

Offloaded Reasoning: Efficient Inference for Large Language Models via Modular Reasoning and Refinement

Anonymous Authors¹

Abstract

Large language models (LLMs) demonstrate strong reasoning capabilities but are expensive to run at inference time, limiting their practical deployment. We propose *Offloaded Reasoning* (OR), a modular strategy where a lightweight model generates intermediate reasoning traces that are then used by a larger model to produce the final answer. We further introduce *Offloaded Reasoning with Refinement* (ORR), where the large model first edits or improves the reasoning trace before answering. Unlike token-level acceleration methods, OR and ORR operate at the reasoning level and require no retraining of the large model. Experiments on GSM8K and Math500 show that OR achieves up to $8\times$ faster inference than full large-model reasoning with minimal accuracy loss, while ORR recovers or exceeds full accuracy at substantially lower cost. Our results highlight the potential of modular, delegation-based reasoning for building more efficient and adaptable LLM systems.

1. Introduction

Large language models (LLMs) have demonstrated strong performance on a wide range of complex reasoning tasks, such as arithmetic problem solving, logical inference, and open-domain question answering (Achiam et al., 2023; Team et al., 2024; Touvron et al., 2023; Roziere et al., 2023; Yang et al., 2024). However, the computational demands of such models during inference—especially at deployment scale—pose significant bottlenecks in resource-constrained applications.

A central challenge is to reduce the *inference-time compute cost* of LLMs without degrading their reasoning capabilities.

¹Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

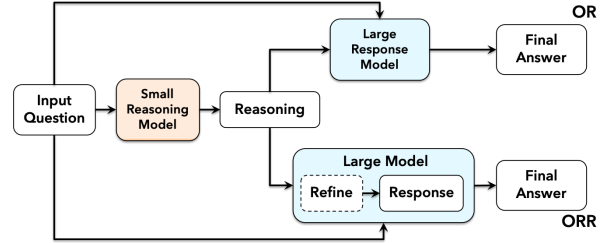


Figure 1: Workflow of Offloaded Reasoning (OR) and Offloaded Reasoning with Refinement (ORR). In OR, a lightweight model generates an initial reasoning trace, which is consumed by a larger model to produce the final answer. In ORR, the large model additionally refines the initial reasoning before generating the answer, improving accuracy while maintaining efficiency.

Recent approaches like speculative decoding (Leviathan et al., 2023) address this by drafting token sequences using smaller models and verifying them with larger ones. While effective for speeding up token generation, such methods operate purely at the token level and remain agnostic to the higher-level reasoning structure required for complex tasks. Moreover, speculative decoding imposes a significant practical constraint—it requires the smaller draft model and the larger verifier model to share the same tokenizer and vocabulary. This restricts model pair selection and limits flexibility in deploying heterogeneous model combinations for efficiency.

In this work, we propose **Offloaded Reasoning (OR)**, a simple yet powerful alternative that targets the reasoning process itself. Instead of having the large model perform all computation, OR offloads the generation of the intermediate reasoning trace (e.g., a chain-of-thought) to a smaller model. The large model then reads this offloaded reasoning and produces the final answer. This division allows the small model to shoulder the “cognitive burden” of reasoning while the large model focuses on synthesis, significantly reducing overall inference cost.

To further improve quality and robustness, we introduce **Offloaded Reasoning with Refinement (ORR)**. Here, the large model not only uses but also refines the offloaded

reasoning—correcting or extending it as needed—before producing the answer. This enables the large model to catch errors in the initial trace while still enjoying substantial compute savings.

Our approach is orthogonal to decoding-level acceleration and complements token-efficient generation. It builds on insights from chain-of-thought prompting, and modular reasoning, but focuses squarely on decoupling and optimizing the reasoning pipeline. We evaluate OR and ORR on two challenging mathematical reasoning benchmarks—**GSM8K** (Cobbe et al., 2021) and **Math500** (Lightman et al., 2023). Our main findings are:

- **OR substantially reduces inference cost:** For example, 7B-OR achieves a $8\times$ speedup over 7B full reasoning (FR) on Math500, with only a minor accuracy drop.
- **ORR recovers and often exceeds full reasoning performance:** Across model sizes, ORR consistently matches or outperforms FR in accuracy, while still reducing latency.
- **These benefits scale across model sizes:** The quality-efficiency tradeoff holds for 7B, 14B, and 32B models, demonstrating robustness across deployment regimes.

Our results show that reasoning can be effectively delegated to smaller models, enabling significant inference savings without compromising quality. We argue that modular reasoning strategies like OR and ORR offer a promising direction for efficient and scalable NLP systems, and point to a future where reasoning becomes a reusable, composable primitive in LLM pipelines.

2. Methodology

Let \mathcal{Q} denote an input question and \mathcal{A} the final answer. In traditional LLM inference, a large model \mathcal{M}_L directly maps \mathcal{Q} to \mathcal{A} by internally generating a reasoning trace \mathcal{R}_L (e.g., a chain-of-thought),

$$\mathcal{A} = \mathcal{M}_L(\mathcal{Q}) = f_{\text{answer}}(\mathcal{R}_L),$$

where $\mathcal{R}_L = f_{\text{reason}}(\mathcal{Q}, \mathcal{M}_L)$. Our core idea in Offloaded Reasoning (OR) is to offload the reasoning component f_{reason} to a smaller model \mathcal{M}_S , significantly reducing inference cost. Specifically, we generate an offloaded reasoning trace $\hat{\mathcal{R}}$ using \mathcal{M}_S :

$$\hat{\mathcal{R}} = f_{\text{reason}}(\mathcal{Q}, \mathcal{M}_S),$$

and feed this to the large model as additional context to generate the final answer:

$$\mathcal{A}_{\text{OR}} = \mathcal{M}_L([\mathcal{Q} \parallel \hat{\mathcal{R}}]),$$

where $[\mathcal{Q} \parallel \hat{\mathcal{R}}]$ denotes concatenation of the input question and the reasoning trace.

This process enables the large model to focus solely on synthesis (answer generation) while delegating the expensive reasoning process to a smaller model. The approach assumes that the small model is capable of generating sufficiently high-quality intermediate reasoning to scaffold the large model’s answer.

2.1. Offloaded Reasoning with Refinement (ORR)

In ORR, we further enhance this setup by allowing the large model to revise or extend the offloaded reasoning before generating the final answer. That is, instead of directly consuming $\hat{\mathcal{R}}$, the large model first computes a refined reasoning trace $\mathcal{R}^* = \text{Refine}(\hat{\mathcal{R}}, \mathcal{Q}, \mathcal{M}_L)$, and then uses \mathcal{R}^* to produce the answer

$$\mathcal{A}_{\text{ORR}} = \mathcal{M}_L([\mathcal{Q} \parallel \mathcal{R}^*]).$$

This formulation allows \mathcal{M}_L to retain control over correctness while still benefiting from the structure provided by \mathcal{M}_S . The refinement can involve correcting factual errors, expanding terse explanations, or realigning the reasoning with the final objective. Importantly, this operation incurs much lower compute than end-to-end large model reasoning, as the input to \mathcal{M}_L is scaffolded with a good starting point.

2.2. Compute Efficiency

Assuming the average number of tokens for reasoning is T_r and for answer generation is T_a , the total cost (in latency units) of standard inference is proportional to

$$\text{Cost}_{\text{baseline}} \propto T_r + T_a,$$

evaluated under \mathcal{M}_L . In contrast, OR and ORR incur

$$\text{Cost}_{\text{OR}} \propto T_r^{(S)} + T_a^{(L)},$$

$$\text{Cost}_{\text{ORR}} \propto T_r^{(S)} + T_r^{(L, \text{refine})} + T_a^{(L)},$$

where superscripts (S) and (L) denote small and large model compute respectively. Typically, $T_r^{(S)} \ll T_r$, and $T_r^{(L, \text{refine})} < T_r$, making both OR and ORR substantially more efficient.

3. Experiment Settings

We evaluate Offloaded Reasoning (OR) and Offloaded Reasoning with Refinement (ORR) on two established mathematical reasoning benchmarks: GSM8K and Math500.

3.1. Datasets

- **GSM8K** (Cobbe et al., 2021)¹: A dataset of 8.5K grade-school math word problems requiring step-by-step arithmetic reasoning. Each problem typically involves 2–8 steps and tests general logical reasoning skills. We evaluate on the official test split (1K samples).
- **Math500** (Lightman et al., 2023)²: A diverse benchmark of 500 mathematical questions from competition-style domains such as algebra, number theory, and geometry.

3.2. Experimental Setup

In our implementation, \mathcal{M}_S is a DEEPSEEK-R1-DISTILL-QWEN-1.5B (Guo et al., 2025) parameter model trained or finetuned to produce high-quality reasoning traces. The large model \mathcal{M}_L , DEEPSEEK-R1-DISTILL-QWEN-{7B, 14B, 32B}, consumes these traces either directly (OR) or via refinement (ORR). Importantly, no retraining of the large model is required; we only condition it on augmented inputs. These act as the answer modules and operate under one of the following configurations:

- No Reasoning (NR): Direct question-to-answer generation without reasoning.
- Full Reasoning (FR): \mathcal{M}_L Performs both reasoning and answer generation internally.
- OR: \mathcal{M}_L Consumes reasoning traces from \mathcal{M}_S to generate an answer.
- ORR: \mathcal{M}_L Refines the reasoning trace from \mathcal{M}_S before generating the final answer.

We offload reasoning to the 1.5B model to significantly reduce compute and latency. While its traces may be imperfect, they are often sufficient for the large model to refine or complete, making this setup far more efficient than using the large model for reasoning end-to-end.

Table 1 depicts the prompts used by Offloaded reasoning (OR) and Offloaded Reasoning with Refinement (ORR). In OR, response model is not allowed to reason but to produce the final response based on the provided reasoning from a smaller size reasoning model. On the other hand, in ORR response model is allowed to refine the offloaded reasoning just by not closing the `< /think >` token.

¹<https://huggingface.co/datasets/openai/gsm8k>

²<https://huggingface.co/datasets/HuggingFaceH4/MATH-500>

Table 1: Prompt templates used to query the large language model (\mathcal{M}_L). Here, *message* is the user query and *reason* is the output of the smaller reasoning model.

Method	Prompt Template
NR	<code>{message}< Assistant ><think>\nNo reasoning available</think></code>
FR	<code>{message}< Assistant ><think></code>
OR	<code>{message}< Assistant ><think>\n{reason}</think></code>
ORR	<code>{message}\n< Assistant ><think>\n nahh.. I got some speculations. let me check that\n<SPECULATIONS>\n{reason}\n</SPECULATIONS> ohh.. no.. But wait.. These speculations could be incorrect.</code>

3.3. Evaluation Metrics

Exact Match (EM) for GSM8K: Measures exact string match with the gold solution.

Pass@1 for Math500: Measures whether the top-1 generated answer is correct. Both the evaluation metrics are borrowed from LIGHTEVAL (Habib et al., 2023).

Average Inference Time (AIT): Measured in seconds per input query, averaged over all the samples per dataset including both reasoning and response generation steps.

4. Results and Analysis

Our central goal is to demonstrate that modularizing the reasoning process via a small model \mathcal{M}_S can deliver efficiency gains without sacrificing (and sometimes improving) performance.

4.1. Offloading Enables Efficient Reasoning.

Using a 1.5B reasoning model to assist a 7B base model (OR) improves accuracy over the standalone 1.5B model (GSM8K: 83.17 vs. 81.65 EM; Math500: 86.6 vs. 85.2 Pass@1), while drastically reducing inference latency relative to 7B Full Reasoning (FR) (Math500: 7.42s vs. 55.89s). This establishes that Offloaded Reasoning offers a superior quality-efficiency tradeoff compared to both small and large standalone reasoning models.

At the 14B scale, OR remains competitive in quality (GSM8K: 85.37 EM; Math500: 87.0 Pass@1) despite using a much smaller reasoning module, while achieving over 7× faster inference on Math500 compared to 14B-FR (11.47s

Table 2: Benchmarking on GSM8K and Math500 across inference setups. *Res* is the LLM generating the final response, *Rea* is the LLM generating the reasoning trace, *Conf* corresponds to different configurations {NR, FR, OR, ORR}, *AIT* is average inference time (seconds) per query, *EM* is Exact Match. Methods compared are *SELF-R* is SELF-REFINE from (Madaan et al., 2023)

Model			GSM8K		Math500	
Res	Rea	Conf	AIT(s)	EM	AIT(s)	Pass@1
1.5B	1.5B	FR	1.30	81.65	3.90	85.2
	x	NR	2.48	77.86	41.34	50.0
7B	7B	FR	3.86	85.22	55.89	89.4
	1.5B	OR	2.40	83.17	7.42	86.6
	1.5B	ORR	5.15	88.70	31.60	89.4
	x	NR	3.67	64.59	61.20	57.6
14B	14B	FR	8.05	90.30	77.56	89.0
	1.5B	OR	4.48	85.37	11.47	87.0
	1.5B	ORR	7.59	92.34	54.70	90.6
	x	NR	3.78	40.64	65.90	53.6
32B	32B	FR	17.55	91.36	158.32	92.8
	1.5B	OR	7.24	84.00	23.16	87.4
	1.5B	ORR	16.28	93.40	128.18	93.2
Comparison with existing approaches (relative gains)						
32B	1.5B-ORR		-1.27	+2.04	-30.14	+0.4
GPT-4	SELF-R		x	+0.2	x	x

vs. 77.56s). Even at 32B, the OR setup produces accurate results (GSM8K: 84.00 EM; Math500: 87.4 Pass@1) while reducing inference time by over $7\times$ on Math500 compared to 32B-FR (23.16s vs. 158.32s).

These trends underscore that Offloaded Reasoning not only scales well to larger backbone models, but also provides substantial latency savings while maintaining strong performance—making it an attractive inference strategy across compute budgets.

4.2. Refinement Boosts Quality

The ORR setup, where the large model refines the small model’s reasoning traces before answering, consistently outperforms all other configurations in accuracy. For instance, on GSM8K, 7B-ORR achieves 88.70 EM, outperforming both 7B-FR (85.22) and 7B-OR (83.17). Similarly, 32B-ORR achieves the highest overall scores on both benchmarks (GSM8K: 93.40 EM; Math500: 93.2 Pass@1). This indicates that large models can effectively refine and enhance imperfect reasoning traces generated by smaller models. Few examples are shown in Appendix A.

4.3. Substantial Inference Gains

Offloading reasoning consistently yields large reductions in average inference time. For example, 7B-OR reduces latency by over $7\times$ compared to 7B-FR on Math500 (7.42s vs. 55.89s) with less than 3% drop in accuracy. Even at the 14B and 32B scale, OR setups demonstrate notable latency improvements over FR while maintaining competitive performance.

Although Offloaded Reasoning with Refinement (ORR) incurs higher latency than OR (e.g., 7B-ORR: 31.60s vs. 7.42s on Math500), it consistently matches or exceeds FR in accuracy (e.g., 7B-ORR and 7B-FR both yield 89.4 Pass@1 on Math500), offering a favorable tradeoff for settings where accuracy is paramount. At 32B, ORR reaches the highest accuracy overall (Math500: 93.2 Pass@1), while still being $1.3\times$ faster than 32B-FR (128.18s vs. 158.32s). These results suggest that **ORR offers a high-quality inference mode with moderate efficiency gains, while OR maximizes speed with minimal quality tradeoff.**

4.4. Modularity Across Model Families

To evaluate the modularity of offloaded reasoning, we tested whether reasoning traces generated by the 1.5B model could transfer to a different model family. Specifically, we used the 1.5B traces with a DEEPSEEK-R1-DISTILL-LLAMA-8B response model. Despite architectural differences, both OR and ORR outperformed the 8B model’s full reasoning (FR) on GSM8K dataset, with ORR achieving an Exact Match (EM) of 84.84 compared to 47.01 for FR. This demonstrates that offloaded reasoning traces are not only efficient but also transferable across model families, supporting the viability of modular, interoperable LLM pipelines.

Discussion and Future Work

We presented a modular reasoning framework that enables efficient inference by combining a small reasoning model with a larger model via trace-guided prompting. Evaluations on GSM8K and Math500 demonstrate strong efficiency gains in structured reasoning settings. While we focus on arithmetic and symbolic domains, the approach is model-agnostic and can extend to open-domain QA or commonsense reasoning, which we leave to future work.

Despite occasional imperfections in generated traces, our results show that even approximate reasoning can provide valuable guidance. Future work may explore methods to improve trace quality, such as selective verification or confidence-based filtering.

Our prompt-based integration keeps the system simple and general. Extensions like dynamic routing based on question difficulty or model uncertainty could further improve

Aspect	Prior Work	This Work (OR / ORR)
Granularity of Control	Operates at the level of reasoning length or token-level interventions, e.g., Chain-of-Thought prompting (Wei et al., 2022), instruction-driven expansion (Jin et al., 2024), multi-agent debate (Liang et al., 2023; Pham et al., 2024), and use of drafter models (Leviathan et al., 2023)	Operates at module-level granularity: a small model performs reasoning, while the large model synthesizes or refines the response.
Adaptivity to Query Complexity	Explicitly controlled using reinforcement learning or heuristic scheduling of reasoning steps (Aggarwal et al., 2023; Xu et al., 2024; Wang et al., 2024), or self-reflective methods (Zelikman et al., 2022; Madaan et al., 2023).	Adaptivity is implicit—reasoning traces are generated by the small model without pre-specified length. ORR enables recovery or improvement without explicit step control.
Efficiency Mechanism	Reduces inference cost through response truncation, token budget optimization, or rollout control (Snell et al., 2024).	Reduces compute by offloading reasoning to a 1.5B model, yielding up to $8\times$ speedup over full large-model reasoning.
Learning Requirements	Requires RL-based or constrained training objectives (Altman, 2021), with risk of reward hacking (Pan et al., 2022; Skalse et al., 2022).	Requires no retraining or fine-tuning. The pipeline is built entirely from prompt augmentation and standard model calls.

Table 3: Comparison of Offloaded Reasoning (OR/ORR) with prior LLM-based reasoning optimization strategies.

compute-efficiency trade-offs.

Finally, while we report latency and accuracy, the framework naturally supports broader system-level metrics (e.g., memory, energy). Future evaluations on these axes will be important for deployment in resource-constrained environments.

Related Work

Prior work on reasoning with language models has focused on controlling inference at fine-grained levels such as token or step length. Chain-of-Thought prompting (Wei et al., 2022), instruction-driven expansions (Jin et al., 2024), and multi-agent collaborations (Liang et al., 2023; Pham et al., 2024) use handcrafted or learned mechanisms to guide reasoning length or structure. Other methods leverage drafter-solver patterns (Leviathan et al., 2023), often requiring model-specific coordination. In contrast, our approach introduces modularity at the system level: a small model generates full reasoning traces, while a large model refines or completes the answer. This decouples the reasoning process from token-level control and simplifies integration.

Adaptivity to query complexity has been addressed via reinforcement learning or heuristic-based control of reasoning steps (Aggarwal et al., 2023; Xu et al., 2024; Wang et al., 2024), as well as through self-reflective techniques that iteratively revise outputs (Zelikman et al., 2022; Madaan et al., 2023). Our method offers implicit adaptivity: the small model autonomously determines the trace length, and ORR enables selective refinement without explicit step tuning or dynamic halting.

Efficiency-oriented approaches typically reduce inference cost through response truncation, budget-aware decoding, or

rollout control (Snell et al., 2024). OR and ORR reduce cost by offloading reasoning to a 1.5B model, achieving up to $8\times$ speedup over full large-model inference while maintaining competitive accuracy.

Finally, many prior methods require task-specific training, reinforcement learning, or constrained optimization objectives (Altman, 2021), which can suffer from instability and reward misalignment (Pan et al., 2022; Skalse et al., 2022). Our framework avoids these complexities: it relies entirely on prompt augmentation and zero-shot calls to pretrained models, requiring no additional training or data. Table 3 contrasts our proposed approach with related reasoning optimization strategies in LLMs.

5. Conclusion

Offloaded Reasoning with Refinement uniquely combines modular reasoning, efficiency, and robustness in a unified framework. While prior works have addressed components of this pipeline, our contribution lies in orchestrating small and large models for reasoning and refinement, resulting in significant inference gains without compromising accuracy. This modularity also enables dynamic scaling, making it highly suitable for real-world deployments with variable compute budgets.

References

- Achiam, J., Adler, S., Agarwal, S., Ahmad, L., Akkaya, I., Aleman, F. L., Almeida, D., Altenschmidt, J., Altman, S., Anadkat, S., et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.
- Aggarwal, P., Madaan, A., Yang, Y., et al. Let’s sample step by step: Adaptive-consistency for efficient reasoning and

- coding with llms. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 12375–12396, 2023.
- Altman, E. *Constrained Markov decision processes*. Routledge, 2021.
- Cobbe, K., Kosaraju, V., Bavarian, M., Chen, M., Jun, H., Kaiser, L., Plappert, M., Tworek, J., Hilton, J., Nakano, R., et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.
- Guo, D., Yang, D., Zhang, H., Song, J., Zhang, R., Xu, R., Zhu, Q., Ma, S., Wang, P., Bi, X., et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.
- Habib, N., Fourrier, C., Kydlíček, H., Wolf, T., and Tunstall, L. Lighteval: A lightweight framework for llm evaluation, 2023. URL <https://github.com/huggingface/lighteval>.
- Jin, M., Yu, Q., Shu, D., Zhao, H., Hua, W., Meng, Y., Zhang, Y., and Du, M. The impact of reasoning step length on large language models. *arXiv preprint arXiv:2401.04925*, 2024.
- Leviathan, Y., Kalman, M., and Matias, Y. Fast inference from transformers via speculative decoding. In *International Conference on Machine Learning*, pp. 19274–19286. PMLR, 2023.
- Liang, T., He, Z., Jiao, W., Wang, X., Wang, Y., Wang, R., Yang, Y., Shi, S., and Tu, Z. Encouraging divergent thinking in large language models through multi-agent debate. *arXiv preprint arXiv:2305.19118*, 2023.
- Lightman, H., Kosaraju, V., Burda, Y., Edwards, H., Baker, B., Lee, T., Leike, J., Schulman, J., Sutskever, I., and Cobbe, K. Let’s verify step by step. *arXiv preprint arXiv:2305.20050*, 2023.
- Madaan, A., Tandon, N., Gupta, P., Hallinan, S., Gao, L., Wiegrefe, S., Alon, U., Dziri, N., Prabhunoye, S., Yang, Y., et al. Self-refine: Iterative refinement with self-feedback. *Advances in Neural Information Processing Systems*, 36:46534–46594, 2023.
- Pan, A., Bhatia, K., and Steinhardt, J. The effects of reward misspecification: Mapping and mitigating misaligned models. In *International Conference on Learning Representations*, 2022.
- Pham, C., Liu, B., Yang, Y., Chen, Z., Liu, T., Yuan, J., Plummer, B. A., Wang, Z., and Yang, H. Let models speak ciphers: Multiagent debate through embeddings. In *ICLR*, 2024.
- Roziere, B., Gehring, J., Gloeckle, F., Sootla, S., Gat, I., Tan, X. E., Adi, Y., Liu, J., Sauvestre, R., Remez, T., et al. Code llama: Open foundation models for code. *arXiv preprint arXiv:2308.12950*, 2023.
- Skalse, J., Howe, N., Krashennikov, D., and Krueger, D. Defining and characterizing reward gaming. *Advances in Neural Information Processing Systems*, 35:9460–9471, 2022.
- Snell, C., Lee, J., Xu, K., and Kumar, A. Scaling llm test-time compute optimally can be more effective than scaling model parameters. *arXiv preprint arXiv:2408.03314*, 2024.
- Team, G., Mesnard, T., Hardin, C., Dadashi, R., Bhupatiraju, S., Pathak, S., Sifre, L., Rivière, M., Kale, M. S., Love, J., et al. Gemma: Open models based on gemini research and technology. *arXiv preprint arXiv:2403.08295*, 2024.
- Touvron, H., Martin, L., Stone, K., Albert, P., Almahairi, A., Babaei, Y., Bashlykov, N., Batra, S., Bhargava, P., Bhosale, S., et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.
- Wang, X., Feng, S., Li, Y., Yuan, P., Zhang, Y., Tan, C., Pan, B., Hu, Y., and Li, K. Make every penny count: Difficulty-adaptive self-consistency for cost-efficient reasoning. *arXiv preprint arXiv:2408.13457*, 2024.
- Wei, J., Wang, X., Schuurmans, D., Bosma, M., Xia, F., Chi, E., Le, Q. V., Zhou, D., et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.
- Xu, M., Li, Y., Sun, K., and Qian, T. Adaption-of-thought: Learning question difficulty improves large language models for reasoning. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 5468–5495, 2024.
- Yang, A., Yang, B., Hui, B., Zheng, B., Yu, B., Zhou, C., Li, C., Li, C., Liu, D., Huang, F., et al. Qwen2 technical report. *arXiv preprint arXiv:2407.10671*, 2024.
- Zelikman, E., Wu, Y., Mu, J., and Goodman, N. Star: Bootstrapping reasoning with reasoning. *Advances in Neural Information Processing Systems*, 35:15476–15488, 2022.

A. Sample Examples

To demonstrate the effectiveness of the OR and ORR, we picked few examples from GSM8k datasets in table 4 and 5 where the 1.5B LLM takes the wrong turn (in RED) while generating reasoning. In Table 4, 32B response generator

able to correctly identify the problem in the reasoning step (in BLUE) both with and without refinement. In Table 5 32B response generator with 1.5B OR does not perform any reasoning and fails to identify the main cause of the wrong reasoning. However, 1.5B ORR able to refine the 1.5B generated reasoning and correctly identifies the main cause of 1.5B reasoning incorrectness (in BLUE).

Table 4: Here, 1.5B Reasoning is incorrect. where the 1.5 B reasoner incorrectly got 450 as actual temperature ...find 4% of 450..., it should be ...find 4% of 520....This is precisely fixed by the 1.5 OR and 1.5B ORR Variants where the 32B able to correct the offloaded reasoning and able to identify the current temperature asked in the query that is *The correct temperature is 520. So...*

Key	Value
Query	Maggie's oven is malfunctioning. When she sets it to 450 the actual temperature is 468. If it's off by the same percentage for any recipe, what temperature should she set it at if her recipe calls for 520 degrees?
Gold response	The oven off by 18 degrees because $468 - 450 = 18$. The oven is off by 4% because $18 / 450 \times 100 = 4\%$. Therefore $X + .04X = 520$. Therefore $1.04X = 520$. The temperature to set it is 500 because $520 / 1.04 = 500$.
1.5B Reasoning	Okay, so I need to figure out what temperature Maggie should set her oven to for her recipe that calls for 520 degrees. She's currently setting it to 450, but the oven is malfunctioning. When she sets it to 450, the actual temperature is 468. That means the oven is overbaking by 18 degrees. First, I should find out by what percentage the oven is overbaking. The formula for percentage overbaking is (Actual Temperature - Set Temperature) divided by Set Temperature, then multiplied by 100. So that's $(468 - 450) / 450 \times 100$. Let me calculate that: $18 / 450$ is 0.04, and 0.04 times 100 is 4%. So the oven is overbaking by 4%. Now, if she wants to adjust the temperature for any recipe, she needs to reduce the temperature by the same percentage. So she's currently at 450, and she wants to reduce it by 4%. To find 4% of 450, I'll do 450×0.04 , which equals 18. So she needs to reduce the temperature by 18 degrees. Subtracting that from the current temperature: $450 - 18 = 432$. So the new temperature should be 432 degrees. Wait, let me double-check that. If she sets it to 432, what would the oven actually be? Using the same percentage calculation: $(432 - 450) / 450 \times 100$. That's $(-18) / 450 \times 100$, which is -4%. So the oven would be underbaking by 4%, which is the same percentage as before. That makes sense because she's adjusting the temperature to compensate for the overbaking. So, yes, 432 degrees should be the correct temperature for her recipe.
1.5B Response	Maggie should set her oven to 432 degrees. This adjustment compensates for the 4% overbaking, ensuring the recipe is correctly baked.
32B Response + 1.5B (OR)	To determine the correct temperature Maggie should set her oven to, we first calculate the percentage by which the oven is overbaking. The oven is set to 450 but reaches 468, which is an overbake of 18. The percentage overbake is: 4%. To compensate for this, Maggie needs to reduce the temperature by 4%. For a recipe requiring 520, the adjusted temperature is: $520 \times (1 - 0.04) = 520 \times 0.96 = 500$. Answer: 500 degrees.
32B Response + 1.5B ORR	Let me think again. Wait, actually, the oven is overbaking by 4% when set to 450. So, to get the correct temperature, she needs to set it lower. The correct temperature is 520. So, she needs to find what temperature to set so that when the oven overbakes by 4%, it reaches 520. So, let me rephrase that. Let x be the temperature she should set. Then, $x \times 1.04 = 520$. So, $x = 520 / 1.04$. Let me calculate that: 520 divided by 1.04. 1.04 times 500 is 520, so x is 500. So, she should set it to 500 degrees. Wait, that makes more sense. Because if she sets it to 500, the oven will overbake by 4%, which is $500 \times 0.04 = 20$, so $500 + 20 = 520$, which is the desired temperature. So, the correct answer is 500 degrees. I think

Table 5: Here, 1.5B Reasoning is incorrect. where the 1.5 B reasoner incorrectly got 450 as actual temperature ...find 4% of 450..., it should be ...find 4% of 520....This is precisely fixed by the 1.5 OR and 1.5B ORR Variants where the 32B able to correct the offloaded reasoning and able to identify the current temperature asked in the query that is *The correct temperature is 520. So...*

Key	Value
Query	If Martin eats Cheerios every day for breakfast, he'll lose 1.25 pounds/week. If he eats donuts every day for breakfast, he'll gain 1.75 pounds/week. What will be the difference in his weight at the end of 5 weeks between the two breakfast options?
Gold response	First find the difference in weight gain per week: 1.75 pounds/week - (-1.25 pounds/week) = 1.75 pounds/week + 1.25 pounds/week = 1.75-(-1.25)=3, 3 pounds/week Then multiply the weight gain per week by the number of weeks to find the total difference at the end of 5 weeks: 5 weeks * 3 pounds/week = 5*3=15, 15 pounds 15
1.5B Reasoning	Okay, so I need to figure out the difference in weight between Martin's two breakfast options over five weeks. Let me break this down step by step. First, Martin eats Cheerios every day for breakfast. Over one week, he loses 1.25 pounds. So, over five weeks, he would lose 1.25 pounds multiplied by 5. Let me calculate that: $1.25 * 5 = 6.25$ pounds lost. On the other hand, he eats donuts every day. This time, he gains 1.75 pounds each week. So, over five weeks, he would gain $1.75 * 5$ pounds. Calculating that gives me 8.75 pounds gained. Now, to find the difference in weight between the two options at the end of five weeks, I subtract the weight lost from the weight gained. That would be $8.75 - 6.25 = 2.5$ pounds. So, Martin would weigh 2.5 pounds more if he eats donuts every day compared to eating Cheerios every day over five weeks.
1.5B Response	Martin would weigh 2.5 pounds more if he eats donuts every day compared to eating Cheerios every day over five weeks.
32B Response + 1.5B (OR)	The difference in weight after 5 weeks is 2.5 pounds.
32B Response + 1.5B ORR	Let me think again. Wait, actually, the difference in weight would be the total weight change from each option. Since one is a loss and the other is a gain, the total difference would be the sum of the absolute values of both changes. So, if he loses 6.25 pounds on Cheerios and gains 8.75 pounds on donuts, the difference between the two would be $6.25 + 8.75 = 15$ pounds. Wait, that makes more sense because the total difference in weight is the sum of the two changes since one is a loss and the other is a gain. So, the correct difference is 15 pounds. ;/think _z