2025 IEEE 16th Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON) | 979-8-3315-6501-5/25/$31.00 ©2025 IEEE | DOI: 10.1109/UEMCON67449.2025.11267619

Trade-offs between Traditional LLM and Chain-of-Thought Reasoning for Mathematical Tasks

Dharani Ravanam
Tagliatela College of Engineering
University of New Haven
West Haven, CT, United States
draval@unh.newhaven.edu

Shivanjali Khare, Ph.D.
Tagliatela College of Engineering
University of New Haven
West Haven, CT, United States
skhare@newhaven.edu

***Abstract*—Large Language Models (LLMs) have progressed from basic text generators to advanced reasoning tools that can handle complex cognitive tasks. Chain-of-Thought (CoT) models (enable step-by-step reasoning) are now widely adopted for their enhanced problem-solving capabilities. This perspective, however, often overlooks that the performance gains from CoT are not universal. This study investigates the relation between accuracy, latency, and token consumption cost across different mathematical task complexities. An empirical analysis is conducted on two variants of the kimi family: traditional model (kimi-dev-72b) and CoT-enabled model (kimi-vl-a3b-thinking). These models were evaluated on 500 questions sampled from three diverse benchmark datasets: the GSM8K, Nguyen-Brat/asdiv, and SAT-MATH. The results show that traditional LLM achieves higher or equal accuracy, with significantly lower latency and greater token efficiency. Our findings indicate that for simpler problems, computational overhead generated by CoT models often outweighs the minimal increase in accuracy. These results highlight the need for dynamic reasoning frameworks, especially for real-time, cost-sensitive applications, such as adaptive prompting, where the type of LLM model used is based on the inferred complexity of the problem.**

***Index Terms*—Large Language Models, Chain-of-Thought, Mathematical problem solving, Empirical analysis, Computational cost comparison, Complex Queries, Explicit Reasoning, Simple Queries, Reasoning Ability, Pre-trained Language Models**

I. Introduction

CoT prompting mimics human-like sequential thinking and boosts the performance of traditional LLM on complex multi-step and logical tasks. These models allow LLMs to analyze their reasoning step-by-step before reaching a final answer. Several research also document the benefits of CoT in improving accuracy [1] [2] [3]. While CoT generally outperforms conventional models, the degree of improvement varies depending on the specific domain and task, and often diminishes as problem complexity increases. With wide adoption of these models, there is a need to analyze the cost-benefit of the entire reasoning process of these models. The lengthy responses produced by CoT inherently consume more tokens and potentially increase latency. In real-time applications, this becomes critical as it introduces delays and creates a disruptive experience for users. As LLMs evolve into more advanced models, an important question arises: *Is explicit reasoning through CoT always essential, or can newer, optimized traditional LLM models achieve comparable or even better performance without the added compute?*

This study conducts empirical analysis between a traditional LLM (kimi-dev-72b) [4] and a CoT-enabled version (kimi-vl-a3b-thinking) [5] provided by MoonshotAI. The goal is to evaluate their performance and efficiency across a variety of mathematical reasoning tasks. By systematically examining accuracy, latency, and token usage, this study provides trade-offs between these two models, guiding best practices for selecting and deploying LLMs for real-time applications.

II. Related Work

Recent research studies, such as AGIEval benchmark, highlight that LLMs are approaching self-stabilization, with zero-shot learning capabilities becoming increasingly comparable across models. However, the impact of CoT prompting is not consistently positive. While it

979-8-3315-6501-5/25$31.002025 IEEE

can improve average performance, it also leads to degradation in certain tasks. For instance, Text-Davinci-003's zero-shot accuracy on LogiQA (Chinese) dropped from 40.3% without CoT to 36.7% with CoT, demonstrating this variability [6].

The paper "Reasoning Models Don't Always Say What They Think" [7] provides deeper insights into CoT reasoning in LLMs. A major finding is that when an LLM's CoT does not accurately reflect its internal reasoning (i.e., "unfaithful" CoT), the explanations are typically longer and more complex. For example, the authors observe that Claude 3.7 Sonnet CoT misrepresented true reasons for its prediction, averaging to 2064 tokens, compared to 1439 tokens for correct reasoning, resulting in higher computational costs. This verbosity often arises from inefficient strategies such as checking all multiple-choice options instead of directly verbalizing a solution hint. The study further shows that CoT faithfulness declines as task difficulty increases, with Claude 3.7 Sonnet exhibiting a 44% relative drop on GPQA compared to MMLU. These findings show that CoT benefits are not universal and highlight the need for more adaptive reasoning frameworks.

Building on this understanding of CoT variability, Sprague et al. in [8] conduct a quantitative meta-analysis and empirical study showing that CoT prompts yield the strongest performance gains in tasks involving mathematics and logic. However, they also report much smaller gains — or even detrimental effects for tasks such as common sense reasoning or text classification. Their analysis also emphasizes that CoT inherently requires more computation than direct answering, with its utility for symbolic tasks largely stemming from enhanced symbolic execution, though still falling short of external symbolic solvers.

III. Methodology and Data Selection

Rationale for selecting Kimi-LLM : The Family of Kimi-LLM was chosen to compare a traditional LLM model with its CoT-enabled counterpart. For the traditional model: *moonshotai/kimi-dev-72b* was selected, and *kimi-vl-a3b-thinking* was selected as it's CoT variant. Both these models are freely available on OpenRouter and provide clear differentiated CoT and non-CoT variants. This minimizes external variability and architectural disparities while ensuring that observed performance differences can be attributed primarily to reasoning strategies, making them well-suited for controlled experimentation.

Rationale for using mathematical Tasks: To quantify the trade-offs between the two models, it is essential to have a precise measurement of accuracy. Mathematical problems provide a clear and structured framework, unlike open-ended dialogue or creative writing. A math problem has a single correct answer. As such, this study focuses on mathematical problem-solving to evaluate the reasoning abilities of traditional LLM and CoT models.

Three datasets were selected to represent a wide range of complexity in tasks. SAT-MATH [6] and Nguyen-Brat Asdiv (Asdiv) [9] contain varied problems. These two datasets contain simple, single-step operations like calculating 15×2.5 to more complex geometry or algebra word problems. GSM8K (Grade School Math) [10] dataset, on the other hand, requires multi-step reasoning, like between 2 and 8 steps to solve. Together, these datasets provide a powerful benchmark to assess LLM's fundamental reasoning and step-by-step logic.

Automated Evaluation Protocol: A standard pipeline, shown in figure 1, was designed to evaluate each model against each mathematical dataset. The following steps were performed for every problem in the dataset. For each run, the script selects specific `MODEL_NAME` to test and the target `sheet_to_process` (e.g., `GSM8K`). To store its results, a unique output (Excel) is generated for each model. To prevent data loss, the script also checks for an existing output file. If found, it loads previously saved results and resumes processing from the last incomplete row. This also provides a quick recovery mechanism in case of interruptions like redundant API calls. In the target sheet, each row is a question with columns being: *Options* (when applicable) and *Expected Answers*. A standard prompt is created for each question using data from the *Questions* column, while the *Options* column was incorporated to provide complete context for multiple-choice questions. Every API request was prefaced with the same system-level instruction: "`You are a helpful assistant designed to solve math problems. Provide a clear, step-by-step solution and state the final answer.`" This instruction served both models consistently. A complete prompt was sent to the designated model via the OpenRouter API. Upon receiving a response, the script systematically records and adds new columns to target sheet that stores the performance metrics.

Evaluation process: To analyze the trade-off of the entire process, the following metrics were used:

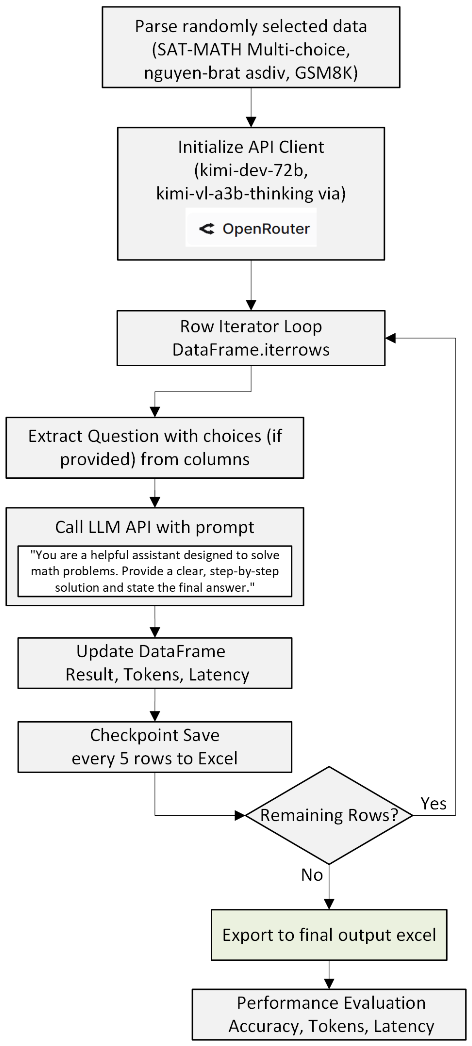
Parse randomly selected data (SAT-MATH Multi-choice, nguyen-brat asdiv, GSM8K)
Initialize API Client (kimi-dev-72b, kimi-vl-a3b-thinking via)
OpenRouter
Row Iterator Loop DataFrame.iterrows
Extract Question with choices (if provided) from columns
Call LLM API with prompt
"You are a helpful assistant designed to solve math problems. Provide a clear, step-by-step solution and state the final answer."
Update DataFrame Result, Tokens, Latency
Checkpoint Save every 5 rows to Excel
Remaining Rows?
Yes
No
Export to final output excel
Performance Evaluation Accuracy, Tokens, Latency

Fig. 1: End-to-end process for solving math problems with Large Language Models

- **Accuracy:** Calculated by manually evaluating each question and marking it as correct or incorrect. The total accuracy was determined using:

$$A_{acc} = \frac{\text{Problems of matched answers}}{\text{Total problems}}$$

- **Latency:** Measured as the time taken to receive a response from the API using `time()` before and after the request. This round-trip request time in seconds is a key metric to test real-time feasibility.
- **Token Usage:** The number of input, output, and total tokens consumed, extracted from the API response's usage, providing a measure of computational cost [11]

IV. Experimental Setup

All experiments were carried out with a 13th Gen Intel(R) Core(TM) i7-1360P processor running at 2.20 GHz, supported by 16 GB of installed RAM (15.7 GB usable). The system was configured as a 64-bit operating system on an x64-based architecture. This hardware environment was sufficient to execute API-based responses and inference requests without introducing bottlenecks.

The selected traditional and CoT models were accessed via the OpenRouter API and were evaluated on three datasets: SAT-MATH [6], Nguyen-Brat Asdiv [9], and GSM8K [10]. A total of 500 problems were randomly sampled from these three datasets, covering diverse reasoning tasks commonly used in LLM evaluation. This was due to the API request limitations, which were restricted to 40 requests per day per account. The selected subset includes: 100 questions from GSM8K, 200 from ASDiv, and 200 from SAT-MATH, covering a range of complexity in mathematical tasks. The automated protocol as described in the previous section was executed sequentially to establish a direct comparison. First, the entire evaluation was run for the **Traditional Model** (`moonshotai/kimi-dev-72b:free`) across all 500 questions to store the results. Next, the process was repeated for the **CoT-enabled Model** (`kimi-vl-a3b-thinking`). The only change in the script configuration between runs was the `MODEL_NAME` parameter to ensure identical experimental conditions.

The script incorporates features to handle errors, address throttling, and uses incremental checkpoints. An exponential backoff retry mechanism was implemented to handle server-side API errors, re-attempting a failed request up to four times. A 1.5-second delay was enforced between successful API calls to adhere to rate limits and to address throttling. To prevent data loss, all progress was saved to the output Excel file every five rows. This guaranteed that in the event of a critical failure, the experiment could be resumed with minimal loss of work.

V. Experimental Results

A. Accuracy

Figure 2 shows an accuracy comparison between the two models on the three datasets. Traditional LLM (`kimi-dev-72b`) achieved higher scores on SAT-MATH and GSM8K datasets and matched the performance of the CoT model (`kimi-vl-a3b-thinking`) on the Nguyen-Brat

ASDiv dataset. It is important to note here that on the most complex dataset, GSM8K, the traditional LLM model achieved a perfect score of 1.0, significantly outperforming the CoT model (0.96). This indicates that despite being more concise, traditional LLM possesses a more robust and effective reasoning capability.

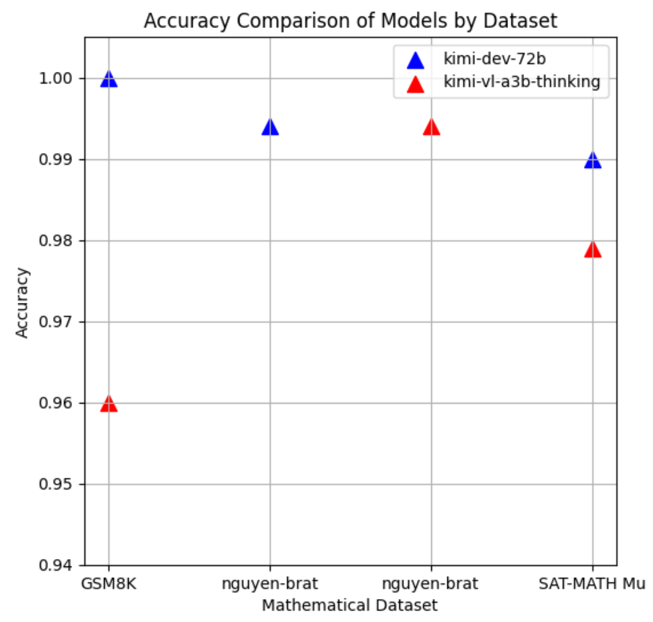

Fig. 2: Accuracy comparison or models ("kimi-dev-72b", "kimi-vl-a3b-thinking") across GSM8K, ASDiv, and SAT-MATH datasets.

B. Latency

Traditional LLM (`kimi-dev-72b`) performs significantly faster than the CoT model (`kimi-vl-a3b-thinking`) as seen in figure 3. It consistently delivered responses in approximately half the time across all datasets. For example, on the SAT-MATH tasks, traditional LLMs' average latency was 18.754 seconds, compared to 49.406 seconds for the CoT model (`kimi-vl-a3b-thinking`). These results suggest that traditional LLM models can be more suitable for user-facing or real-time applications.

C. Token Usage

Table I compares the average input and output tokens generated by each model. Results show that the traditional model (`kimi-dev-72b`) uses approx. 40–65% fewer tokens than CoT model (`kimi-vl-a3b-thinking`) to arrive at answers for each dataset. Table I also compares the average total tokens generated by two models on the three datasets. This indicates that traditional LLM is vastly more efficient in its token consumption. For instance, on the SAT-MATH dataset, traditional LLM (`kimi-dev-72b`) required 911 tokens versus 2342 tokens for CoT model (`kimi-vl-a3b-thinking`). The same pattern of lower average token count generated by the traditional model is observed across all datasets. This lower token count translates into substantially reduced operational costs.

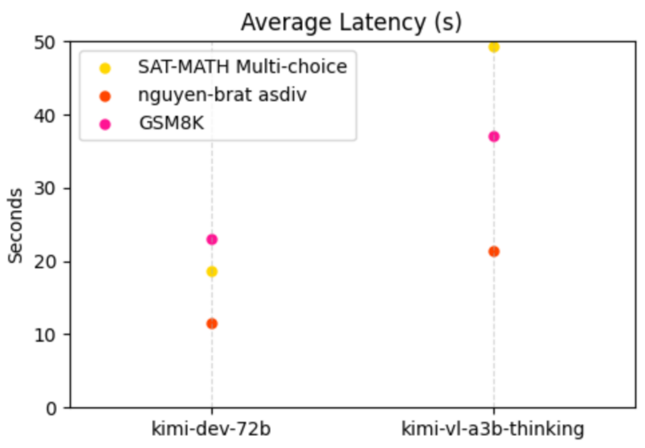

Fig. 3: Latency comparison (seconds) for models ("kimi-dev-72b", "kimi-vl-a3b-thinking") across GSM8K, ASDiv, and SAT-MATH datasets.

Average Total Tokens		
Dataset	**kimi-dev-72b**	**kimi-vl-a3b-thinking**
SAT-MATH Multi-choice	911.135	2341.682
nguyen-brat asdiv	512.04	968.654
GSM8K	604.737	1756.606

TABLE I: Average tokens for models ("kimi-dev-72b", "kimi-vl-a3b-thinking") comparison across GSM8K, ASDiv, and SAT-MATH datasets.

D. Interpretation of Results

The results show that the CoT-enabled model (`kimi-vl-a3b-thinking`) did not consistently outperform the traditional LLM model (`kimi-dev-72b`). This work contradicts common expectations from prior research [1] and validates that recent advancements in model architecture and training strategies have significantly enhanced the capabilities of traditional models, supporting some of the previous works [8], [12], [13]. The traditional LLM model not only matched but also exceeded CoT in accuracy with lower latency and fewer tokens. These results suggest that for relatively straightforward mathematical problems, especially those not requiring multi-step logic, a well-trained, efficient non-CoT model can outperform CoT-enabled models in both speed and cost-effectiveness.

VI. Challenges and Limitations

Several challenges were encountered during experimental evaluation. Processing large Excel files risked data corruption and was addressed by using incremental checkpoints for recoverability. Models occasionally produced irrelevant, endless outputs. To address this, timeout controls were implemented to prevent process hangs.

For empirical analysis between traditional and CoT models, selecting appropriate LLM models for research purposes is challenging. Open-source LLM models frequently become unavailable, while some have rate limits of free tiers and require paid APIs for long-term access. The use of free API tiers on OpenRouter introduced strict limitations (e.g., 40 requests/day) and instability in model availability, complicating large-scale experiments.

Additionally, both the models (traditional LLM and CoT) were also evaluated on a third, more complex mathematical dataset, AllenAI Lila dataset [14], that contains multi-step problems. While the selected models show effective results for basic arithmetic and algebra using standard prompts, they struggled to generate any results on questions selected from the complex AllenAI Lila dataset. This reinstates the importance of task-aligned model selection. The research is limited to SAT-MATH Multichoice, Nguyen-Brat, and GSM8K datasets, but it can be extended to similar datasets.

VII. Conclusion

CoT models may prove useful in domains requiring deeper reasoning and complex understanding (e.g., programming, logic-based tasks). This study compares the trade-offs between the traditional LLM model (`kimi-dev-72b`) and CoT-enabled model (`kimi-vl-a3b-thinking`) across mathematical datasets: GSM8K, Nguyen-Brat ASDiv, and SAT-MATH datasets. Overall, the results show the traditional LLM model performs better than the CoT model. Traditional LLM achieves higher or equal accuracy, responds faster, and consumes fewer average tokens, while CoT models add overhead. For simple to moderately complex mathematical problems. This study validates that model selection should be guided by task complexity, rather than defaulting to CoT, particularly in real-time or cost-sensitive applications. This approach prevents the LLM from generating unnecessarily complex explanations for simpler problems, improving efficiency and usefulness.

VIII. Future Work

Future research would be to extend this analysis beyond mathematical benchmarks to other complex domains. The evaluation can also be performed on different LLM families and models. Research includes key parameters such as accuracy, latency, token usage, and cost, but future work could explore better parameters to further evolve the system. Trade-off between accuracy and efficiency should be empirically analyzed for other tasks such as programming, legal reasoning, and medical decision support, etc. To get a deeper understanding, the benchmark dataset can be extended to include multi-modal and real-world problem-solving tasks. Additionally, replicating experiments across multiple LLM model families, such as GPT, Claude, Mistral, and LLaMA, should be validated to generalize findings. Another promising direction is the design of adaptive systems that dynamically decide when to invoke CoT based on task complexity, reducing unnecessary overhead. Such approaches could enable practical deployment of LLMs that balance accuracy, response time, and computational cost across diverse applications.

References

[1] 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. *Advances in neural information processing systems*, 35:24824–24837, 2022.

[2] Janice Ahn, Rishu Verma, Renze Lou, Di Liu, Rui Zhang, and Wenpeng Yin. Large language models for mathematical reasoning: Progresses and challenges, 2024. *URL https://arxiv.org/abs/2402.00157*, 2, 2021.

[3] Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*, 2022.

[4] Kimi-Dev Team. Introducing kimi-dev-72b: A strong and open coding llm for issue resolution, June 2025.

[5] Kimi Team, Angang Du, Bohong Yin, Bowei Xing, Bowen Qu, Bowen Wang, Cheng Chen, Chenlin Zhang, Chenzhuang Du, Chu Wei, Congcong Wang, Dehao Zhang, Dikang Du, Dongliang Wang, Enming Yuan, Enzhe Lu, Fang Li, Flood Sung, Guangda Wei, Guokun Lai, Han Zhu, Hao Ding, Hao Hu, Hao Yang, Hao Zhang, Haoning Wu, Haotian Yao, Haoyu Lu, Heng Wang, Hongcheng Gao, Huabin Zheng, Jiaming Li, Jianlin Su, Jianzhou Wang, Jiaqi Deng, Jiezhong Qiu, Jin Xie, Jinhong Wang, Jingyuan Liu, Junjie Yan, Kun Ouyang, Liang Chen, Lin Sui, Longhui Yu, Mengfan Dong, Mengnan Dong, Nuo Xu, Pengyu Cheng, Qizheng Gu, Runjie Zhou,

Shaowei Liu, Sihan Cao, Tao Yu, Tianhui Song, Tongtong Bai, Wei Song, Weiran He, Weixiao Huang, Weixin Xu, Xiaokun Yuan, Xingcheng Yao, Xingzhe Wu, Xinxing Zu, Xinyu Zhou, Xinyuan Wang, Y. Charles, Yan Zhong, Yang Li, Yangyang Hu, Yanru Chen, Yejie Wang, Yibo Liu, Yibo Miao, Yidao Qin, Yimin Chen, Yiping Bao, Yiqin Wang, Yongsheng Kang, Yuanxin Liu, Yulun Du, Yuxin Wu, Yuzhi Wang, Yuzi Yan, Zaida Zhou, Zhaowei Li, Zhejun Jiang, Zheng Zhang, Zhilin Yang, Zhiqi Huang, Zihao Huang, Zijia Zhao, and Ziwei Chen. Kimi-VL technical report, 2025.

[6] Wanjun Zhong, Ruixiang Cui, Yiduo Guo, Yaobo Liang, Shuai Lu, Yanlin Wang, Amin Saied, Weizhu Chen, and Nan Duan. Agieval: A human-centric benchmark for evaluating foundation models, 2023.

[7] Yanda Chen, Joe Benton, Ansh Radhakrishnan, Jonathan Uesato, Carson Denison, John Schulman, Arushi Somani, Peter Hase, Misha Wagner, Fabien Roger, et al. Reasoning models don't always say what they think. *arXiv preprint arXiv:2505.05410*, 2025.

[8] Zayne Sprague, Fangcong Yin, Juan Diego Rodriguez, Dongwei Jiang, Manya Wadhwa, Prasann Singhal, Xinyu Zhao, Xi Ye, Kyle Mahowald, and Greg Durrett. To cot or not to cot? chain-of-thought helps mainly on math and symbolic reasoning. *arXiv preprint arXiv:2409.12183*, 2024.

[9] Shen-Yun Miao, Chao-Chun Liang, and Keh-Yih Su. A diverse corpus for evaluating and developing english math word problem solvers. *arXiv preprint arXiv:2106.15772*, 2021.

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

[11] Haiwei Dong and Shuang Xie. Large language models (llms): Deployment, tokenomics and sustainability. *arXiv preprint arXiv:2405.17147*, 2024.

[12] Binquan Zhang, Li Zhang, Zhiwen Luo, Yuxin Du, Fang Liu, Song Wang, and Lin Shi. Are they all good? evaluating the quality of cots in llm-based code generation. *arXiv preprint arXiv:2507.06980*, 2025.

[13] Junhang Cheng, Fang Liu, Chengru Wu, and Li Zhang. Adaptivellm: A framework for selecting optimal cost-efficient llm for code-generation based on cot length. *arXiv preprint arXiv:2506.10525*, 2025.

[14] Swaroop Mishra, Matthew Finlayson, Pan Lu, Leonard Tang, Sean Welleck, Chitta Baral, Tanmay Rajpurohit, Oyvind Tafjord, Ashish Sabharwal, Peter Clark, and Ashwin Kalyan. Lila: A unified benchmark for mathematical reasoning. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2022.

Appendix

Dataset Reference Links:

- Nguyen-Brat ASDiv: https://huggingface.co/datasets/nguyen-brat/asdiv
- GSM8K: https://huggingface.co/datasets/openai/gsm8k
- SAT-MATH Multi-choice: https://huggingface.co/datasets/dmayhem93/agieval-sat-math
- AllenAI Lila: https://huggingface.co/datasets/allenai/lila