Finding the Sweet Spot: Trading Quality, Cost, and Speed During Inference-Time LLM Reflection

Jack Butler
Amazon Web Services
The United Kingdom
jackbtlr@amazon.co.uk

Nikita Kozodoi
Amazon Web Services
Germany
kozodoi@amazon.de

Zainab Afolabi
Amazon Web Services
The United Kingdom
zafolabi@amazon.co.uk

Brian Tyacke
Zalando
Germany
brian.tyacke@zalando.de

Gaiar Baimuratov
Zalando
Germany
gaiar.baimuratov@zalando.de

Abstract

As Large Language Models (LLMs) continue to evolve, practitioners face increasing options for enhancing inference-time performance without model retraining, including budget tuning and multi-step techniques like self-reflection. While these methods improve output quality, they create complex trade-offs among accuracy, cost, and latency that remain poorly understood across different domains. This paper systematically compares self-reflection and budget tuning across mathematical reasoning, text-to-SQL generation, sentiment classification, and translation tasks. We evaluate prominent LLMs from the Amazon Nova and Anthropic Claude families under varying reflection depths and compute budgets to derive Pareto-optimal performance frontiers. Our analysis reveals substantial domain-dependent variation in self-reflection effectiveness – with performance gains up to 220% in mathematical reasoning but mixed or negative effects in translation and SQL tasks. We further investigate how reflection round depth and feedback mechanism quality influence performance across model families. Additionally, our findings were validated through a real-world case study at Lounge by Zalando, where self-reflection showed market-dependent effectiveness, reinforcing the importance of domain-specific evaluation when deploying these techniques. Our results provide actionable guidance for selecting optimal inference strategies given specific domains and resource constraints.

Keywords

Large Language Models, Reasoning evaluation, Inference-time compute, Self-reflection

ACM Reference Format:
Jack Butler, Nikita Kozodoi, Zainab Afolabi, Brian Tyacke, and Gaiar Baimuratov. 2025. Finding the Sweet Spot: Trading Quality, Cost, and Speed During Inference-Time LLM Reflection. In *Proceedings of Workshop on Evaluation and Trustworthiness of Agentic and Generative AI Models (KDD'25).* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

KDD'25, Toronto, ON

ACM ISBN 978-x-xxxx-xxxx-x/YYYY/MM
https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 Introduction

Recent advances in large language models (LLMs) introduced methods to enhance inference-time performance by dynamically adapting sampling procedures on a per-input basis [20]. These approaches allow LLMs to achieve improved results without retraining, offering users fine-grained control over inference-time behavior based on task difficulty, available budget, and latency requirements.

Two established approaches for adjusting inference-time resources are multi-step inference and budget tuning. Budget tuning, available for select LLMs such as Anthropic Claude 3.7 Sonnet and OpenAI o1, enables users to configure inference parameters like maximum thinking tokens or reasoning tiers (e.g., low or high), allocating greater computational effort to more challenging inputs. Multi-step approaches such as self-reflection [17] are model-agnostic and involve directly prompting a model to revise its initial responses through sequential follow-up calls to the model.

Prior work demonstrates that self-reflection improves LLM performance in tasks with clearly defined evaluation criteria [17] and structured domains with informative feedback signals, such as programming or math [4]. For instance, executing generated code and providing the outputs back to the LLM as context creates concrete feedback signals for accurate self-reflection. However, many real-world tasks such as translation or classification involve more ambiguous objectives and weaker feedback signals. The effectiveness of self-reflection on such tasks remains underexplored, making it unclear whether additional computation consistently yields performance gains across diverse domains.

The rapid expansion of LLMs across diverse providers, model sizes, and supported inference regimes has created a fragmented landscape, introducing significant uncertainty for practitioners. For example, when selecting a model and inference configuration under given resource constraints, users often cannot determine whether employing a smaller model with advanced inference strategies might achieve desired accuracy more cost-effectively than adopting a larger LLM without such optimizations.

This paper makes two primary contributions. First, we benchmark self-reflection and budget tuning across multiple LLMs and application domains including mathematical reasoning, text-to-SQL generation, sentiment classification, and translation. The derived

Pareto-optimal frontiers illustrate accuracy-latency trade-offs of different strategies and provide actionable recommendations for selecting a suitable inference method based on domain-specific requirements, resource constraints, and the base model. Second, we analyze self-reflection trajectories across different LLMs and feedback mechanisms, revealing how reflection depth and feedback quality critically influence self-reflection performance. To the best of our knowledge, this work presents the first direct performance comparison between these two approaches.

2 Related Work

2.1 Inference-Time Compute

LLMs such as Anthropic's Claude family [2] have been increasing in size over recent years, which has brought profound improvements in performance across a wide range of applications while simultaneously increasing training time and compute requirements [8]. Inference-time compute optimization techniques avoid modifying the pre-trained model and instead enable dynamic allocation of computational resources depending on the specific requirements of each input. This offers the ability to tune performance according to task demands, scaling performance at inference time [20].

One of the prominent inference optimization approaches is self-reflection [17], which performs sequential follow-up calls to the LLM, allowing it to revise its initial responses. Other studies have explored drawing parallel samples from a language model and implementing more sophisticated sampling and verification procedures such as tree-of-thoughts [23] and graph-of-thoughts [3]. Recent work has also leveraged techniques such as temporary fine-tuning [1] and nearest neighbour retrieval-based fine-tuning [9] where model's parameters are temporarily updated during inference.

In this paper, we explore the trade-offs between two established inference strategies: model-agnostic self-reflection [17] and built-in reasoning capabilities exposed through some model provider APIs. Our goal is to provide insights for practitioners who lack the resources to conduct large-scale per-sample fine-tuning or complex multi-step inference processes with intricate feedback loops.

2.2 LLM Post-Training

Another area of research aimed at enhancing reasoning capabilities of LLMs is the use of reinforcement learning on specialised reasoning datasets. These approaches implement reinforcement learning with access to either outcome supervision [21, 22], step-by-step process supervision [14, 19] or LLM-driven feedback mechanisms [12].

Reasoning models are commonly deployed via API interfaces with configuration settings that allow users to adjust computational resources allocated per sample (e.g. Anthropic Claude 3.7 Sonnet thinking tokens budget). Crucially, the reasoning in these models happens as internal processing tokens, and discrete proposed solutions are not, to our knowledge, validated using external feedback during generation.

2.3 LLM Evaluation

As LLMs have become more capable and businesses increasingly incorporate them into their products and services, robust evaluation frameworks have become essential. Various evaluation platforms exist across different domains, such as HELM [13] and Chatbot Arena [5], where models undergo evaluation and receive automated or human feedback scores depending on the task and domain.

While these platforms provide standardised evaluation of different LLMs, including base models and fine-tuned or quantised variants, there is a lack of comparable evaluation frameworks for inference techniques. This gap makes it challenging for practitioners to understand and navigate trade-offs when combining LLMs with various inference-time compute methods. Throughout our experimentation, we demonstrate these trade-offs across model families, task domains, and inference budgets.

3 Experimental Setup

3.1 Datasets

We perform experiments across four distinct domains using established benchmarks:

- **Flores-200 (Translation)** [6]: Multilingual translation benchmark spanning 200 languages, allowing assessment of cross-lingual capabilities.
- **Math500 (Mathematical reasoning)** [15]: Dataset containing 500 problems across algebra, arithmetic, and word problems, testing symbolic manipulation and logical reasoning.
- **Spider (Text-to-SQL)** [24]: Complex text-to-SQL task involving 200 databases with multiple tables, evaluating structured SQL query generation capabilities.
- **IMDB Reviews (Sentiment analysis)** [16]: Binary sentiment classification dataset on movie reviews, assessing natural language classification performance.

The diverse set of tasks allows us to evaluate structured mathematical and programming reasoning (Math500, Spider) and natural language understanding (Flores-200, IMDB) capabilities. Due to resource constraints, we use a subset of each dataset for evaluation. For Spider, we use tasks from 5 databases (voter_1, battle_death, museum_visits, employee_hire_evaluation, orchestra). For Flores-200, we sample 200 examples across 15 language pairs to ensure cross-linguistic variation (English to Arabic, German, Spanish, French, Hebrew, Hindi, Italian, Japanese, Korean, Dutch, Portuguese, Russian, Turkish, Chinese, Polish). For IMDB and Math500, we randomly sample 100 examples.

3.2 Models and Inference Techniques

We compare performance across 7 model variants from 2 providers:

- Amazon Nova (Premier, Pro, Micro, Lite [10])
- Anthropic Claude (Sonnet 3.7, Sonnet 3.5 v2, Haiku 3.5 [2]).

For inference, we employ self-reflection across all LLMs using 0, 1, and 3 reflection rounds. Self-reflection is implemented through repeated model invocations, where at the end of each round we prompt the model to reflect on its response and update it if necessary. This implementation of self-reflection does not take advantage of any prompt caching [7] mechanisms, processing the full conversation history with each round of reflection. It's worth noting that, based on our understanding of the budget tuning implementations, self-reflection would uniquely benefit from prompt caching resulting in 10x potential reductions in cost and latency.

On the text-to-SQL task, we also compare 2 feedback mechanisms, which provide additional context before each self-reflection round. For Claude 3.7, we additionally utilize the built-in reasoning mode by defining 2 thinking budgets (4096 tokens and 1024 tokens). We refer to these thinking budgets as high and low, respectively.

We run experiments using Amazon Bedrock, maintaining default temperature and inference parameters associated with each model to ensure fair comparison. The cost per input/output token are recorded as of 02/05/2025, assuming on-demand pricing. Latency is measured as the total elapsed time between the prompt input and the completion of the full response. Prompts used for each of the 4 domains, as well as for self-reflection and feedback mechanisms, are available in the Appendix.

3.3 Evaluation Metrics

We employ task-specific metrics for each dataset to evaluate LLM performance. For translation (Flores-200), we use METEOR [11], which accounts for both precision and recall while handling synonyms and paraphrases. Sentiment analysis quality (IMDB) is assessed using classification accuracy on the binary prediction task. Spider and Math500 use additional verification procedures.

For Math500, we evaluate accuracy through normalized comparison and symbolic verification. We use string matching on normalized and cleaned LaTeX expressions, followed by symbolic equivalence checking with SymPy [18] to identify mathematically equivalent answers even when expressed differently. For Spider, we assess both exact matches and functional equivalence by executing SQL queries, discarding failures, and comparing normalized result tables against ground truth. When exact row matches are not found, we calculate partial credit based on matching cell values.

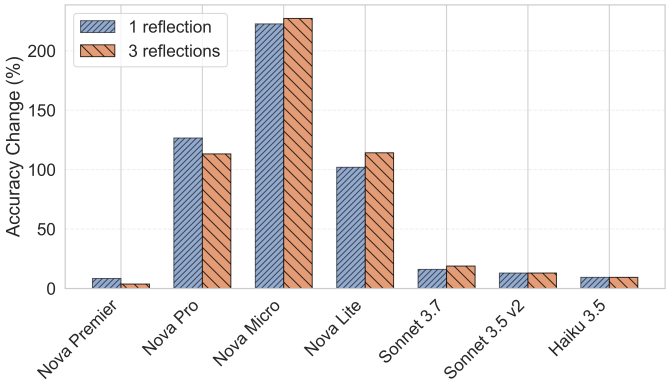
4 Results

This section overviews empirical results. For each dataset, we illustrate the percentage change in accuracy relative to zero reflections for each model, highlighting improvements for each configuration. Next, we construct Pareto-optimal frontiers showing accuracy-latency trade-off when employing inference strategies and provide cost information for each model and strategy combination. In Section 5, we dive deeper on how performance of self-reflection is affected by different factors.

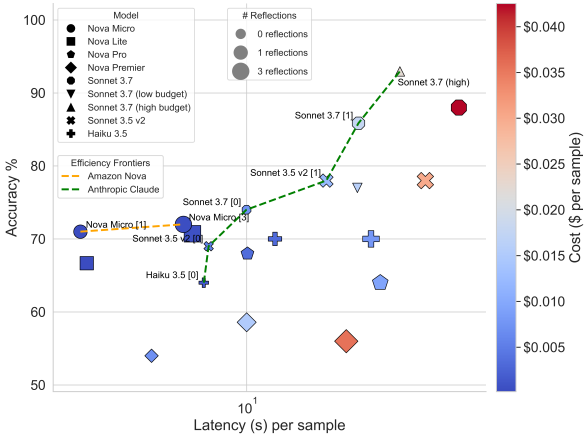
4.1 Mathematical Reasoning (Math500)

As depicted on Figure 1(a), all LLMs benefit from self-reflection. Amazon Nova Micro shows the largest gains, with accuracy improving by approximately 220% with 1 reflection and maintaining this improvement after 3 reflections. This suggests that Amazon Nova Micro's base mathematical reasoning capabilities are significantly enhanced through iterative self-correction. Similarly, Amazon Nova Lite and Pro show substantial improvements of approximately 100-130% with reflection, indicating that smaller models in the Amazon Nova family particularly benefit from reflection in mathematical reasoning.

In contrast, Amazon Nova Premier and Claude LLMs exhibit more modest but consistent improvements from reflection. Sonnet 3.7 shows approximately a 16% increase in accuracy with 1 and 20% with 3 reflections. Sonnet 3.5 v2 and Haiku 3.5 demonstrate similar patterns with gains of 13% and 9%, respectively. These results suggest that while Claude models benefit from reflection in mathematical tasks, their initial performance is already strong, resulting in less dramatic relative improvements.



(a) Relative Self-Reflection Gains



(b) Accuracy-Latency Pareto Frontiers

Figure 1: Inference-Time Techniques Performance (Math500)

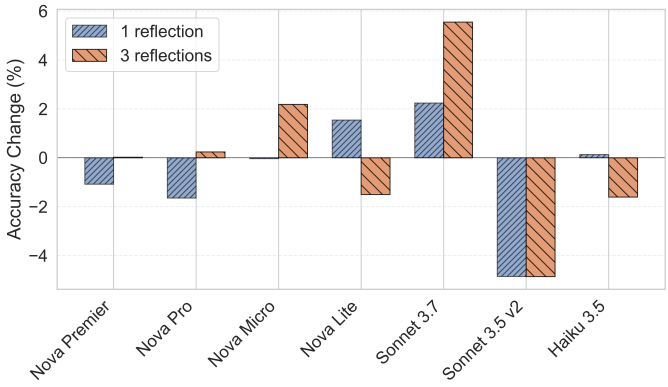
Figure 1(b) provides the absolute accuracy measurements across model configurations. Overall, Sonnet 3.7 demonstrates superior mathematical reasoning capabilities, starting with a baseline accuracy of 74% without reflection and improving to 86% with 1 and 88% with 3 reflections. Amazon Nova Micro starts with at just 22% accuracy without reflection (omitted from the plot) but jumps to 71% accuracy with 1 and 72% with 3 reflections. This pattern is similar for other Amazon Nova and Claude LLMs and suggests that for mathematical reasoning, a single well-implemented reflection round captures most of the potential performance benefit, with diminishing returns for additional rounds.

The Pareto frontier for the Claude family offers a rich selection in the accuracy-latency space, ranging from Haiku 3.5 with no reflections that offers 64% accuracy at $0.0015 per example and latency of 7.51 seconds, up to Sonnet 3.7 with a high thinking budget, which reaches 93% accuracy at $0.0224 and 27.88 seconds

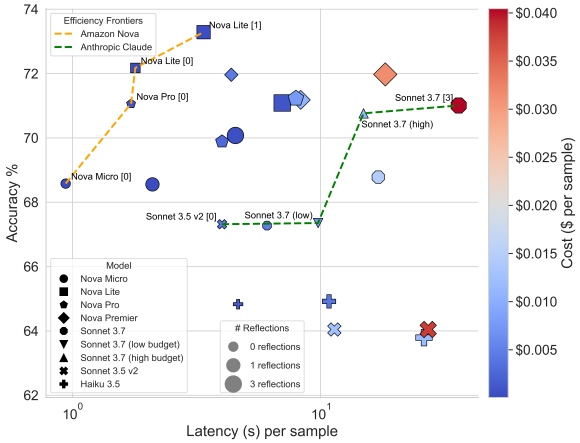
latency. At the same time, Sonnet 3.7 with a low thinking budget is dominated by Sonnet 3.7 with 1 self-reflection, which reaches a higher accuracy at the same latency. Considering the Amazon Nova family, we observe that Amazon Nova Micro with 1 and 3 reflections dominate Haiku 3.5 and Sonnet 3.5 in low-latency space but can not reach the same accuracy as higher-end Claude models even with self-reflection. This implies that practitioners should consider Amazon Nova Micro with self-reflection under strict cost/latency constraints and switch to Sonnet 3.7 with high reasoning for the best performance.

4.2 Text-to-SQL (Spider)

In contrast to Math500, in text-to-SQL generation Sonnet 3.7 is the only model to show consistent and limited improvements with additional reflections, gaining 2.3% accuracy with 1 round and a 5.6% gain with 3 rounds. Most other LLMs show mixed or negative responses to self-reflection. Sonnet 3.5 v2 demonstrates the most pronounced quality degradation, with accuracy declining by approximately 4.8% with 1 and reflection rounds. Similarly, Amazon Nova Pro and Haiku 3.5 show noticeable performance decreases with added reflection rounds.



(a) Relative Self-Reflection Gains



(b) Accuracy-Latency Pareto Frontiers

Figure 2: Inference-Time Techniques Performance (Spider)

Amazon Nova Micro is the only model besides Sonnet 3.7 showing positive outcomes with additional reflections. It maintains neutral performance with one reflection round and achieves a 2.2% accuracy improvement with 3 reflections. Amazon Nova Lite displays an inconsistent pattern, with a slight improvement (1.5%) at 1 reflection but declining by 1.5% with 3 rounds. Overall, these results suggest that self-reflection is less useful in domains like text-to-SQL, where revising the generated query without any additional context may mislead LLMs to change previously correct SQL queries.

The Amazon Nova Pareto frontier on Figure 2(b) represents optimal configurations for lower-latency applications. Overall, Amazon Nova models consistently outperform Claude LLMs, with 2 Amazon Nova Lite variants dominating all Claude counterparts. Amazon Nova Lite with 1 reflection achieves the highest absolute accuracy (74%) with approximately 3-second latency, while Amazon Nova Micro with 0 reflections offers the fastest and cheapest option to reach 68% accuracy. The Claude frontier represents a different set of trade-offs, with Sonnet 3.7 using 3 reflections achieving 71% accuracy but at a substantially higher latency (>35 seconds) and cost. Interestingly, built-in reasoning modes with both budget sizes fall behind the model variant with 3 reflections in terms of the accuracy, but are available at a lower latency and cost.

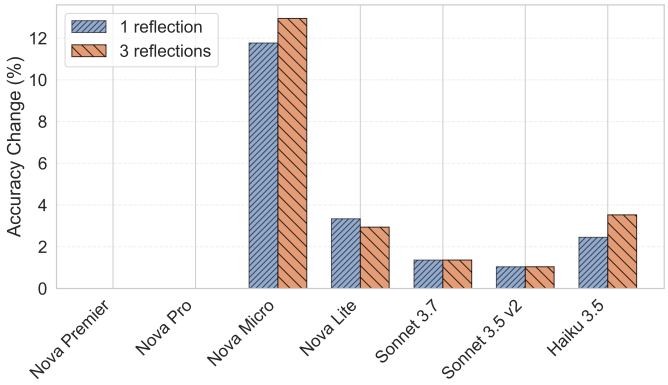
These results highlight the importance of model-specific optimization strategies for SQL tasks, with Amazon Nova models generally performing best with minimal reflection, while Claude Sonnet 3.7 uniquely benefits from both reflection and built-in reasoning despite the increased latency and cost.

4.3 Sentiment Classification (IMDB)

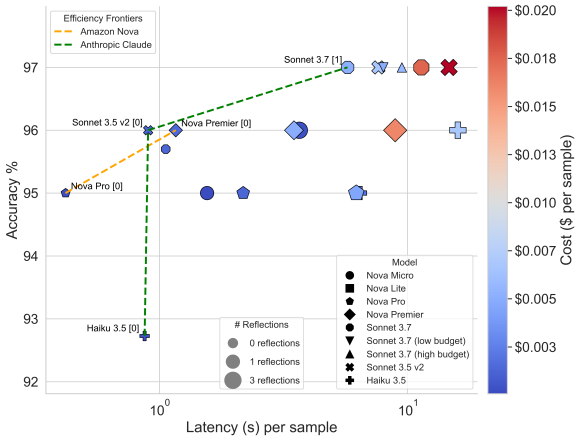
On sentiment analysis, Figure 3(a) clearly illustrates the positive impact of self-reflection on the accuracy across the LLMs. For the most models, adding reflection rounds improves accuracy, though with diminishing returns after the 1st reflection. Amazon Nova Micro shows the highest relative improvement from 0 to 1 reflections (85% to 95% accuracy), while maintaining similar latency to 0 reflections of Sonnet 3.7 (1.56 vs 1.06) and similar resulting accuracy (95% vs 95.7%) at 1/18th of the cost. Amazon Nova Pro and Premier are the only models whose accuracy is not affected by reflection.

As depicted on Figure 3(b), for applications requiring the highest possible accuracy, Sonnet 3.5 without reflection or Sonnet 3.7 with 1 reflection round offer the best performance. Built-in reasoning in Claude 3.7 performs similar to 1 round of self-reflection regardless of the thinking budget, but introduces higher latency and cost, making these configurations less attractive. For cost-sensitive deployments with moderate latency requirements, Amazon Nova Premier with 0 reflections presents a good compromise. Interestingly, Amazon Nova Micro with 3 reflections is able to reach a higher accuracy compared to Amazon Nova Premier, but results in a substantially higher overall latency and a marginally higher cost.

The results indicate that despite the ambiguity of the sentiment analysis task, LLMs consistently benefit from self-reflection in this domain. At the same time, the average gains are one order of magnitude smaller compared to mathematical reasoning, which makes inference-time techniques less attractive in terms of the cost-latency implications that may outweigh the accuracy gains.

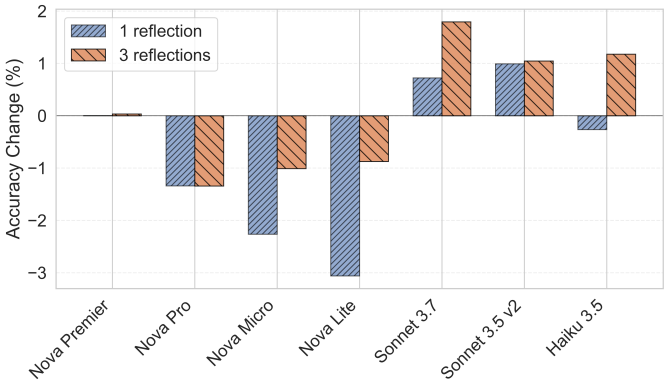


(a) Relative Self-Reflection Gains

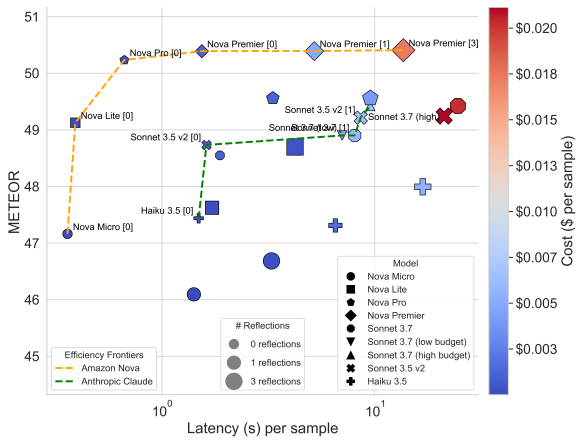


(b) Accuracy-Latency Pareto Frontiers

Figure 3: Inference-Time Techniques Performance (IMDB)



(a) Relative Self-Reflection Gains



(b) Accuracy-Latency Pareto Frontiers

Figure 4: Inference-Time Techniques Performance (Flores-200)

4.4 Translation (Flores-200)

Figure 4(a) highlights distinct divergence in translation performance patterns across model families. Claude models generally demonstrate enhanced performance after reflection. In contrast, all Amazon Nova models except Amazon Nova Premier exhibit an inverse trend, where incorporating 1 reflection diminishes translation accuracy. Despite some recovery when increasing from 1 to 3 reflection rounds, Amazon Nova Micro, Lite an Pro still under-perform compared to their baseline configurations with 0 reflections. This implies that using self-reflection for Amazon Nova LLMs in translation tasks is not recommended.

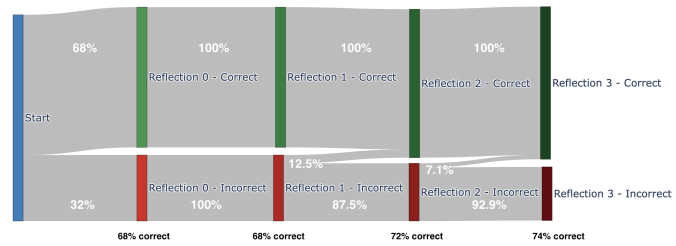
Figure 4(b) suggests that Amazon Nova models dominate all considered Claude models in the latency-accuracy space. Amazon Nova Pro reaches higher accuracy when all Claude variants at a lower latency compared to Haiku 3.5. Furthermore, Amazon Nova Premier with a different number of reflections provides a further marginal gain in translation accuracy, but brings a substantial latency increase. This suggests that Amazon Nova models perform particularly well in translation tasks, with an important caveat that integrating self-reflection for smaller variants hurts their performance. Focusing on Claude family, we note that Sonnet 3.7 built-in reasoning with a high thinking budget achieves the best METEOR score among Claude models, outperforming low thinking budget, self-reflections, and other Claude variants.

5 Ablation Studies

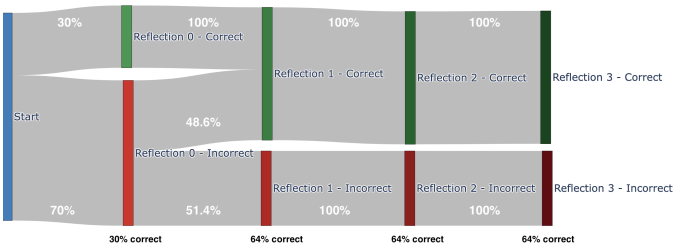
5.1 Reflection Transitions

Figure 5 illustrates how LLM performance evolves throughout multiple self-reflection rounds. We focus on mathematical reasoning, which proves to benefit most from reflection and showcase results for 2 LLMs from different model families with distinct performance patterns: Claude Sonnet 3.5 and Amazon Nova Micro. Results for other models are provided in the Appendix.

Claude Sonnet 3.5 v2 demonstrates superior initial accuracy at 68% compared to Amazon Nova Micro's 30%. Through 3 reflection stages, Sonnet 3.5 shows consistent incremental improvements, ultimately reaching 74% accuracy. Interestingly, while the first reflection does not change Sonnet's accuracy, each subsequent round successfully corrects a portion of initially incorrect responses. In



(a) Claude Sonnet 3.5 v2



(b) Amazon Nova Micro

Figure 5: Error Distributions Across Self-Reflection Steps (Math500)

contrast, Amazon Nova Micro exhibits a dramatic improvement during the first reflection, jumping to 64% accuracy after successfully correcting 48.6% of its initial errors.

However, Amazon Nova's performance plateaus thereafter, showing no further improvement in subsequent reflection stages. Another notable pattern across both LLMs is their perfect preservation of initially correct responses throughout all reflection rounds. These findings suggest that smaller models like Amazon Nova Micro have considerable capacity for initial self-correction, whereas more capable LLMs like Claude 3.5 Sonnet v2 have both stronger foundational performance and greater potential for continuous improvement through iterative reflection rounds.

5.2 Reflection Feedback

Table 1 investigates if providing informative feedback to the LLM between self-reflection rounds facilitates stronger accuracy gains. We focus on text-to-SQL and compare 2 feedback mechanisms as LLM context: i) output of SQL query execution; ii) LLM-as-a-judge response with Amazon Nova Pro judge.

The results reveal no clearly dominating feedback strategy. On average, incorporating feedback mechanisms improves reflection quality in 61% of cases, confirming that additional feedback can be beneficial. However, model families respond differently to feedback types: Amazon Nova generally performs better with LLM-as-judge feedback or no feedback at all, while Claude shows higher accuracy with SQL execution feedback. This may be partly explained by the fact that Amazon Nova Pro judge is not able to provide stronger feedback to Claude models compared to their own reasoning, which may risk getting them off track. These findings emphasize the importance of identifying optimal configurations for specific business applications by accounting for resource constraints, the particular LLM being used, and the task domain. Throughout our experiments, we consistently find that no single inference optimization strategy proves universally effective across the diverse range of scenarios we tested.

5.3 Prompt Caching

Prompt caching, as described generally in [7], is a set of techniques for caching computed model states so they can be re-used over future invocations of an LLM. Amazon Bedrock has released a prompt caching feature which allows users to set cached checkpoints during their conversation history and then save on the cost of recomputing these past messages. This capability is often used to cache very long initial system prompts or initial context which is used across multiple interactions with the LLM. Additionally, our results such as Figure 3 have shown that while self-reflection can bring improved performance, the additional cost and latency can hurt the feasibility of integrating these techniques.

Self-reflection, as we have defined in the earlier sections, has the potential to benefit from the prompt caching approach as we are frequently asking the model to reflect on past messages and revise the response. This is different to reasoning models such as Claude Sonnet 3.7, as their thought process is typically contained within the internal thinking tokens and not explicitly defined as sequences of messages in a conversation chain, preventing the use of prompt caching features available in Amazon Bedrock.

To analyse this trade-off further, we explore the differences in cost and latency across multiple rounds of self-reflection with and without leveraging the prompt caching feature in Amazon Bedrock. Figure 6 shows the cost and latency for a typical sequence of self-reflection rounds, with the model prompted to solve a Text-to-SQL question using an initial prompt size of approximately 1,000 tokens. Interestingly, Figure 6a shows that prompt caching combined with self-reflection has minimal benefits in terms of reducing the latency. We hypothesise that this could be due to the additional overhead of reading from cache databases being approximately equal to the latency required to generate the relatively minimal 100's of tokens. However, Figure 6b demonstrates that integrating self-reflection with prompt caching can being up to 28% reduction in cost when sampling over 3 rounds of reflection.

This method allows for more cost effective, linear scaling of self-reflection where only the incremental cost of additional output tokens is expensed with each round of reflection. For practitioners, it implies that leveraging self-reflection techniques can be more valuable with the LLMs and model providers that support prompt caching, as it offsets a significant part of additional costs on reflection rounds. We see potential for these techniques to grow in impact as more model providers enable improved prompt caching mechanisms in the future.

6 Use Case Study: Marketing Content Localization at Lounge by Zalando

To demonstrate the practical application of our findings in an industry setting, we evaluate self-reflection on a real-world dataset of marketing content localization provided by Lounge by Zalando.

Table 1: Impact of Feedback Mechanisms on Self-Reflection

Model	No feedback		LLM judge feedback		SQL execution feedback	
	1 round	3 rounds	1 round	3 rounds	1 round	3 rounds
Amazon Nova Premier	72.58	**74.98**	73.97	72.58	73.74	71.14
Amazon Nova Pro	71.75	**73.67**	71.71	66.96	68.62	73.50
Amazon Nova Lite	75.41	73.05	**79.57**	74.02	72.63	72.83
Amazon Nova Micro	70.73	72.14	**77.34**	75.77	73.15	70.41
Claude Sonnet 3.7	70.78	72.69	70.82	66.78	67.20	**73.32**
Claude Sonnet 3.5 v2	65.71	64.99	67.28	65.43	67.22	**67.33**
Claude Haiku 3.5	67.09	66.36	68.16	68.64	68.56	**72.58**

This case study serves to validate our benchmark results and provide actionable insights for similar production deployments.

Zalando is an online multi-brand fashion destination with more than 52 million active customers. Lounge by Zalando represents a shopping club, where customers can browse through a curated selection of fashion products. One of the critical business tasks at Lounge By Zalando is localizing the marketing content for different European markets, including different channels such as newsletters, push notifications, and display. In addition to translating the content to a local language, localization includes adjusting the content to follow local tonality and style guides developed by Zalando (e.g. using formal or informal pronouns when referring to a customer), adhering to local regulations and legal terminology (e.g. use correct terms for different sales types), and ensuring consistent brand voice.

To reduce the time spent on localization and increase the speed to market, Lounge by Zalando partnered with AWS to build a generative AI powered localization tool that incorporates Zalando's tonality guides and supports 17 European languages. This paper uses data from one of the marketing campaigns to test performance of self-reflection. The dataset consists of 102 marketing content examples, including email newsletter items and push notification texts in English. We also have ground truth localized versions produced by copywriters for three markets: Germany, France and Spain. Example English items from the campaign are illustrated below:

- Let your style speak for itself! Shop in-style brands now
- Hard to dress to impress on a budget? Not with these styles
- Seasons change but style stays

Based on our findings in Section 4, we select the most promising inference configurations for this task and compare a setting with no self-reflection to a single self-reflection round. We also leverage a feedback mechanism, where LLM-as-a-judge produces an evaluation report that analyses the generated localization against the tonality guidelines using multiple evaluation criteria developed by Zalando and suggests possible revisions as context for the second-round LLM call. We fix the base model to Claude 3.5 Sonnet, as it demonstrated the best performance in the no-reflection localization setting in our preliminary experiments.

To evaluate the localization quality, we use three metrics:

- BLEU score comparing the generated and ground truth localizations averaged across the dataset;
- METEOR score comparing the generated and ground truth localizations averaged across the dataset;
- LLM-as-a-judge score. The judge picks the best localization out of the generated and ground truth versions, without knowing which is which. The metric is then aggregated across the dataset to calculate the share of examples where the judge prefers the generated version or indicates a tie. Here, we use Claude 3.5 Sonnet as a judge.

Table 2 illustrates the localization results on three markets. Considering the LLM-as-a-judge metric, we see that self-reflection improves the localization quality on all languages, with the strongest gains observed for German. Here, the number of cases where the generated localization is better than or the same quality as the human translation increases from 38% to 47%. On French and Spanish markets, the no-reflection version already demonstrates better performance with 61% and 49%, correspondingly, which diminishes the observed gains from self-reflection.

Considering the text similarity metrics BLEU and METEOR, we observe mixed results with consistent improvements from self-reflection on the German market, its negative impact on localization quality on the French market and similar performance with and without reflection on the Spanish market. It is important to note that manual inspection of selected localizations indicated that LLM-as-a-judge provides a more reliable quality measure, as similarity metrics do not incorporate the tonality guidelines and do not account for multiple accepted alternatives of formulating a sentence.

Overall, the results indicate it is valuable to employ self-reflection on markets with more challenging localization rules (such as German), whereas using it on markets where the base model already achieves high quality provides minimal quality gains that may not justify the additional LLM cost. While this confirms some of the findings from Sections 4 and 5, it also emphasizes the importance of testing self-reflection performance on a specific dataset, as the gains may vary significantly depending on the use case.

7 Conclusion

This paper presents a systematic analysis of inference optimization techniques such as self-reflection and budget tuning across different domains, base models, and reflection parameters. We derive Pareto frontiers in accuracy-latency space and provide actionable recommendations to practitioners regarding suitable inference optimization methods for real-world applications.

Our results reveal no universally dominant inference strategy, with both the magnitude and direction of performance impacts

Table 2: Self-Reflection Performance on Real-World Marketing Content Localization Task

Language	No reflection			Self-reflection with LLM judge feedback		
	BLEU	METEOR	LLM judge score	BLEU	METEOR	LLM judge score
German	0.32	0.61	0.38	**0.33**	**0.62**	**0.47**
French	**0.16**	**0.47**	0.61	0.14	0.42	**0.62**
Spanish	0.29	**0.61**	0.49	**0.29**	0.59	**0.50**

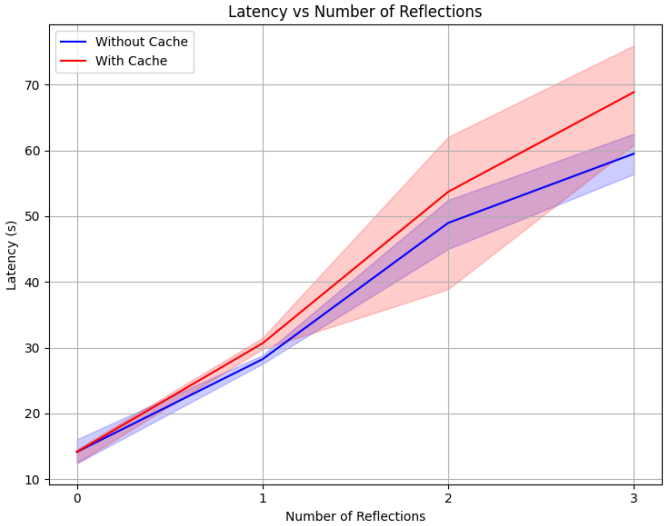
varying significantly across tasks. Self-reflection consistently improves performance in math (with gains up to 220%) and sentiment analysis, while showing mixed or negative effects in translation and text-to-SQL generation. This domain-specific variability highlights the importance of empirical evaluation before deploying inference optimization techniques in production.

Several key patterns emerge from our analysis: (i) smaller models often benefit more dramatically from reflection than larger ones; (ii) a single reflection round frequently captures most potential performance benefits; (iii) in several cases, smaller models with reflection outperform larger models without it, offering potential cost savings; and (iv) Claude's built-in reasoning sometimes underperforms compared to manual self-reflection techniques despite its specialized design, and results in higher additional cost as it does not benefit from prompt caching.

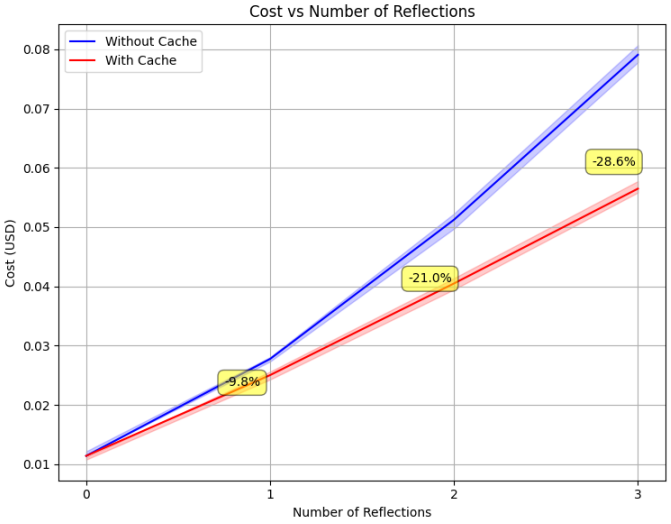
For practitioners, these findings suggest task-specific optimization strategies. For math, self-reflection is highly recommended, with Amazon Nova Micro offering an excellent cost-performance balance. For text-to-SQL, Amazon Nova generally outperform Claude variants, with minimal reflection recommended. For sentiment analysis, most models benefit from reflection, though gains may not justify increased costs. For translation, Claude generally benefits from reflection while Amazon Nova performs better without it. The case study on real-world marketing content localization data confirms that self-reflection gains with Claude models may be higher on the tasks that are more challenging.

In future work, we aim to conduct a deeper interpretative analysis of the budget tuning methods, including providing transition analysis of the generated thinking tokens. We also wish to expand our analysis outside of the Amazon Nova and Anthropic Claude model families to understand the influence of inference-time compute techniques on other leading model providers. We would also want to understand the benefits of combining complimentary techniques from inference-time compute such as parallel sampling, best-of-N majority voting and others. Another promising direction involves exploring reasoning paradigms that reduce overall token usage while preserving accuracy, such as conversation summarisation techniques or encouraging compressed responses, which may also help eliminate redundancy.

Finally, given the diversity of results we have observed across different tasks and domains, we would be interested in exploring approaches to automatically select an optimal inference configuration for a given pairing of model and prompt. The patterns that we have presented across model families show that there is a rich space of optimal configurations per model.



(a) Latency (seconds) Trade-off for Prompt Caching and Self-reflection



(b) Cost Trade-off for Prompt Caching and Self-reflection. The percentage difference is calculated as the difference in mean cost across repeats.

Figure 6: Prompt Caching cost ($) and latency trad-eoff results for a sampled Text-to-SQL prompt, repeated over 3 distinct rounds of generation with the mean and variance shown as $\mu \pm \sigma$

References

[1] Ekin Akyürek, Mehul Damani, Adam Zweiger, Linlu Qiu, Han Guo, Jyothish Pari, Yoon Kim, and Jacob Andreas. 2025. The Surprising Effectiveness of Test-Time Training for Few-Shot Learning. arXiv:2411.07279 [cs.AI] https://arxiv.org/abs/2411.07279

[2] Anthropic. 2024. The Claude 3 Model Family: Opus, Sonnet, Haiku. (2024).

[3] Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Michal Podstawski, Lukas Gianinazzi, Joanna Gajda, Tomasz Lehmann, Hubert Niewiadomski, Piotr Nyczyk, and Torsten Hoefler. 2024. Graph of Thoughts: Solving Elaborate Problems with Large Language Models. *Proceedings of the AAAI Conference on Artificial Intelligence* 38, 16 (March 2024), 17682–17690. doi:10.1609/aaai.v38i16.29720

[4] Xinyun Chen, Maxwell Lin, Nathanael Schärli, and Denny Zhou. 2024. Teaching Large Language Models to Self-Debug. In *The Twelfth International Conference on Learning Representations*. https://openreview.net/forum?id=KuPixIqPiq

[5] Wei-Lin Chiang, Lianmin Zheng, Ying Sheng, Anastasios Nikolas Angelopoulos, Tianle Li, Dacheng Li, Hao Zhang, Banghua Zhu, Michael Jordan, Joseph E. Gonzalez, and Ion Stoica. 2024. Chatbot Arena: An Open Platform for Evaluating LLMs by Human Preference. arXiv:2403.04132 [cs.AI] https://arxiv.org/abs/2403.04132

[6] Marta R Costa-Jussà, James Cross, Onur Çelebi, Maha Elbayad, Kenneth Heafield, Kevin Heffernan, Elahe Kalbassi, Janice Lam, Daniel Licht, Jean Maillard, et al. 2022. No language left behind: Scaling human-centered machine translation. *arXiv preprint arXiv:2207.04672* (2022).

[7] In Gim, Guojun Chen, Seung seob Lee, Nikhil Sarda, Anurag Khandelwal, and Lin Zhong. 2024. Prompt Cache: Modular Attention Reuse for Low-Latency Inference. arXiv:2311.04934 [cs.CL] https://arxiv.org/abs/2311.04934

[8] Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom Hennigan, Eric Noland, Katie Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karen Simonyan, Erich Elsen, Jack W. Rae, Oriol Vinyals, and Laurent Sifre. 2022. Training Compute-Optimal Large Language Models. arXiv:2203.15556 [cs.CL] https://arxiv.org/abs/2203.15556

[9] Jonas Hübotter, Sascha Bongni, Ido Hakimi, and Andreas Krause. 2025. Efficiently Learning at Test-Time: Active Fine-Tuning of LLMs. In *The Thirteenth International Conference on Learning Representations*. https://openreview.net/forum?id=NS1G1Uhny3

[10] Amazon Artificial General Intelligence. 2024. The Amazon Nova family of models: Technical report and model card. *Amazon Technical Reports* (2024). https://www.amazon.science/publications/the-amazon-nova-family-of-models-technical-report-and-model-card

[11] Alon Lavie and Abhaya Agarwal. 2007. Meteor: an automatic metric for MT evaluation with high levels of correlation with human judgments. In *Proceedings of the Second Workshop on Statistical Machine Translation* (Prague, Czech Republic) *(StatMT '07)*. Association for Computational Linguistics, USA, 228–231.

[12] Harrison Lee, Samrat Phatale, Hassan Mansoor, Kellie Ren Lu, Thomas Mesnard, Johan Ferret, Colton Bishop, Ethan Hall, Victor Carbune, and Abhinav Rastogi. 2024. RLAIF: Scaling Reinforcement Learning from Human Feedback with AI Feedback. https://openreview.net/forum?id=AAxIs3D2ZZ

[13] Percy Liang, Rishi Bommasani, Tony Lee, Dimitris Tsipras, Dilara Soylu, Michihiro Yasunaga, Yian Zhang, Deepak Narayanan, Yuhuai Wu, Ananya Kumar, Benjamin Newman, Binhang Yuan, Bobby Yan, Ce Zhang, Christian Cosgrove, Christopher D. Manning, Christopher Ré, Diana Acosta-Navas, Drew A. Hudson, Eric Zelikman, Esin Durmus, Faisal Ladhak, Frieda Rong, Hongyu Ren, Huaxiu Yao, Jue Wang, Keshav Santhanam, Laurel Orr, Lucia Zheng, Mert Yuksekgonul, Mirac Suzgun, Nathan Kim, Neel Guha, Niladri Chatterji, Omar Khattab, Peter Henderson, Qian Huang, Ryan Chi, Sang Michael Xie, Shibani Santurkar, Surya Ganguli, Tatsunori Hashimoto, Thomas Icard, Tianyi Zhang, Vishrav Chaudhary, William Wang, Xuechen Li, Yifan Mai, Yuhui Zhang, and Yuta Koreeda. 2023. Holistic Evaluation of Language Models. arXiv:2211.09110 [cs.CL] https://arxiv.org/abs/2211.09110

[14] 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:2305.20050 [cs.LG] https://arxiv.org/abs/2305.20050

[15] Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2024. Let's Verify Step by Step. In *The Twelfth International Conference on Learning Representations*. https://openreview.net/forum?id=v8L0pN6EOi

[16] Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. 2011. Learning Word Vectors for Sentiment Analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, Portland, Oregon, USA, 142–150. http://www.aclweb.org/anthology/P11-1015

[17] Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. 2023. Self-Refine: Iterative Refinement with Self-Feedback. In *Thirty-seventh Conference on Neural Information Processing Systems*. https://openreview.net/forum?id=S37hOerQLB

[18] Aaron Meurer, Christopher P. Smith, Mateusz Paprocki, Ondřej Čertík, Sergey B. Kirpichev, Matthew Rocklin, AMiT Kumar, Sergiu Ivanov, Jason K. Moore, Sartaj Singh, Thilina Rathnayake, Sean Vig, Brian E. Granger, Richard P. Muller, Francesco Bonazzi, Harsh Gupta, Shivam Vats, Fredrik Johansson, Fabian Pedregosa, Matthew J. Curry, Andy R. Terrel, Štěpán Roučka, Ashutosh Saboo, Isuru Fernando, Sumith Kulal, Robert Cimrman, and Anthony Scopatz. 2017. SymPy: symbolic computing in Python. *PeerJ Computer Science* 3 (Jan. 2017), e103. doi:10.7717/peerj-cs.103

[19] Amrith Setlur, Chirag Nagpal, Adam Fisch, Xinyang Geng, Jacob Eisenstein, Rishabh Agarwal, Alekh Agarwal, Jonathan Berant, and Aviral Kumar. 2025. Rewarding Progress: Scaling Automated Process Verifiers for LLM Reasoning. In *The Thirteenth International Conference on Learning Representations*. https://openreview.net/forum?id=A6Y7AqlzLW

[20] Charlie Victor Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. 2025. Scaling LLM Test-Time Compute Optimally Can be More Effective than Scaling Parameters for Reasoning. In *The Thirteenth International Conference on Learning Representations*. https://openreview.net/forum?id=4FWAwZtd2n

[21] Luong Trung, Xinbo Zhang, Zhanming Jie, Peng Sun, Xiaoran Jin, and Hang Li. 2024. ReFT: Reasoning with Reinforced Fine-Tuning. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Lun-Wei Ku, Andre Martins, and Vivek Srikumar (Eds.). Association for Computational Linguistics, Bangkok, Thailand, 7601–7614. doi:10.18653/v1/2024.acl-long.410

[22] Zhiheng Xi, Wenxiang Chen, Boyang Hong, Senjie Jin, Rui Zheng, Wei He, Yiwen Ding, Shichun Liu, Xin Guo, Junzhe Wang, Honglin Guo, Wei Shen, Xiaoran Fan, Yuhao Zhou, Shihan Dou, Xiao Wang, Xinbo Zhang, Peng Sun, Tao Gui, Qi Zhang, and Xuanjing Huang. 2024. Training large language models for reasoning through reverse curriculum reinforcement learning. In *Proceedings of the 41st International Conference on Machine Learning* (Vienna, Austria) *(ICML'24)*. JMLR.org, Article 2217, 19 pages.

[23] Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik Narasimhan. 2023. Tree of Thoughts: Deliberate Problem Solving with Large Language Models. arXiv:2305.10601 [cs.CL] https://arxiv.org/abs/2305.10601

[24] Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, Zilin Zhang, and Dragomir Radev. 2018. Spider: A Large-Scale Human-Labeled Dataset for Complex and Cross-Domain Semantic Parsing and Text-to-SQL Task. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, Ellen Riloff, David Chiang, Julia Hockenmaier, and Jun'ichi Tsujii (Eds.). Association for Computational Linguistics, Brussels, Belgium, 3911–3921. doi:10.18653/v1/D18-1425

Appendix

A. Prompt Templates

This Appendix provides prompt templates used for each of the four predictions tasks considered in the paper. We also provide the prompt templates for the LLM-as-judge feedback mechanism and for the self-reflection iterations.

A.1. Prediction Tasks

Flores-200

Translate the following text into {language}. Please output only the translated text with no prefix or introduction and put in in <translation></translation> XML tags.
Text to be translated: {source}

Math500

What is the answer to the following math problem: {problem}. Make sure to always state your final answer in <answer> </answer> tags.

Spider

You are a data scientist sqlite expert. Your job is to take user questions and translate them into SQL queries. For reference, today's date is 16/04/2025.
{table_name_and_schema}
<instruction> Only fetch the relevant columns for example partition is not generally required. </instruction>
The user question is provided inside <question></question> XML tags. Aim to generate a valid sqlite query for the user question using the table above. Always provide your thinking in <reasoning></reasoning> tags and then output the SQL statement in <SQL></SQL> tags.
Here is the question:{question}

IMDB Reviews

Read the following movie review. Classify the review sentiment as either positive or negative. Do not add any other words. Please output only the sentiment in <sentiment></sentiment> XML tags. Review to be classified: {review}

A.2. Self-Reflections and Feedback Mechanisms

Self-Reflection

Please reiterate your answer by thinking step by step, making sure to state your answer at the end of the response.
{feedback_mechanism_output}
As a reminder, the original question is {first_user_message}

LLM-as-a-Judge Feedback

You are evaluating the accuracy of a response to a question. Review the following context containing both a question and answer.
For your evaluation:

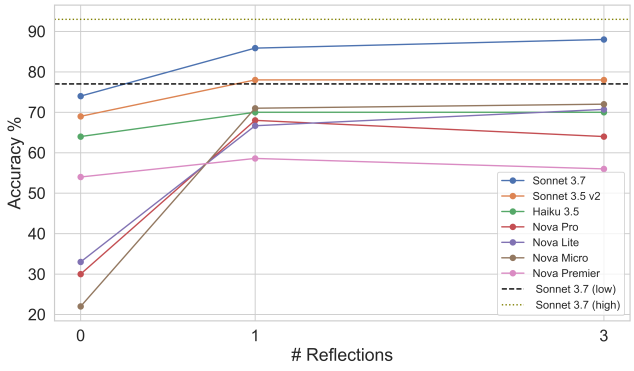
- Determine if the answer is factually correct and fully addresses the question
- Make a binary judgment: CORRECT or INCORRECT
- Provide a brief justification (1-2 sentences)
- If you don't have enough information to make a judgment, say so

User question: {user_query}
Provided response: {context}

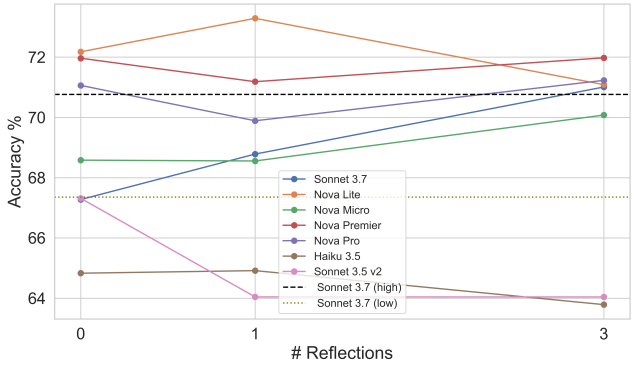
B. Extended Results

This Appendix provides extended empirical results, including the plots depicting the impact of number of self-reflection rounds on the LLM accuracy, as well as additional Sankey diagrams revealing the transition dynamics during the self-reflection rounds on Math500.

B.1. Impact of the Number of Reflections on Accuracy

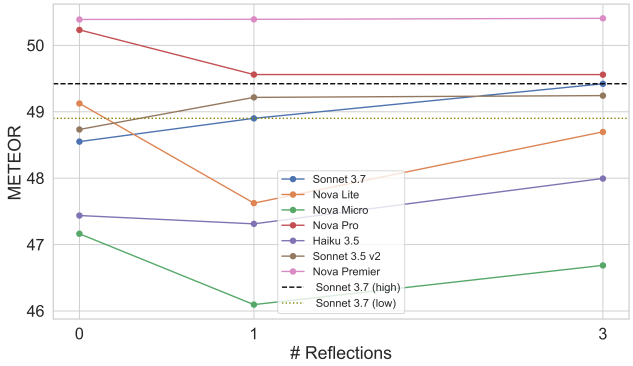


(a) Math500, Reflection Impact

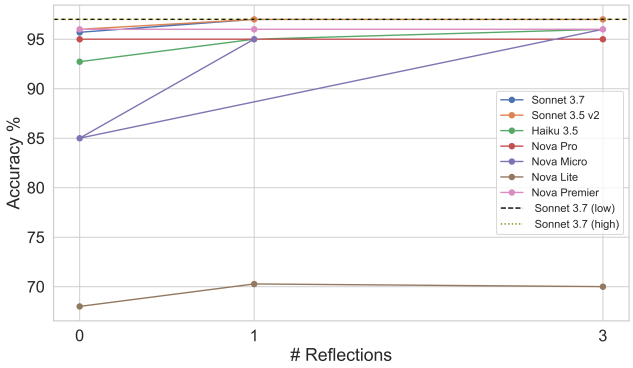


(b) Text-to-SQL (Spider), Reflection Impact

Figure 7: Number of Reflections (Math500 and Spider)



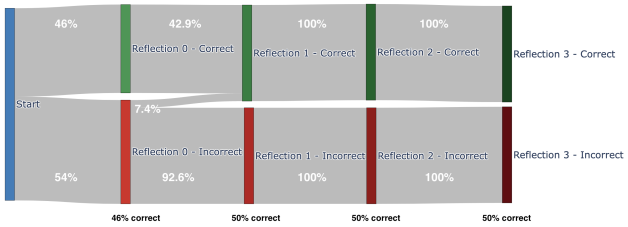
(a) Translation, Reflection Impact



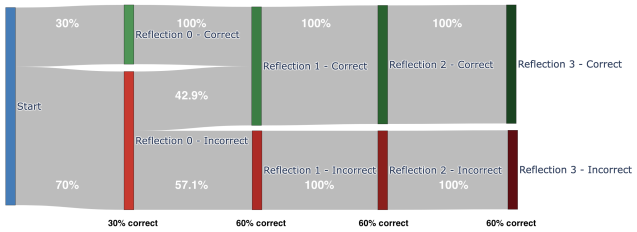
(b) Sentiment Classification, Reflection Impact

Figure 8: Number of Reflections (Flores-200 and IMDB)

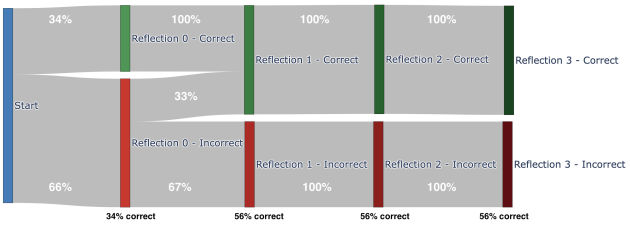
B2. Self-Reflection Transitions The Sankey diagrams (Figure 9 a-e) provide detailed visualisation of reflection pathways for additional models beyond Claude Sonnet 3.5 v2 and Amazon Nova Micro discussed in the main text. These diagrams reveal consistent patterns across model families while highlighting unique characteristics. Models with varying initial accuracy (34%-70%) all demonstrate perfect retention of correct answers through subsequent reflection stages; a pattern consistent across all tested LLMs. For models with moderate initial performance (46%-50%), we observe that the first reflection stage provides the most substantial correction opportunity, with 42.9%-67% of initially incorrect responses remaining incorrect after Reflection 0, while subsequent reflections yield minimal improvements. This mirrors Amazon Nova Micro's behavior described in the main text. In contrast, models with higher initial accuracy (70%) show more nuanced improvement patterns, with 13.3% of initially incorrect responses being corrected at Reflection 0 and accuracy stabilising at 74%—similar to Claude Sonnet 3.5's incremental improvement pattern. These findings reinforce our main conclusion that smaller models primarily benefit from initial self-correction, while more capable models can leverage both strong foundational performance and iterative improvement through extended reflection processes.



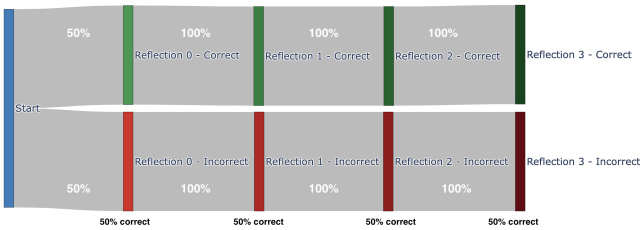
(a) Math500, Amazon Nova Premier Reflection Transitions



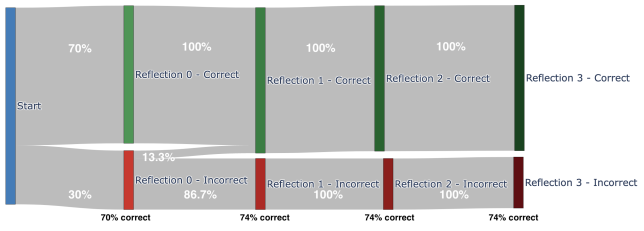
(b) Math500, Amazon Nova Pro Reflection Transitions



(c) Math500, Amazon Nova Lite Reflection Transitions



(d) Math500, Anthropic Claude 3.5 Haiku Reflection Transitions



(e) Math500, Anthropic Claude 3.7 Sonnet Reflection Transitions

Figure 9: Reflections Transitions (Math500)