University of Bonn

CAISA lab

Natural Language Processing Lab

(Summer Semester 2025)

Final Report of Team 9

Solution / Verification Tradeoffs in Reasoning Models

Group Members:

Umutcan Dogan 50324204

Instructor: Prof. Dr. Lucie Flek
Advisor: Dr. Florian Mai

September 21, 2025

1. Introduction

Large Language Models (LLMs) have shown remarkable capabilities in complex reasoning tasks, particularly through recent advances in training models with hidden scratchpads (reasoning tokens) via reinforcement learning [1, 2]. Unlike traditional Chain-of-Thought (CoT) prompting which guides models to generate visible intermediate steps [3], modern reasoning models employ hidden thinking processes that are not exposed to the user but significantly enhance problem-solving capabilities.

The ability to verify another model's output represents a crucial advancement in LLM deployment with multiple important applications. Self-verification and self-correction mechanisms enable models to iteratively improve their performance by checking and refining their own outputs [4, 5]. Beyond performance enhancement, verification plays a critical role in AI Control and safety, as outlined by [6]. In safety-critical applications, employing a second model as a verifier to assess whether a generator model's output is safe and accurate becomes essential for responsible AI deployment. This verification paradigm ensures that potentially harmful or incorrect outputs can be caught before reaching end users, establishing a crucial safety layer in AI systems.

While these thinking processes enhance accuracy, they come at a significant computational cost. The generation of reasoning tokens during inference consumes substantial resources, and this cost scales with the complexity and length of the reasoning chain. This presents a critical trade-off between performance and efficiency. For practical applications, it is desirable to allocate just enough computational resources for a model to reliably verify a solution without wasteful over-computation.

Recent advances in reasoning models have demonstrated the power of reinforcement learning-trained systems with dynamic thinking capabilities. DeepSeek-R1 [7] exemplifies this approach, showing how models can be trained to develop sophisticated reasoning through RL techniques. The S1 architecture [8] introduces dynamic adjustment of thinking tokens, allowing models to adaptively allocate computational resources based on problem complexity. In this project, we utilize OpenAI's and Google's Gemini models, which, while their exact training methodologies remain proprietary, very likely employ similar reinforcement learning approaches as demonstrated in DeepSeek-R1 to achieve their advanced reasoning capabilities.

To the best of our knowledge, our specific research question has only been addressed by the concurrent work of [9], who examined scaling laws for verification and found that verification costs scale predictably with problem complexity. However, their work focused on aggregate scaling patterns rather than instance-level prediction of optimal budgets, which is the focus of our investigation.

Our central research question is: **Can a machine learning model accurately predict the minimum reasoning token budget required for a verifier LLM to correctly assess the solution to a mathematical word problem?** We hypothesize that textual features from the problem statement and a generated solution can be used to train a classifier that predicts the optimal reasoning budget, thereby enabling more efficient resource allocation for LLM inference.

2. Related Work

This project is situated at the intersection of several key areas in LLM research: reasoning models with hidden scratchpads, self-verification, and computational efficiency.

Reasoning Models and Hidden Scratchpads: Modern reasoning models represent a significant evolution beyond CoT prompting. DeepSeek-R1 [7] demonstrates how reinforcement learning can train models to develop sophisticated internal reasoning capabilities through hidden thinking tokens. The S1 architecture [8] introduces dynamic reasoning token allocation, allowing models to adjust their computational budget based on problem complexity. Google's Gemini models [10] offer unique capabilities for our research, providing both automatic budget determination and explicit budget specification options—a feature that makes them particularly suitable for investigating verification costs.

Self-Verification and Self-Correction: The ability of LLMs to evaluate their own outputs has been extensively studied. [4] introduced Self-Refine, showing that models can iteratively improve outputs through self-feedback. [5] developed Reflexion, using reinforcement learning to teach models self-reflection. [11] demonstrated that dedicated verifier models can significantly improve performance on benchmarks like GSM8K. Our project adopts the concept of a separate verifier agent and focuses specifically on the computational cost associated with verification rather than the verification mechanism itself.

AI Control and Safety: [6] outline comprehensive strategies for AI control, emphasizing the importance of verification in ensuring safe AI deployment. Their framework highlights how verifier models can serve as cru-

cial safety checks, preventing potentially harmful outputs from reaching users—a motivation that underscores the practical importance of efficient verification.

LLM Inference Costs: The operational cost of deploying large models remains a significant barrier. Inference optimization research includes architectural innovations [12], quantization techniques [13], and deployment strategies [14]. [9] specifically examined scaling laws for verification, finding predictable relationships between problem complexity and verification costs at an aggregate level. Our work extends this by attempting instance-level prediction of optimal verification budgets.

GSM8K Dataset: We use the Grade School Math 8K (GSM8K) dataset introduced by [15], consisting of over 8,500 high-quality math word problems designed to test multi-step reasoning. It has become a standard benchmark for evaluating arithmetic reasoning capabilities and was notably used in the original Chain-of-Thought paper to demonstrate state-of-the-art performance.

3. Methodology

Our methodology is a multi-stage pipeline designed to first generate data on verification costs and then use that data to train a predictive model. The process is broken down into four key phases as illustrated in Figure 1.

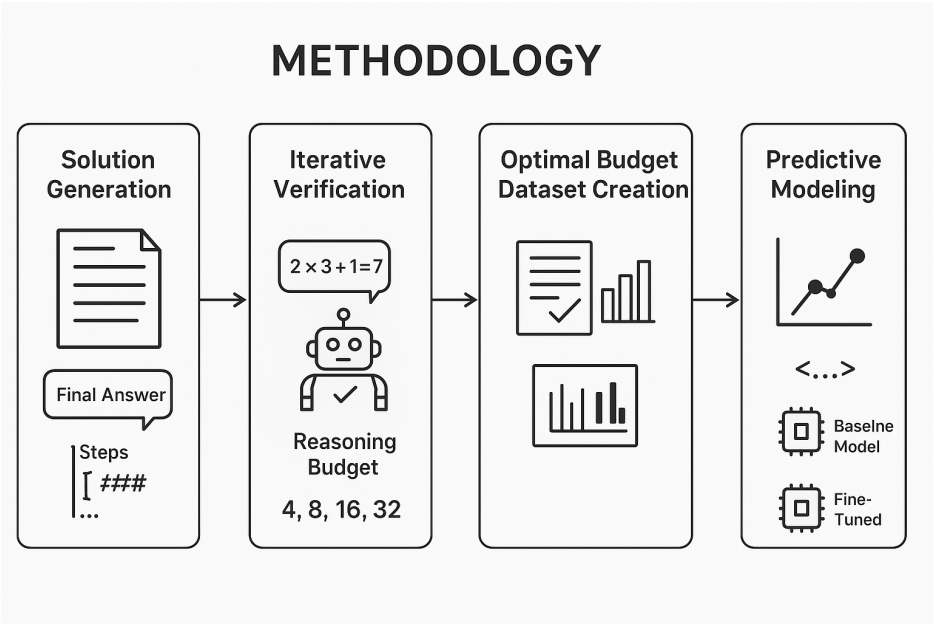

Figure 1: High-level overview of the project methodology.

1. **Solution Generation:** We first used a generator LLM (google/gemini-2.5-flash) to solve mathematical problems from the GSM8K dataset. For each problem, the model was prompted to produce a final numerical answer followed by a detailed, step-by-step explanation, with the two components separated by a special token (####). The correctness of each generated answer was determined by comparing it against the ground truth from the dataset.

2. **Iterative Verification with Budgeting:** A second LLM instance, acting as a verifier, was tasked with assessing the correctness of the solutions generated in the previous step. For each problem-solution pair, the verifier was prompted multiple times. Each verification attempt was conducted with a different, predefined reasoning token budget. We used a range of budgets: [4, 8, 16, 32, 64, 128, 256, 512] tokens. The verifier's task was to output a simple "true" or "false" judgment.

3. **Optimal Budget Dataset Creation:** The core of our data creation process was to identify the minimal computational cost for successful verification. For each problem, we analyzed the results from the iterative verification step. We defined the "optimal reasoning budget" as the smallest token budget that enabled the verifier to correctly assess the solution's validity (i.e., its output of "true" or "false" matched the ground

truth). This process yielded a new, derived dataset where each instance consists of a prompt (the original question and the generated solution) and a completion (the identified optimal reasoning budget).

4. **Predictive Modeling:** With the new dataset, we framed the final task as a multi-class classification problem: given the text of a problem and its proposed solution, predict the optimal reasoning budget required for verification. We developed and evaluated two models:

   - **Baseline Model:** A Logistic Regression classifier trained on text embeddings. The prompts were converted into high-dimensional vectors using the text-embedding-ada-002 model, and the classifier was trained to map these embeddings to one of the discrete budget classes.
   - **Fine-Tuned Model:** We set up an experiment to fine-tune a generative model (text-ada-001) directly on the prompt-completion pairs from our derived dataset. This approach trains the model to directly output the optimal budget as a text string.

4. Data Set

The primary dataset for this project is the Grade School Math 8K (GSM8K) dataset. It contains 8,500 high-quality, linguistically diverse word problems designed to be solvable by a bright middle school student. The solutions require between 2 and 8 steps of basic arithmetic operations.

From this foundation, we generated a novel, derived dataset specifically for our prediction task, which we call the **smallest_length_dataset**. This dataset was constructed by identifying the minimum reasoning budget required for successful verification for each problem in our initial run, as described in the methodology.

4..1 Data Statistics and Analysis

The resulting dataset for our classification task contains 7,473 samples. Our solution-generation process produced over 90 % correct answers, likely because the questions were relatively easy for the model. An analysis of the class distribution reveals a significant imbalance in terms of required verification budgets, as shown in Table 1.

Table 1: Distribution of Optimal Reasoning Budgets

Reasoning Budget (Tokens)	**Count of Problems**	**Percentage**
64	7,111	95.5%
128	39	0.5%
256	25	0.3%
512	28	0.4%
Total	**7,437**	**100%**

The most striking feature of our derived dataset is the overwhelming prevalence of the 64-token budget class, which accounts for over 95% of the data. This indicates that for the vast majority of the GSM8K problems, the verifier model required a relatively small and consistent amount of "thinking" to reach a correct conclusion. We hypothesized that this might be even lower and tested with budgets as low as 4 tokens, finding that over 90% of questions were still correctly verified.

This phenomenon can be explained by the fundamental asymmetry between generation and evaluation tasks. As noted by [16] in their work on AI safety via debate, evaluation is generally easier than generation—a principle that underlies many scalable oversight techniques. In our case, the verifier merely needs to identify a single error in the provided step-by-step reasoning to determine that a solution is false, rather than solving the entire problem from scratch. This evaluation advantage means that the verifier can often make correct judgments with minimal computational effort, especially when provided with detailed reasoning steps that make errors more apparent.

5. Experimental Setup

Our experiments were designed to evaluate the feasibility of predicting the optimal reasoning budget. We focused on baseline models to establish performance benchmarks while carefully considering model selection and design choices.

Models:

- **Generator & Verifier:** We chose google/gemini-2.5-flash for both generating initial solutions and subsequent verification steps. We selected Gemini over alternatives like OpenAI's o3 because, to the best of our knowledge, Gemini is the only model that provides explicit mechanisms for specifying token budgets directly. We opted for the Flash variant over Pro due to cost considerations while maintaining sufficient capability for our tasks.
- **Embedding Model:** text-embedding-ada-002 was used to convert textual prompts (question + generated solution) into numerical vectors for the baseline model.
- **Baseline Classifier:** We employed a LogisticRegression model from scikit-learn, configured with a maximum of 500 iterations to ensure convergence. We chose logistic regression as our initial baseline because it provides interpretable linear decision boundaries and serves as a simple benchmark. The linear nature of this model, while limiting its ability to capture complex non-linear relationships, allows us to establish a lower bound on performance that more sophisticated models should exceed.

Data Split:

The derived smallest_length_dataset was split into a training set (80%) and a testing set (20%) using a standard random split with a fixed seed for reproducibility.

Procedure:

1. Text prompts from training and testing sets were fed to the embedding model to obtain vector representations.
2. The Logistic Regression classifier was trained on embeddings and corresponding budget labels from the training set.
3. The trained classifier predicted budget labels for unseen embeddings in the testing set.
4. Performance was evaluated using overall accuracy and per-class accuracy to account for severe class imbalance.

Fine-Tuning Setup:

For the fine-tuning experiment, training data was formatted into a .jsonl file where each line contained a JSON object with "prompt" and "completion" keys. This file was prepared for submission to a fine-tuning API using the text-ada-001 model with 4 epochs.

6. Results

The evaluation focused on the performance of the baseline Logistic Regression model, providing a clear benchmark for this novel classification task. We also present findings from an analysis of the relationship between generator and verifier computational effort.

6..1 Evaluation Metrics

The primary metric is accuracy, defined as the proportion of correct predictions. Given the severe class imbalance in our dataset, we report both overall accuracy and per-class accuracy. Per-class accuracy is crucial as it reveals how well the model performs on minority classes, which are often the most challenging and interesting cases.

6..2 Baseline Model Performance

The Logistic Regression baseline achieved an overall accuracy of 94.6%. While this appears very high, it largely reflects the model's success in predicting the dominant "64-token" class. The per-class accuracy, shown in Table 2, provides a more nuanced picture.

Table 2: Per-Class Accuracy of the Baseline Logistic Regression Model

Reasoning Budget	Correctly Predicted	Total in Test Set	Per-Class Accuracy
64	545	546	99.8%
128	0	25	0.0%
256	0	5	0.0%
512	0	1	0.0%

The results clearly show that the baseline model essentially learned to be a majority-class classifier. It identified nearly all instances requiring a 64-token budget but failed to correctly classify a single instance from any other budget class. This performance underscores the difficulty of the task given the available data distribution and the limitations of linear models in capturing complex patterns.

6..3 Generator vs. Verifier Token Analysis

To better understand verification dynamics and potential compute overuse, we analyzed the relationship between reasoning tokens used by the generator and verifier models. Importantly, in these experiments, we allowed the verifier to automatically determine its own budget rather than constraining it, enabling us to observe natural verification behavior.

Figure 2 shows a scatter plot of generator vs. verifier reasoning tokens for correctly verified problems when the verifier was given unlimited budget to determine its own computational needs.

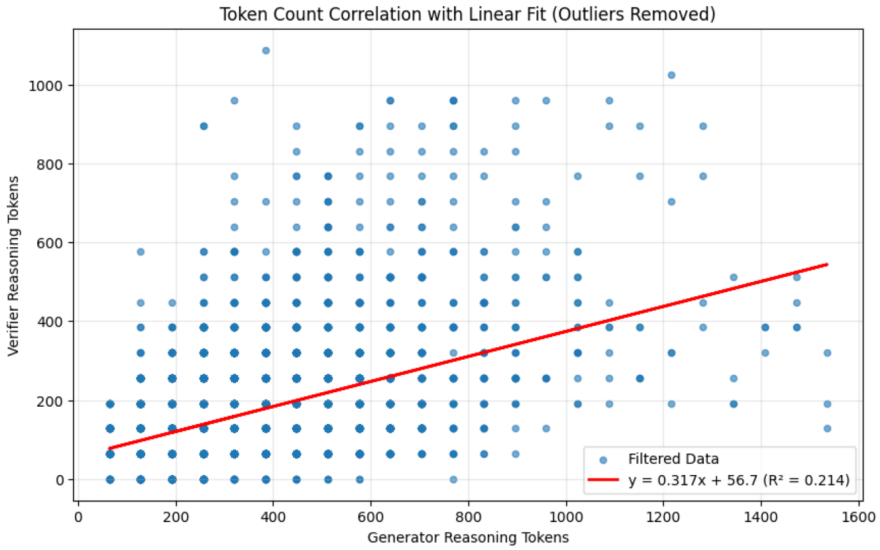

Figure 2: Generator vs. Verifier reasoning tokens for correctly verified problems (verifier with self-determined budget)

A statistical analysis yielded a Pearson correlation coefficient of 0.463, indicating a moderate positive linear relationship. When the generator "thinks" longer to produce a solution, the verifier also tends to "think" longer to verify it. However, the correlation is not strong, suggesting that verification effort is influenced by factors beyond generator effort.

To investigate compute efficiency, we compared the distribution of optimal budgets (from our controlled experiments) with the distribution of self-determined budgets when Gemini operates without constraints:

Table 3: Comparison of Optimal vs. Self-Determined Budgets

Budget Range	Optimal (Our Analysis)	Self-Determined (Gemini)
0-64 tokens	95.5%	42.3%
65-128 tokens	0.5%	31.2%
129-256 tokens	0.3%	18.7%
257+ tokens	0.4%	7.8%

This comparison reveals significant compute overuse: Gemini allocates substantially more verification tokens than our analysis suggests is optimal. While 95.5% of problems could be correctly verified with just 64 tokens, Gemini uses this minimal budget for only 42.3% of cases, indicating considerable room for efficiency improvements through better budget prediction.

7. Conclusion

7..1 Summary of Findings

This project successfully framed the problem of optimizing LLM verification costs as a predictive modeling task. Our key findings are:

- We developed a methodology to create a dataset for predicting optimal reasoning budgets for verification tasks, revealing that over 95% of GSM8K problems can be correctly verified with minimal (64-token) budgets.
- A baseline Logistic Regression model achieved high overall accuracy (94.6%) by exploiting severe class imbalance. However, its 0% per-class performance on minority classes highlights both the challenge of this task and the limitations of linear models for capturing complex verification patterns.
- We identified a moderate positive correlation (Pearson's r = 0.463) between generator and verifier reasoning tokens, suggesting that problem complexity influences both generation and verification effort, though not in a strictly linear fashion.
- Comparison of optimal budgets with self-determined budgets reveals significant compute overuse in current models, with Gemini using excessive tokens in nearly 58% of cases where minimal budgets would suffice.

7..2 Challenges Encountered

The primary challenge was profound data imbalance. The combination of the GSM8K dataset and the powerful gemini-2.5-flash model meant most verification tasks were "easy" and required the lowest common budget (64 tokens). This left insufficient data for the model to learn features distinguishing problems requiring higher budgets. The fundamental asymmetry between generation and evaluation—where verification merely requires finding errors rather than solving problems—contributed to this imbalance.

7..3 Interpretation of Outcomes

The results suggest that for this particular problem domain (grade school math) and model pairing, current verification approaches significantly overuse computational resources. While a sophisticated predictive model proved less effective than simple heuristics due to data distribution, our analysis reveals substantial efficiency gains are possible. The project demonstrates that predictive system effectiveness depends heavily on task complexity relative to model capabilities, and that evaluation tasks may require fundamentally different resource allocation strategies than generation tasks.

7..4 Future Work

This exploratory study opens several avenues for future research:

- **More Complex Datasets:** Applying this methodology to challenging reasoning datasets (e.g., advanced mathematics, legal reasoning, logic puzzles) where verification is non-trivial could yield more balanced budget distributions, making prediction more meaningful.

- **Non-Linear Model Architectures:** The linear classifier used in this study is fundamentally unable to learn non-linear relationships, which may be a core bottleneck. Exploring gradient-boosted trees, neural networks, or transformer-based classifiers could potentially capture subtle textual features that our baseline model missed.
- **Regression Formulation:** Framing the problem as regression—predicting exact token counts rather than discrete classes—could provide more granular control over resource allocation and better handle the continuous nature of computational requirements.
- **Feature Engineering:** Instead of relying solely on raw text embeddings, future work could engineer specific features correlating with complexity: number of solution steps, types of mathematical operations, problem statement length, or linguistic complexity markers.
- **Cross-Model Generalization:** Investigating whether budget prediction models trained on one LLM (e.g., Gemini) can generalize to others (e.g., GPT-4, Claude) would provide insights into universal verification complexity patterns.

References

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

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

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

[4] Aman Madaan, Niket Tandon, Prakhar Gupta, Skye Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Sean Welleck, et al. Self-refine: Iterative refinement with self-feedback. In *Advances in Neural Information Processing Systems*, volume 36, 2023.

[5] Noah Shinn, Beck Labash, and Ashwin Gopinath. Reflexion: an autonomous agent with dynamic memory and self-reflection. *arXiv preprint arXiv:2303.11366*, 2023.

[6] Ryan Greenblatt, Buck Shlegeris, Kshitij Sachan, and Fabien Roger. Ai control: Improving safety despite intentional subversion, 2024.

[7] DeepSeek-AI and Daya Guo et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning, 2025.

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

[9] Nishad Singhi, Hritik Bansal, Arian Hosseini, Aditya Grover, Kai-Wei Chang, Marcus Rohrbach, and Anna Rohrbach. When to solve, when to verify: Compute-optimal problem solving and generative verification for llm reasoning, 2025.

[10] Google. Gemini: A family of highly capable multimodal models. Technical report, Google, 2024.

[11] Debjit Paul, Mete Ismayilzada, Maxime Peyrard, Beatriz Borges, Antoine Bosselut, Robert West, and Boi Faltings. Refiner: Reasoning feedback on intermediate representations. *arXiv preprint arXiv:2304.01904*, 2023.

[12] Tri Dao, Daniel Y Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. Flashattention: Fast and memory-efficient exact attention with io-awareness. In *Advances in Neural Information Processing Systems*, volume 35, pages 16344–16359, 2022.

[13] Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. Gptq: Accurate post-training quantization for generative pre-trained transformers. In *International Conference on Learning Representations*, 2023.

[14] Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, and Ion Stoica. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th Symposium on Operating Systems Principles*, pages 611–626, 2023.

[15] Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Pavlov, Jordan Power, Aaron van den Oord, John Whalen, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

[16] Jan Leike, David Krueger, Tom Everitt, Miljan Martic, Vishal Maini, and Shane Legg. Scalable agent alignment via reward modeling: a research direction, 2018.