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

Ishan Jindal[a], Jayant Taneja[a], Chandana Badrinath[b]*, Vikas Kapur[a], Sachin Dev Sharma[a]
[a]Samsung R&D Institute India - Delhi
[b]Mohamed bin Zayed University of AI, Abu Dhabi, UAE
{ishan.jindal, j.taneja, vikas.kapur, sachin.dev}@samsung.com,
badrinath.chandana@mbzuai.ac.ae

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× 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. 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.

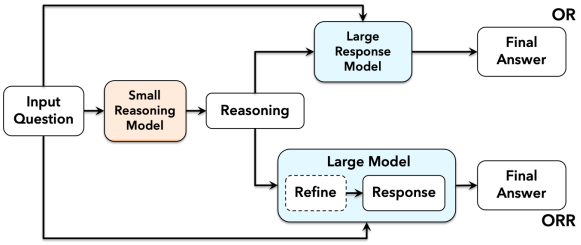

Figure 1: Workflow of Offloaded Reasoning (OR) and Offloaded Reasoning with Refinement (ORR). In OR, a lightweight model generates an initial reasoning trace, which is consumed by a larger model to produce the final answer. In ORR, the large model additionally refines the initial reasoning before generating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

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

*work conducted while at Samsung R&D Institute India - Delhi

Findings of the Association for Computational Linguistics: EMNLP 2025, pages 7450–7458
November 4-9, 2025

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 a 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. Specific prompts used for OR and ORR are depicted in Appendix A.

[1] https://huggingface.co/datasets/openai/gsm8k

[2] https://huggingface.co/datasets/HuggingFaceH4/MATH-500

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
7B	x	NR	2.48	77.86	41.34	50.0
	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
14B	x	NR	3.67	64.59	61.20	57.6
	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
32B	x	NR	3.78	40.64	65.90	53.6
	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

Table 1: 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)

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 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× 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 B .

4.3 Substantial Inference Gains

Offloading reasoning consistently yields large reductions in average inference time. For example, 7B-OR reduces latency by over 7× 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× 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 test the modularity and transferability of offloaded reasoning, we applied the 1.5B reasoning traces to a different response model family—DEEPSEEK-R1-DISTILL-LLAMA-8B. As shown in Table 2, the OR and ORR configurations significantly outperform full reasoning (FR) by the same 8B model. In particular, ORR improves Exact Match (EM) from 47.01 to 84.84, demonstrating that reasoning traces generated by the 1.5B model can generalize across model architectures and families. This highlights the plug-and-play potential of offloaded reasoning for diverse LLM deployments.

Res	Rea	Conf	AIT(s)	EM
8B	8B	FR	4.83	47.01
8B	1.5B	OR	1.26	76.95
8B	1.5B	ORR	4.36	84.84

Table 2: Evaluating modularity of 1.5B-generated reasoning traces to a LLAMA-8B response model.

5 Related Work

The goal of reducing inference cost while maintaining or improving quality has been explored through multiple research avenues, including efficient decoding strategies, modular computation, and few-shot reasoning techniques. Our approach draws from and contributes to each of these directions. Table 3 contrasts our proposed approach with related reasoning optimization strategies in LLMs.

Efficient Inference in Large Language Models Recent work has addressed the challenge of high inference cost in large models by proposing techniques such as early exit (Schuster et al., 2022), speculative decoding (Leviathan et al., 2023), and distillation (Sreenivas et al., 2024; Liu et al., 2024). Speculative decoding in particular leverages smaller draft models to speed up generation by deferring verification to a larger model, though it remains limited to token-level prediction rather than higher-level reasoning. Unlike token-level acceleration methods, our approach operates at the level of full reasoning traces, offering orthogonal benefits in structured reasoning tasks.

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× 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.

Chain-of-Thought and Few-Shot Reasoning Chain-of-thought prompting (Wei et al., 2022) and its self-consistency variants (Wang et al., 2023) have demonstrated large gains in reasoning tasks. Recent approaches have further improved quality by generating and selecting diverse reasoning paths (Creswell et al., 2022). Our method builds on this paradigm by explicitly offloading the reasoning to a smaller model, showing that reasoning quality and cost efficiency need not be tightly coupled to model scale.

Collaborative and Multi-Agent LMs Multi-agent systems (Park et al., 2023; Shinn et al., 2023) explore the use of multiple language models in collaborative roles. These works often involve dialogue or planning tasks. Our formulation is a specific instance of this philosophy: one model is tasked with reasoning and another with validating or refining it. Unlike prior works that use agents with similar capacities, we show how models of vastly different scales can collaborate to achieve higher efficiency and reliability.

6 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.

Limitations

This work focuses on arithmetic and symbolic reasoning tasks (GSM8K and Math500), which offer a controlled and well-understood setting for evaluating modular reasoning. While we do not evaluate on other domains, such as commonsense or open-domain QA, our approach is model-agnostic and readily extensible to these areas in future work.

The small reasoning model may occasionally generate imperfect traces, but our results show that even approximate reasoning often provides a strong scaffold for the large model to refine or complete. Investigating methods to further improve the robustness of reasoning quality—e.g., through selective verification or trace filtering—is a promising direction.

We rely on lightweight prompt-based integration, which keeps the system simple and broadly applicable. While prompt optimization and dynamic routing mechanisms (e.g., based on question difficulty or model confidence) are not explored here, they are orthogonal to our core contributions and could further enhance the efficiency-accuracy trade-off.

Finally, while we primarily report latency and accuracy, our framework naturally supports fine-grained analysis of compute, memory, or energy cost—all of which can be incorporated in future deployment-focused evaluations. Overall, this work introduces a novel modular reasoning paradigm that enables significant inference savings with minimal system changes.

References

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

Pranjal Aggarwal, Aman Madaan, Yiming Yang, et al. 2023. 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*, pages 12375–12396.

Eitan Altman. 2021. *Constrained Markov decision processes*. Routledge.

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

Antonia Creswell, Murray Shanahan, and Irina Higgins. 2022. Selection-inference: Exploiting large language models for interpretable logical reasoning. *arXiv preprint arXiv:2205.09712*.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

Nathan Habib, Clémentine Fourrier, Hynek Kydlíček, Thomas Wolf, and Lewis Tunstall. 2023. Lighteval: A lightweight framework for llm evaluation.

Mingyu Jin, Qinkai Yu, Dong Shu, Haiyan Zhao, Wenyue Hua, Yanda Meng, Yongfeng Zhang, and Mengnan Du. 2024. The impact of reasoning step length on large language models. *arXiv preprint arXiv:2401.04925*.

Yaniv Leviathan, Matan Kalman, and Yossi Matias. 2023. Fast inference from transformers via speculative decoding. In *International Conference on Machine Learning*, pages 19274–19286. PMLR.

Tian Liang, Zhiwei He, Wenxiang Jiao, Xing Wang, Yan Wang, Rui Wang, Yujiu Yang, Shuming Shi, and Zhaopeng Tu. 2023. Encouraging divergent thinking in large language models through multi-agent debate. *arXiv preprint arXiv:2305.19118*.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2023. Let's verify step by step. *arXiv preprint arXiv:2305.20050*.

Jiaheng Liu, Chenchen Zhang, Jinyang Guo, Yuanxing Zhang, Haoran Que, Ken Deng, Jie Liu, Ge Zhang, Yanan Wu, Congnan Liu, et al. 2024. Ddk: Distilling domain knowledge for efficient large language models. *Advances in Neural Information Processing Systems*, 37:98297–98319.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, et al. 2023. Self-refine: Iterative refinement with self-feedback. *Advances in Neural Information Processing Systems*, 36:46534–46594.

Alexander Pan, Kush Bhatia, and Jacob Steinhardt. 2022. The effects of reward misspecification: Mapping and mitigating misaligned models. In *International Conference on Learning Representations*.

Joon Sung Park, Joseph O'Brien, Carrie Jun Cai, Meredith Ringel Morris, Percy Liang, and Michael S Bernstein. 2023. Generative agents: Interactive simulacra of human behavior. In *Proceedings of the 36th annual acm symposium on user interface software and technology*, pages 1–22.

Chau Pham, Boyi Liu, Yingxiang Yang, Zhengyu Chen, Tianyi Liu, Jianbo Yuan, Bryan A Plummer, Zhaoran Wang, and Hongxia Yang. 2024. Let models speak ciphers: Multiagent debate through embeddings. In *ICLR*.

Baptiste Roziere, Jonas Gehring, Fabian Gloeckle, Sten Sootla, Itai Gat, Xiaoqing Ellen Tan, Yossi Adi, Jingyu Liu, Romain Sauvestre, Tal Remez, et al. 2023. Code llama: Open foundation models for code. *arXiv preprint arXiv:2308.12950*.

Tal Schuster, Adam Fisch, Jai Gupta, Mostafa Dehghani, Dara Bahri, Vinh Tran, Yi Tay, and Donald Metzler. 2022. Confident adaptive language modeling. *Advances in Neural Information Processing Systems*, 35:17456–17472.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2023. Reflexion: Language agents with verbal reinforcement learning. *Advances in Neural Information Processing Systems*, 36:8634–8652.

Joar Skalse, Nikolaus Howe, Dmitrii Krasheninnikov, and David Krueger. 2022. Defining and characterizing reward gaming. *Advances in Neural Information Processing Systems*, 35:9460–9471.

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

Sharath Turuvekere Sreenivas, Saurav Muralidharan, Raviraj Joshi, Marcin Chochowski, Ameya Sunil Mahabaleshwarkar, Gerald Shen, Jiaqi Zeng, Zijia Chen, Yoshi Suhara, Shizhe Diao, et al. 2024. Llm pruning and distillation in practice: The minitron approach. *arXiv preprint arXiv:2408.11796*.

Gemma Team, Thomas Mesnard, Cassidy Hardin, Robert Dadashi, Surya Bhupatiraju, Shreya Pathak, Laurent Sifre, Morgane Rivière, Mihir Sanjay Kale, Juliette Love, et al. 2024. Gemma: Open models based on gemini research and technology. *arXiv preprint arXiv:2403.08295*.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Xinglin Wang, Shaoxiong Feng, Yiwei Li, Peiwen Yuan, Yueqi Zhang, Chuyi Tan, Boyuan Pan, Yao Hu, and Kan Li. 2024. Make every penny count: Difficulty-adaptive self-consistency for cost-efficient reasoning. *arXiv preprint arXiv:2408.13457*.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V Le, Ed H Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023. Self-consistency improves chain of thought reasoning in language models. In *The Eleventh International Conference on Learning Representations*.

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

Mayi Xu, Yongqi Li, Ke Sun, and Tieyun Qian. 2024. 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*, pages 5468–5495.

An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, et al. 2024. Qwen2 technical report. *arXiv preprint arXiv:2407.10671*.

Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah Goodman. 2022. Star: Bootstrapping reasoning with reasoning. *Advances in Neural Information Processing Systems*, 35:15476–15488.

A Prompt

Table 4 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.

B Sample Examples

To demonstarte the effectiveness of the OR and ORR, we picked few examples from GSM8k datasets in table 5 and 6 where the 1.5B LLM takes the wrong turn (in RED) while generating reasoning. In Table 5, 32B response generator able to correctly identify the problem in the reasoning step (in BLUE) both with and without refinement. In Table 6 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).

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

Table 4: Here, *message* is the actual query and *reason* is the offloaded reasoning from a smaller language model. These prompts directly feed to the response LLM ($\mathcal{M}_{\mathcal{L}}$) for next token generation.

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 = «468-450=18»18The oven is off by 4% because 18 / 450 x 100 = «18/450*100=4»4%Therefore X + .04X = 520Therefore 1.04X = 520The temperature to set it is 500 because 520 / 1.04 = «520/1.04=500»500 500
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 * 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 * 100. That's (-18)/450 * 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 * (1 - 0.04) = 520 * 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 * 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 * 0.04 = 20, so 500 + 20 = 520, which is the desired temperature.So, the correct answer is 500 degrees.</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/weekThen 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>

Table 6: Here, 1.5B Reasoning is incorrect. where the 1.5 B reasoner end up subtracting the correctly computed weights gains instead of adding them *...ould be 8.75 - 6.25 = 2.5 ...*, it should be *...ould be 8.75 + 6.25 = 15 ...*.This is precisely fixed by the 1.5 ORR Variants where the 32B able to refine the offloaded reasoning and able to identify that the total difference in this question is in fact the sum of absolute values *the total difference would be the sum of the absolute values of both changes....*