-the-art reasoning accuracy.

Overall, DRQA achieves an average accuracy improvement of 1.58 percentage points and an average token usage reduction of 30.4% across all evaluated benchmarks and all three model variants. These results provide compelling evidence that DRQA effectively transfers the benefits of “resource competition pressure” from batch inference to single-question settings, establishing a strong foundation for the efficient and scalable deployment of RLLMs.

4.3 GENERALIZATION TO CODE GENERATION

We further assess DRQA on the LiveCodeBench benchmark (Jain et al., 2024), a contamination-free suite of code-related tasks collected from competitive programming platforms. Our evaluation uses 342 newly released Python problems spanning September 2024 and April 2025. As shown in Table 4, DRQA consistently reduces token usage by about 23%–29% across all three model sizes, while also improving accuracy. For example, on *DeepSeek-R1-Distill-Qwen-7B*, DRQA shortens outputs from 8724.27 to 6648.77 tokens (-23.79%) and improves accuracy by 1.75%, demonstrating its strong generalizability to the code generation domain.

378
 379 Table 4: Performance on LiveCodeBench. **Include GRPO and GRPO+Length Penalty as additional baselines. DRQA achieves the best accuracy across all model sizes while significantly**
 380 **reducing token usage. While GRPO+Length Penalty achieves extreme shortness, it suffers**
 381 **from accuracy degradation, whereas DRQA maintains a superior balance.**

Method	DeepSeek-R1-Distill-Qwen-1.5B Acc	DeepSeek-R1-Distill-Qwen-1.5B Tokens	DeepSeek-R1-Distill-Qwen-7B Acc	DeepSeek-R1-Distill-Qwen-7B Tokens	DeepSeek-R1-Distill-Llama-8B Acc	DeepSeek-R1-Distill-Llama-8B Tokens
Vanilla	13.16%	11261.72	30.70%	8724.27	31.87%	9012.31
GRPO	13.45%	10845.20	31.29%	8412.33	32.16%	8640.12
GRPO+Length Penalty	11.99%	6514.50	29.24%	5920.45	30.12%	5890.76
DRQA(our)	13.74%	8124.20	32.45%	6648.77	32.75%	6426.68

388
 389 Table 5: Ablation experiments across different training paradigms.
 390

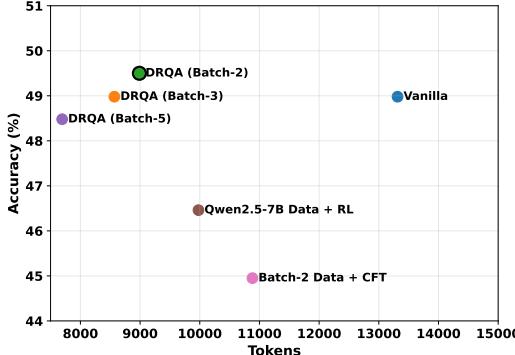
Method	GSM8K		MATH-500		AIME 2024		Overall	
	Acc	tokens	Acc	tokens	Acc	tokens	Acc	tokens
Vanilla	91.33%	1735.5	90.40%	5099.95	53.33%	13712.6	78.35%	6849.35
DRQA (Batch-2)	92.67%	1324.24	91.33%	3902.74	54.67%	10007.18	79.58% _{+1.23}	5078.05 _{-25.86%}
DRQA (Batch-3)	91.67%	1212.59	90.20%	3311.20	53.33%	8805.24	78.40% _{+0.05}	4443.01 _{-35.13%}
DRQA (Batch-5)	90.67%	1158.88	89.80%	2675.81	49.33%	7366.80	76.60% _{-1.75}	3733.83 _{-45.49%}
Qwen2.5-7B Data + RL	90.00%	1434.65	89.60%	3313.12	50.67%	12190.59	76.76% _{-1.60}	5646.12 _{-17.57%}
Batch-2 Data + CFT	89.67%	1361.00	88.20%	3973.54	49.66%	10012.55	75.84% _{-2.51}	5115.70 _{-25.31%}

397
 398
 399

4.4 ABLATION STUDY

400
 401 To thoroughly assess the contribution of each core component in DRQA, we conduct a series of
 402 ablation studies that isolate the effects of different training paradigms and input conciseness on
 403 reasoning performance and efficiency. All experiments are performed using the same benchmark
 404 datasets, evaluation metrics, and base model (DeepSeek-R1-Distill-Qwen-7B) as in the main study,
 405 with consistent inference configurations to ensure fair comparison.406
 407 **Effect of Batch Size in DRQA Data Construction.** We investigate the impact of different batch
 408 sizes on model performance. Specifically, we construct preference datasets by prompting DeepSeek-
 409 R1 with batches of 2, 3, or 5 questions, then splitting the outputs into individual reasoning chains
 410 for downstream RL training. This design allows us to analyze how increasing levels of resource
 411 competition influence both answer accuracy and response efficiency within the DRQA framework.412
 413 **Replacing Batch Reasoning Data with Qwen2.5-7B Concise Chains** To evaluate the impor-
 414 tance of batch-induced resource competition, we consider an alternative setting where the prefer-
 415 ence dataset is constructed using concise reasoning chains generated directly by Qwen2.5-7B (Qwen
 416 et al., 2025), without leveraging batch inference. This comparison allows us to disentangle the ef-
 417 fects of resource-driven compression from those achieved solely through the model’s inherent ability
 418 to generate concise outputs.419
 420 **Critique Fine-Tuning with Preference data** Beyond reinforcement learning, we also evaluate
 421 the Critique Fine-Tuning (CFT) paradigm (Wang et al., 2025) as an alternative training strategy.
 422423
 424

4.4.1 RESULTS AND ANALYSIS

425
 426 Table 5 presents the results of our ablation study. As batch size increases, the model pro-
 427 duces increasingly concise outputs, with token
 428 usage reduced by up to 45% for larger batches.
 429 However, this efficiency gain comes at the cost
 430 of declining accuracy, highlighting a trade-off
 431 between efficiency and correctness. Notably,
 432 a batch size of 2 achieves the best balance,
 433 improving accuracy while significantly reduc-8
 Figure 4: The efficiency-accuracy trade-off on GPQA-diamond for DRQA and ablation variants.

432 ing token consumption compared to the vanilla
 433 baseline.

434 When compared to concise reasoning chains generated directly by Qwen2.5-7B Qwen et al. (2025)
 435 without batch inference, we observe that only batch-induced compression achieves both high effi-
 436 ciency and strong accuracy. Similarly, while Critique Fine-Tuning helps reduce output length, it
 437 leads to a notable accuracy drop, underscoring the importance of reinforcement learning for pre-
 438 serving reasoning quality. Figure 4 further supports these insights, showing that DRQA achieves the
 439 best overall trade-off on the OOD dataset GPQA-Diamond, highlighting its robustness across both
 440 in-distribution and out-of-distribution scenarios.

442 5 RELATED WORK

444 5.1 REASONING LARGE LANGUAGE MODELS

446 Recent advances in reasoning large language models (RLLMs), such as OpenAI-O3 (OpenAI,
 447 2025), Deepseek-R1 (DeepSeek-AI et al., 2025), and QwQ (Team, 2025) leverage chain-of-
 448 thought (Wei et al., 2023) for step-by-step reasoning, achieving state-of-the-art performance across
 449 tasks including mathematical reasoning, coding, and complex question answering. CoT allows
 450 these models to leverage inference-time scaling by generating multiple reasoning steps that ex-
 451 plore alternative solution paths, thereby significantly enhancing accuracy over single-pass gener-
 452 ation. To further improve correctness, a variety of methods have been proposed, including self-
 453 consistency (Wang et al., 2023), beam search (Yao et al., 2023), and reinforcement learning-based
 454 post-training (DeepSeek-AI et al., 2025), which encourage iterative self-reflection and help reduce
 455 logical errors. Additional search-based approaches, such as Monte Carlo Tree Search (MCTS) (Gao
 456 et al., 2024), have been employed to expand the scope of exploration in complex problem-solving
 457 scenarios. Our work focuses on further improving the efficiency of such reasoning models.

458 5.2 EFFICIENT REASONING

460 Reasoning efficiency in RLLMs (Qu et al., 2025; Sui et al., 2025) refers to balancing task quality
 461 and computational cost. Models like OpenAI-O3 (OpenAI, 2025) and DeepSeek-R1 (DeepSeek-AI
 462 et al., 2025) often generate too long and redundant reasoning chains, over explaining simple prob-
 463 lems while sometimes offering shallow reasoning for complex ones. Main approaches for improving
 464 efficiency include:

- 466 • **Inference time control:** Methods such as TALE (Han et al., 2025), DEER (Yang et al., 2025b)
 467 apply token budgets or early exit strategies inspired by dual-system theory.
- 468 • **Chain compression and supervised tuning:** TokenSkip (Xia et al., 2025), CoT-Valve (Ma et al.,
 469 2025), and AutoL2S (Luo et al., 2025a) use supervised fine-tuning or distillation to shorten rea-
 470 soning chains, often improving conciseness but sometimes at the expense of complex reasoning.
- 471 • **Reinforcement learning approaches:** DAST (Shen et al., 2025), O1-Pruner (Luo et al., 2025b),
 472 and S-GRPO (Dai et al., 2025) introduce reward functions to penalize lengthy outputs and promote
 473 token efficiency, supporting adaptive reasoning with little loss of accuracy.

474 These methods largely depend on fixed budgets or hand crafted rewards. Our DRQA instead trans-
 475 fers the ‘resource competition pressure’ observed in batch inference to single-question settings, en-
 476 abling models to automatically adjust reasoning length according to problem complexity, providing
 477 brief responses for simple questions and detailed explanations for challenging ones without manual
 478 constraints.

481 5.3 BATCH PROMPTING AND RESOURCE COMPETITION

483 Batch prompting (Cheng et al., 2023) was originally proposed to improve inference throughput and
 484 reduce API costs by grouping multiple samples into a single prompt, allowing shared instructions
 485 and few-shot exemplars. Subsequent works have optimized this paradigm to address stability and
 performance issues:

486 • **Robustness and ordering:** Lin et al. (2024) introduced permutation and self-consistency mechanisms to mitigate position bias and performance degradation often observed in batched inputs.
 487 • **Demonstration utilization:** Feng et al. (2024) proposed “Auto-Demo Prompting,” which leverages generated outputs from earlier queries within a batch as demonstrations for subsequent ones to enhance performance.
 488
 489
 490

491 These methods primarily view batching as an **inference-time optimization** technique to amortize
 492 computational overhead across multiple queries. Our work draws inspiration from the context
 493 constraints observed in these studies but fundamentally differs in objective and mechanism. While prior
 494 works aim to maintain performance while maximizing batch size for throughput, DRQA identifies
 495 that the “resource competition” inherent in batching naturally suppresses redundancy in reasoning
 496 chains. Instead of using batching solely for inference speedup, we leverage it as a **data generation**
 497 **mechanism** for training. By teaching the model to internalize this concise reasoning pattern via
 498 reinforcement learning, DRQA transfers the efficiency benefits of batching to **single-question inference**,
 499 enabling dynamic reasoning quota allocation without requiring batched inputs at test time.
 500
 501

502 6 CONCLUSION

503 This paper introduces Dynamic Reasoning Quota Allocation (DRQA), a novel approach aimed at
 504 addressing the overthinking problem in reasoning large language models (RLLMs). Motivated by
 505 the observation that resource competition pressure in batch inference naturally encourages efficient
 506 reasoning, DRQA leverages batch-generated data and reinforcement learning to transfer the benefits
 507 of resource competition from batch inference to single-question scenarios. Specifically, the model is
 508 trained to develop an internal preference for reasoning processes that balance conciseness with
 509 accuracy, allowing it to produce short answers for straightforward questions while preserving adequate
 510 reasoning depth when tackling more complex ones. Extensive experimental results and analysis
 511 show that DRQA significantly reduces token consumption while maintaining, or even improving,
 512 accuracy. By effectively alleviating overthinking, DRQA offers a new direction for more efficient
 513 and scalable deployment of RLLMs.
 514

516 7 ETHICS STATEMENT

517 This work adheres to the ICLR Code of Ethics and follows responsible research practices. Our study
 518 focuses on improving the efficiency of reasoning in large language models (LLMs) by mitigating
 519 overthinking through the proposed Dynamic Reasoning Quota Allocation (DRQA) framework.
 520

521 The research does **not** involve human subjects, personal data, or sensitive information. All datasets
 522 used (e.g., GSM8K, MATH-500, AIME, AMC, GPQA-Diamond, LiveCodeBench, DeepScaleR)
 523 are publicly available, widely adopted in the AI community, and contain no personally identifiable
 524 information; data usage strictly complies with their respective licenses.
 525

526 Our experiments are performed entirely in silico, and the outputs are automatically generated by
 527 models without human intervention. The proposed method is designed to **reduce computational**
 528 **cost and energy usage** by shortening reasoning chains for simple queries while retaining depth for
 529 complex ones, thereby contributing positively to environmental sustainability.
 530

531 We have considered possible risks, including unintended accuracy degradation in complex tasks or
 532 misuse of the system for harmful automated decision making. Mitigation strategies include thorough
 533 benchmark evaluation across diverse domains, public release of methodology for reproducibility,
 534 and clear documentation of model limitations.
 535

536 The study promotes fairness and avoids discrimination by focusing on general-purpose mathematical
 537 and scientific datasets with balanced coverage; no content in our dataset or method is intended to
 538 target or disadvantage any demographic group.
 539

540 All code, data handling, and result reporting are conducted with scientific integrity, transparency,
 541 and reproducibility in mind. We believe this work supports the responsible advancement of AI that
 542 serves the public good, efficiency, and well-being.
 543

540 **8 REPRODUCIBILITY STATEMENT**

541

542 We have implemented several measures to ensure that all reported results are fully reproducible. A
 543 comprehensive description of the proposed DRQA algorithm, including the procedures for batch
 544 data collection, preference dataset construction, and the reinforcement learning framework, is pro-
 545 vided in 3.2. The precise model variants, datasets, and evaluation metrics are specified in section 4
 546 and in Appendix B. Inference hyperparameters and training configurations, such as the optimizer
 547 type, batch size, learning rate, number of rollouts, and maximum generation length, are detailed in
 548 section 4.1.

549 Two code modules are provided as supplementary material. The first module, `ver1/`, contains
 550 the implementation of DRQA training using Group Relative Policy Optimization (GRPO). The sec-
 551 ond module, `eval/`, contains scripts for generating training data via batch inference, conducting
 552 evaluations on all benchmarks, and reproducing the tables and figures reported in the paper.

553 The supplementary code package includes complete, end-to-end instructions for dataset prepro-
 554 cessing, model training, and evaluation. All datasets used in training and evaluation, including
 555 GSM8K, MATH-500, AIME 2024, AIME 2025, AMC 2023, GPQA-Diamond, DeepScaleR, and
 556 LiveCodeBench, are publicly available. Our preprocessing steps are documented in the supple-
 557 mentary material.

558 With the provided source code, configuration files, and dataset references, independent researchers
 559 can exactly reproduce our experiments and validate all results under identical conditions.

560

561 **REFERENCES**

562

563 AI-MO. Amc 2023, 2024. URL <https://huggingface.co/datasets/AI-MO/aimo-validation-amc>.

564

565 Xingyu Chen, Jiahao Xu, Tian Liang, Zhiwei He, Jianhui Pang, Dian Yu, Linfeng Song, Qiuzhi Liu,
 566 Mengfei Zhou, Zhuosheng Zhang, Rui Wang, Zhaopeng Tu, Haitao Mi, and Dong Yu. Do not
 567 think that much for 2+3=? on the overthinking of o1-like llms, 2025. URL <https://arxiv.org/abs/2412.21187>.

568 Zhoujun Cheng, Jungo Kasai, and Tao Yu. Batch prompting: Efficient inference with large language
 569 model apis, 2023. URL <https://arxiv.org/abs/2301.08721>.

570

571 Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser,
 572 Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John
 573 Schulman. Training verifiers to solve math word problems, 2021. URL <https://arxiv.org/abs/2110.14168>.

574 Muzhi Dai, Chenxu Yang, and Qingyi Si. S-grpo: Early exit via reinforcement learning in reasoning
 575 models, 2025. URL <https://arxiv.org/abs/2505.07686>.

576

577 DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu,
 578 Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu,
 579 Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, Bing Xue, Bingxuan Wang, Bochao
 580 Wu, Bei Feng, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan,
 581 Damai Dai, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao,
 582 Guanting Chen, Guowei Li, H. Zhang, Han Bao, Hanwei Xu, Haocheng Wang, Honghui Ding,
 583 Huajian Xin, Huazuo Gao, Hui Qu, Hui Li, Jianzhong Guo, Jiashi Li, Jiawei Wang, Jingchang
 584 Chen, Jingyang Yuan, Junjie Qiu, Junlong Li, J. L. Cai, Jiaqi Ni, Jian Liang, Jin Chen, Kai
 585 Dong, Kai Hu, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang,
 586 Liang Zhao, Litong Wang, Liyue Zhang, Lei Xu, Leyi Xia, Mingchuan Zhang, Minghua Zhang,
 587 Minghui Tang, Meng Li, Miaojun Wang, Mingming Li, Ning Tian, Panpan Huang, Peng Zhang,
 588 Qiancheng Wang, Qinyu Chen, Qiushi Du, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang,
 589 R. J. Chen, R. L. Jin, Ruyi Chen, Shanghai Lu, Shangyan Zhou, Shanhuang Chen, Shengfeng
 590 Ye, Shiyu Wang, Shuiping Yu, Shunfeng Zhou, Shuting Pan, S. S. Li, Shuang Zhou, Shaoqing
 591 Wu, Shengfeng Ye, Tao Yun, Tian Pei, Tianyu Sun, T. Wang, Wangding Zeng, Wanjia Zhao, Wen
 592 Liu, Wenfeng Liang, Wenjun Gao, Wenqin Yu, Wentao Zhang, W. L. Xiao, Wei An, Xiaodong
 593

594 Liu, Xiaohan Wang, Xiaokang Chen, Xiaotao Nie, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu,
 595 Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, X. Q. Li, Xiangyue Jin, Xiaojin Shen, Xi-
 596 aosh Chen, Xiaowen Sun, Xiaoxiang Wang, Xinnan Song, Xinyi Zhou, Xianzu Wang, Xinxia
 597 Shan, Y. K. Li, Y. Q. Wang, Y. X. Wei, Yang Zhang, Yanhong Xu, Yao Li, Yao Zhao, Yaofeng
 598 Sun, Yaohui Wang, Yi Yu, Yichao Zhang, Yifan Shi, Yiliang Xiong, Ying He, Yishi Piao, Yisong
 599 Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Yuan Ou, Yuduan Wang, Yue Gong,
 600 Yuheng Zou, Yujia He, Yunfan Xiong, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou,
 601 Y. X. Zhu, Yanhong Xu, Yanping Huang, Yaohui Li, Yi Zheng, Yuchen Zhu, Yunxian Ma, Ying
 602 Tang, Yukun Zha, Yuting Yan, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhenda
 603 Xie, Zhengyan Zhang, Zhewen Hao, Zhicheng Ma, Zhigang Yan, Zhiyu Wu, Zihui Gu, Zijia Zhu,
 604 Zijun Liu, Zilin Li, Ziwei Xie, Ziyang Song, Zizheng Pan, Zhen Huang, Zhipeng Xu, Zhongyu
 605 Zhang, and Zhen Zhang. Deepseek-r1: Incentivizing reasoning capability in llms via reinforce-
 606 ment learning, 2025. URL <https://arxiv.org/abs/2501.12948>.

607 Longyu Feng, Mengze Hong, and Chen Jason Zhang. Auto-demo prompting: Leveraging generated
 608 outputs as demonstrations for enhanced batch prompting, 2024. URL <https://arxiv.org/abs/2410.01724>.

609 Yichao Fu, Junda Chen, Yonghao Zhuang, Zheyu Fu, Ion Stoica, and Hao Zhang. Reasoning without
 610 self-doubt: More efficient chain-of-thought through certainty probing. In *ICLR 2025 Workshop
 611 on Foundation Models in the Wild*, 2025. URL <https://openreview.net/forum?id=wpK4IMJfdX>.

612 Zitian Gao, Boye Niu, Xuzheng He, Haotian Xu, Hongzhang Liu, Aiwei Liu, Xuming Hu, and
 613 Lijie Wen. Interpretable contrastive monte carlo tree search reasoning, 2024. URL <https://arxiv.org/abs/2410.01707>.

614 Tingxu Han, Zhenting Wang, Chunrong Fang, Shiyu Zhao, Shiqing Ma, and Zhenyu Chen. Token-
 615 budget-aware llm reasoning, 2025. URL <https://arxiv.org/abs/2412.18547>.

616 Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song,
 617 and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset, 2021.
 618 URL <https://arxiv.org/abs/2103.03874>.

619 Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando
 620 Solar-Lezama, Koushik Sen, and Ion Stoica. Livecodebench: Holistic and contamination free
 621 evaluation of large language models for code, 2024. URL <https://arxiv.org/abs/2403.07974>.

622 Nathan Lambert, Jacob Morrison, Valentina Pyatkin, Shengyi Huang, Hamish Ivison, Faeze Brah-
 623 man, Lester James V. Miranda, Alisa Liu, Nouha Dziri, Shane Lyu, Yuling Gu, Saumya Ma-
 624 lik, Victoria Graf, Jena D. Hwang, Jiangjiang Yang, Ronan Le Bras, Oyyind Tafjord, Chris
 625 Wilhelm, Luca Soldaini, Noah A. Smith, Yizhong Wang, Pradeep Dasigi, and Hannaneh Ha-
 626 jishirzi. Tulu 3: Pushing frontiers in open language model post-training, 2025. URL <https://arxiv.org/abs/2411.15124>.

627 Jianzhe Lin, Maurice Diesendruck, Liang Du, and Robin Abraham. Batchprompt: Accomplish more
 628 with less, 2024. URL <https://arxiv.org/abs/2309.00384>.

629 Feng Luo, Yu-Neng Chuang, Guanchu Wang, Hoang Anh Duy Le, Shaochen Zhong, Hongyi Liu,
 630 Jiayi Yuan, Yang Sui, Vladimir Braverman, Vipin Chaudhary, and Xia Hu. Autol2s: Auto long-
 631 short reasoning for efficient large language models, 2025a. URL <https://arxiv.org/abs/2505.22662>.

632 Haotian Luo, Li Shen, Haiying He, Yibo Wang, Shiwei Liu, Wei Li, Naiqiang Tan, Xiaochun Cao,
 633 and Dacheng Tao. O1-pruner: Length-harmonizing fine-tuning for o1-like reasoning pruning,
 634 2025b. URL <https://arxiv.org/abs/2501.12570>.

635 Michael Luo, Sijun Tan, Justin Wong, Xiaoxiang Shi, William Tang, Manan Roongta, Colin Cai,
 636 Jeffrey Luo, Tianjun Zhang, Erran Li, Raluca Ada Popa, and Ion Stoica. Deepscaler: Surpassing
 637 o1-preview with a 1.5b model by scaling rl, 2025c. URL <https://pretty-radio-b75.notion.site>.

648 Yun Luo, Zhen Yang, Fandong Meng, Yafu Li, Jie Zhou, and Yue Zhang. An empirical study
 649 of catastrophic forgetting in large language models during continual fine-tuning, 2025d. URL
 650 <https://arxiv.org/abs/2308.08747>.

651 Xinyin Ma, Guangnian Wan, Runpeng Yu, Gongfan Fang, and Xinchao Wang. Cot-valve: Length-
 652 compressible chain-of-thought tuning, 2025. URL <https://arxiv.org/abs/2502.09601>.

653 MAA Committees. Aime problems and solutions. https://artofproblemsolving.com/wiki/index.php/AIME_Problems_and_Solutions.

654 OpenAI. Introducing O3 and O4 Mini. <https://openai.com/index/introducing-o3-and-o4-mini/>, 2025. Accessed: 2025-04-16.

655 Xiaoye Qu, Yafu Li, Zhaochen Su, Weigao Sun, Jianhao Yan, Dongrui Liu, Ganqu Cui, Daizong
 656 Liu, Shuxian Liang, Junxian He, Peng Li, Wei Wei, Jing Shao, Chaochao Lu, Yue Zhang, Xian-
 657 Sheng Hua, Bowen Zhou, and Yu Cheng. A survey of efficient reasoning for large reasoning
 658 models: Language, multimodality, and beyond, 2025. URL <https://arxiv.org/abs/2503.21614>.

659 Qwen, ;, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan
 660 Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang,
 661 Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin
 662 Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li,
 663 Tianyi Tang, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang,
 664 Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 technical report, 2025.
 665 URL <https://arxiv.org/abs/2412.15115>.

666 David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien
 667 Dirani, Julian Michael, and Samuel R. Bowman. Gpqa: A graduate-level google-proof q&a
 668 benchmark, 2023. URL <https://arxiv.org/abs/2311.12022>.

669 ByteDance Seed. Doubao-seed-1.6, June 2025. URL https://seed/bytedance.com/zh/seed1_6.

670 Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang,
 671 Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. Deepseekmath: Pushing the limits of mathe-
 672 matical reasoning in open language models, 2024. URL <https://arxiv.org/abs/2402.03300>.

673 Yi Shen, Jian Zhang, Jieyun Huang, Shuming Shi, Wenjing Zhang, Jiangze Yan, Ning Wang, Kai
 674 Wang, Zhaoxiang Liu, and Shiguo Lian. Dast: Difficulty-adaptive slow-thinking for large reason-
 675 ing models, 2025. URL <https://arxiv.org/abs/2503.04472>.

676 Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng,
 677 Haibin Lin, and Chuan Wu. Hybridflow: A flexible and efficient rlhf framework. *arXiv preprint*
 678 *arXiv*: 2409.19256, 2024.

679 Yang Sui, Yu-Neng Chuang, Guanchu Wang, Jiamu Zhang, Tianyi Zhang, Jiayi Yuan, Hongyi Liu,
 680 Andrew Wen, Shaochen Zhong, Hanjie Chen, and Xia Hu. Stop overthinking: A survey on effi-
 681 cient reasoning for large language models, 2025. URL <https://arxiv.org/abs/2503.16419>.

682 Qwen Team. Qwq-32b: Embracing the power of reinforcement learning, March 2025. URL
 683 <https://qwenlm.github.io/blog/qwq-32b/>.

684 Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdh-
 685 ery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models,
 686 2023. URL <https://arxiv.org/abs/2203.11171>.

687 Yubo Wang, Xiang Yue, and Wenhui Chen. Critique fine-tuning: Learning to critique is more effec-
 688 tive than learning to imitate, 2025. URL <https://arxiv.org/abs/2501.17703>.

702 Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc
 703 Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models,
 704 2023. URL <https://arxiv.org/abs/2201.11903>.

705 Bo Xia, Yilun Kong, Yongzhe Chang, Bo Yuan, Zhiheng Li, Xueqian Wang, and Bin Liang. Deer: A
 706 delay-resilient framework for reinforcement learning with variable delays, 2024. URL <https://arxiv.org/abs/2406.03102>.

707 Heming Xia, Chak Tou Leong, Wenjie Wang, Yongqi Li, and Wenjie Li. Tokenskip: Control-
 708 lable chain-of-thought compression in llms, 2025. URL <https://arxiv.org/abs/2502.12067>.

709 An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang
 710 Gao, Chengan Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu,
 711 Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin
 712 Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang,
 713 Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui
 714 Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang
 715 Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger
 716 Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan
 717 Qiu. Qwen3 technical report, 2025a. URL <https://arxiv.org/abs/2505.09388>.

718 Chenxu Yang, Qingyi Si, Yongjie Duan, Zheliang Zhu, Chenyu Zhu, Qiaowei Li, Zheng Lin, Li Cao,
 719 and Weiping Wang. Dynamic early exit in reasoning models, 2025b. URL <https://arxiv.org/abs/2504.15895>.

720 Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik
 721 Narasimhan. Tree of thoughts: Deliberate problem solving with large language models, 2023.
 722 URL <https://arxiv.org/abs/2305.10601>.

723 Jingyang Yi, Jiazheng Wang, and Sida Li. Shorterbetter: Guiding reasoning models to find opti-
 724 mal inference length for efficient reasoning, 2025. URL <https://arxiv.org/abs/2504.21370>.

725 Jiajie Zhang, Nianyi Lin, Lei Hou, Ling Feng, and Juanzi Li. Adaptthink: Reasoning models can
 726 learn when to think, 2025. URL <https://arxiv.org/abs/2505.13417>.

727

728

729

730

731

732

733

734

735

736

737

738

739

740

741

742

743

744

745

746

747

748

749

750

751

752

753

754

755

756 **A BASELINE METHODS**
757758 We consider the following baseline methods in our experiments:
759

- 760 • **GRPO**: We train a model on the DeepScaleR (Luo et al., 2025c) dataset using the Group Relative
761 Policy Optimization algorithm, where only answer correctness is used as the reward signal.
- 762 • **GRPO+Length Penalty**: This variant further introduces a length penalty to the reward design:
763 for correct answers, shorter responses yield higher rewards, while for incorrect answers, longer
764 responses incur greater penalties. This encourages the model to produce concise and accurate
765 reasoning.
- 766 • **SFT (Supervised Fine-Tuning)**: We perform full-parameter supervised fine-tuning on the
767 model using question-answer pairs generated via batch inference of Deepseek-R1 on the Deep-
768 ScaleR (Luo et al., 2025c) dataset.
- 769 • **AdaptThink** (Zhang et al., 2025): This approach encourages adaptive selection between direct
770 answer and step-by-step reasoning (Chain-of-Thought) based on question difficulty. Training ob-
771 jectives and sample balancing enable the model to flexibly explore both thinking modes, improv-
772 ing reasoning efficiency and performance.
- 773 • **AutoL2S** (Luo et al., 2025a): A dynamic, model-agnostic framework that annotates each ques-
774 tion with both long and short Chain-of-Thought (CoT) solutions. By marking simple questions
775 with <EASY>, the model is trained to automatically select concise CoT for simple problems and
776 detailed reasoning for complex ones. **Distinct from methods that distill reasoning models, Au-**
777 **tol2S is typically trained starting from a base model (e.g., Qwen2.5-7B) to achieve efficient**
778 **reasoning.**
- 779 • **DAST** (Shen et al., 2025): DAST explicitly quantifies problem difficulty via a token length budget
780 and employs a reward that penalizes redundant reasoning on simple problems while encouraging
781 extensive CoT for difficult ones. This preference data is optimized via SimPO, enabling efficient
782 dynamic control over reasoning path length.
- 783 • **O1-Pruner** Luo et al. (2025b): Based on reinforcement learning, this method rewards shorter
784 CoT traces without compromising accuracy. It employs an offline PPO-like procedure to prune
785 redundant reasoning while preserving or even improving correctness.
- 786 • **ShorterBetter** (Yi et al., 2025): This RL-based approach defines the optimal length for each
787 question as the shortest possible correct response and leverages this dynamic signal as a reward
788 for GRPO-based training, guiding the model toward concise yet accurate answers.
- 789 • **Dynasor-CoT** (Fu et al., 2025): Without extra training, this method dynamically truncates rea-
790 soning by probing intermediate answers, monitoring consistency, and detecting hesitancy tokens.
791 This yields substantial token savings while preserving accuracy.
- 792 • **DEER** (Xia et al., 2024): DEER employs a dynamic early-exit mechanism by monitoring reason-
793 ing transitions (such as “Wait”) to induce trial answers. Decisions to terminate CoT generation
794 are based on confidence estimation, reducing reasoning length without additional training.

795 All baseline models are tested under identical inference configurations and on the same benchmark
796 datasets to guarantee fair and reliable comparison. For each baseline, we use either the officially
797 released model or reproduce the method using released data and code.798 **B DATASET DETAILS**
799800 **Mathematical Reasoning Datasets**
801

- 802 • **GSM8K**: This dataset contains 8,500 English elementary school single-step math reasoning ques-
803 tions. It serves as one of the mainstream benchmarks for evaluating the math reasoning abilities
804 of large language models, focusing on basic arithmetic reasoning skills.
- 805 • **MATH-500**: Includes 500 medium-difficulty mathematical problems covering algebra, geome-
806 try, number theory, and other areas, designed to test the model’s comprehensive mathematical
807 reasoning ability.
- 808 • **AIME 2024/2025**: Originating from the American Invitational Mathematics Examination 2024
809 and 2025, each set contains 30 high-difficulty math questions, mainly assessing complex mathe-
810 matical reasoning and problem-solving skills.

810
 811 • **AMC 2023:** 40 questions from the American Mathematics Competitions (AMC), covering middle
 812 to high school levels, examining fundamental and advanced mathematics knowledge and problem
 813 solving abilities.

814
 815 **Scientific Reasoning Dataset** To evaluate model reasoning performance in other domains, we
 816 use the high-quality GPQA-diamond subset from the GPQA dataset. GPQA-diamond is a refined
 817 version of GPQA, focusing on challenging, high-quality scientific domain questions and designed
 818 to provide a comprehensive assessment of scientific understanding and reasoning ability.

819
 820 **C PROMPT TEMPLATE**
 821

822 Prompt for Batch Inference

823
 824 Please answer the following math problems in order and summarize all answers at the end:
 825 Your response should be in the following format:
 826
 827 [Solution Process]
 828 Provide a detailed solution for each problem...
 829
 830 [Final Answer]
 831 1. \boxed{{Answer1}}
 832 2. \boxed{{Answer2}}
 833 ...
 834 n. \boxed{{Answern}}
 835 Below is the list of questions:
 836 {numbered_questions}

837
 838 Prompt for Evaluation

839
 840 {origin_question}\n\n
 841 Please reason step by step, and put your final answer
 842 within \boxed{}.

843
 844 **D THE USE OF LARGE LANGUAGE MODELS (LLMs)**
 845

846 In the preparation of this manuscript, we made limited use of publicly available large language
 847 models (specifically GPT-5) solely to improve the clarity, grammar, and readability of the English
 848 text. The LLM was **not** used for research ideation, hypothesis generation, experiment design, data
 849 analysis, or the production of any scientific content. All technical ideas, experiments, analyzes,
 850 and conclusions presented in this paper are completely authored by the listed human authors. The
 851 authors have carefully reviewed and verified the accuracy of all LLM-assisted edits and take full
 852 responsibility for the final content.

853
 854 **E QUALITATIVE EXAMPLES**
 855

856
 857 **In this section, we provide qualitative comparisons between the baseline *DeepSeek-R1-Distill-***
 858 ***Qwen-7B* (Vanilla) and our *DRQA-7B* model. We explicitly denote the total token consumption**
 859 **for each response.**

864 E.1 CASE STUDY 1: GSM8K
865866 **Question**
867868 **A church has 120 members. 40% are adults. The rest are children. How many more
869 children are there than adults?**

870

871 **DeepSeek-R1-Distill-Qwen-7B (Vanilla)** [Total Tokens: 289]
872873 **<think>** First, I need to determine the number of adults in the church. Since 40% of the 120
874 members are adults, I'll calculate 0.40×120 , which gives me 48 adults. Next, I'll find the number
875 of children by subtracting the number of adults from the total membership. So, $120 - 48$ equals
876 72 children. Finally, to find out how many more children there are than adults, I'll subtract the
877 number of adults from the number of children: $72 - 48$, which equals 24. **</think>**878 **Solution:** Let's determine how many more children there are than adults in the church. Total
879 Members: The church has a total of 120 members. Number of Adults: 40% of the members are
adults.

880
$$\text{Number of Adults} = 0.40 \times 120 = 48$$

881 **Number of Children:** The rest of the members are children.

882
$$\text{Number of Children} = 120 - 48 = 72$$

883 **Difference:**

884
$$\text{Difference} = 72 - 48 = 24$$

885 **Final Answer:**886 24887 **DRQA-7B (Ours)** [Total Tokens: 82]888 **<think>** **1. Adults:** $120 \times 0.4 = 48$. **2. Children:** $120 - 48 = 72$. **3. Difference:** $72 - 48 = 24$.
889 **</think>** The number of more children than adults is:890 24

891

892

893

894

895

896

897

898

899

900

901

902

903

904

905

906

907

908

909

910

911

912

913

914

915

916

917

918

919

920

E.2 CASE STUDY 2: MATH-500

921

922 Question

923

924 How many positive whole-number divisors does 196 have?

925

926 DeepSeek-R1-Distill-Qwen-7B (Vanilla)

927

[Total Tokens: 740]

928 <think> Alright, so I need to find how many positive whole-number divisors 196 has. Hmm,
 929 okay, I remember that to find the number of divisors, I need to factorize the number into its
 930 prime factors first. Let me see, 196 is a number I see a bit often, is it a square? Let me check. 14
 931 squared is 196 because 14 times 14 is 196. Yeah, so that might be useful. Wait, instead of just
 932 stopping there, maybe I should confirm. If 14 times 14 is 196, then 14 is a factor of 196. But since
 933 14 isn't a prime number, I guess I need to break it down further into prime factors. So, breaking
 934 down 14, it's 2 times 7, right? So, both 2 and 7 are primes. So, if 196 is 14 squared, then it's 14
 935 times 14, which is (2 times 7) times (2 times 7). So, putting that together, 2 multiplied by 2 is 2^2 ,
 936 and 7 multiplied by 7 is 7^2 . So, 196 is equal to $2^2 \times 7^2$. Okay, so now in prime factorization form,
 937 196 is written as $2^2 \times 7^2$. I remember that the number of positive divisors is found by taking the
 938 exponents of each prime factor, adding one to each, and then multiplying them together. So, the
 939 formula is, if a number N can be expressed as a product of primes where $N = p_1^a \times p_2^b \times \dots \times p_n^k$,
 940 then the number of positive divisors is $(a+1)(b+1)\dots(k+1)$ (omit) ... </think>
 941 To find the number of positive whole-number divisors of 196, we start by performing the prime
 942 factorization of 196. First, we note that 196 is a perfect square, specifically 14^2 . Breaking down
 943 14 into its prime factors, we get 2×7 . Therefore:

$$14^2 = (2 \times 7)^2 = 2^2 \times 7^2$$

944

945

946

Using the formula for the number of divisors, if a number N can be expressed as $p_1^a \times p_2^b \times \dots \times p_n^k$, the number of positive divisors is $(a+1)(b+1)\dots(k+1)$. For 196, the prime factorization is $2^2 \times 7^2$, so the exponents are 2 and 2. Applying the formula, we get:

$$(2+1)(2+1) = 3 \times 3 = 9$$

947

948

To verify, we list all the divisors of 196: 1, 2, 4, 7, 14, 28, 49, 98, and 196. Counting these, we
 949 find there are 9 divisors. Thus, the number of positive whole-number divisors of 196 is 9.

950

951

DRQA-7B (Ours)

[Total Tokens: 165]

952

953

<think> We are asked: "How many positive whole-number divisors does 196 have?" Number:
 954 196. Factorization: $196 = 14^2 = (2 \times 7)^2 = 2^2 \times 7^2$. Divisors count formula: $(a+1)(b+1)$.
 955 Calculation: $(2+1)(2+1) = 3 \times 3 = 9$. Answer: 9. </think>

956

Step 1: Prime Factorization

$$196 = 2^2 \times 7^2$$

957

958

Step 2: Count the Divisors The number of positive divisors is given by the formula:

959

960

$$(2+1) \times (2+1) = 3 \times 3 = 9$$

961

962

Final Answer

9

963

964

965

966

967

968

969

970

971