Meta-Reflection: A Feedback-Free Reflection Learning Framework

Anonymous ACL submission

Abstract

Despite the remarkable capabilities of large language models (LLMs) in natural language understanding and reasoning, they often display undesirable behaviors, such as generating hallucinations and unfaithful reasoning. A prevalent strategy to mitigate these issues is the use of reflection, which refines responses through an iterative process. However, while promising, reflection heavily relies on high-quality external feedback and requires iterative multi-agent inference processes, thus hindering its practical application. In this paper, we propose *Meta-Reflection*, a novel feedback-free reflection mechanism that necessitates only a single inference pass without external feedback. Motivated by the human ability to remember and retrieve reflections from past experiences when encountering similar problems, Meta-Reflection integrates reflective insights into a codebook, allowing the historical insights to be stored, retrieved, and used to guide LLMs in problem-solving. To thoroughly investigate and evaluate the practicality of Meta-Reflection in real-world scenarios, we introduce an industrial e-commerce benchmark named E-commerce Customer Intent Detection (`ECID`). Extensive experiments conducted on both public datasets and the `ECID` benchmark highlight the effectiveness and efficiency of our proposed approach. Project is available at https://anonymous.4open.science/r/Meta-Reflection-62F5/

1 Introduction

Large Language Models (LLMs) (Achiam et al., 2023; Yang et al., 2024; Dubey et al., 2024) have demonstrated exceptional proficiency in diverse natural language processing tasks, *e.g.*, general language understanding (Wei et al., 2022a), generation (Pu and Demberg, 2023), and reasoning (Wei et al., 2022b; Yao et al., 2024). However, recent quantitative analyses revealed that contemporary frontier LLMs frequently exhibit undesirable and inconsistent behaviors, including unfaithful reasoning (Turpin et al., 2024) and the production of seemingly plausible yet inaccurate hallucinations (Rawte et al., 2023), especially when applying for intricate tasks. Such flawed outputs significantly undermine trust in LLMs and pose substantial obstacles to their widespread adoption in real-world applications.

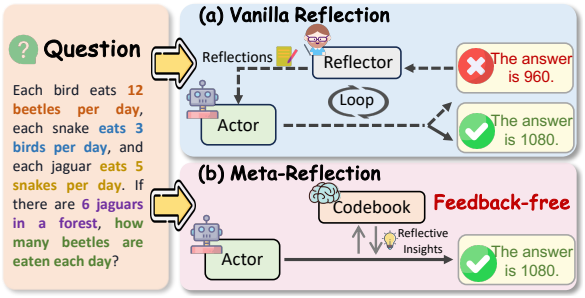


Figure 1: Illustration of different reflection mechanisms. **(a)** Vanilla reflection requires multi-agent inference and external feedback. **(b)** Meta-Reflection achieves feedback-free reflection in a single inference pass.

The undesirable phenomenon of LLMs is somewhat similar to human problem-solving, *i.e.*, we humans do not always generate the best answer on our first try in complex real-life scenarios. While dealing with complex problems, individuals has the capacity to actively refine their answers through a cycle of trial, inspection and correction (Pan et al., 2023). This capacity called *Reflection*, enables us to perform better than machines in high-level reasoning and would be the most precious capacity for modern AI. To simulate this ability, *LLMs' Reflection* (Madaan et al., 2024; Shinn et al., 2023) is devised to mitigate the flawed outputs of LLMs, which utilizes feedback from external sources (*e.g.*, the environment or other LLMs) to prompt the models to adapt their responses. This approach, as shown in Figure 1(a), enables the models to iteratively improve their performance by incorporating new information and adjusting their outputs based on external input, thereby enhancing their

accuracy and reliability over time. Upon reflection, however, contemporary approaches heavily rely on high-quality external feedback or ground-truth golden labels (Huang et al., 2024; Dou et al., 2024), which are often unavailable during inference scenarios. Besides, reflection typically requires iterative multi-agent inference processes (Du et al., 2023), which are resource-intensive. These aforementioned issues significantly constrain the practical deployment of LLMs in real-world scenarios.

In this paper, we propose *Meta-Reflection*, a novel reflection mechanism that operates without external feedback and requires only a single inference pass. Drawing inspiration from human cognitive processes (Kolodner, 1992), where individuals leverage past experiences and reflections to address similar questions without additional trials, we introduce a learnable meta-reflection codebook to store and retrieve reflective insights, as shown in Figure 1(b). During optimization, reflections are constructed using the vanilla reflection mechanism and integrated into the meta-reflection codebook. At inference, question-specific insights are retrieved from the codebook to guide the LLM in solving problems. This method enables LLMs to produce high-quality responses in a single pass, effectively mimicking how humans utilize prior experiences in analogous situations. Extensive experiments are conducted with open-source LLMs on diverse benchmarks, including programming, mathematical reasoning, and customer intent detection in E-commerce Intelligent Customer Service (ICS) for industry-specific scenarios. To evaluate our method in the ICS domain, we introduce E-commerce Customer Intent Detection (ECID), a new Chinese dataset designed to identify users' core intents, critical for enhancing service quality. Results across domains validate the efficiency and effectiveness of our approach. Key contributions of this work include:

- We propose Meta-Reflection, an innovative approach that achieves reflection in a single pass without iterative trials and feedback through well-designed codebook-based storage and retrieval mechanisms.
- We present a new dataset for E-commerce Customer Intent Detection (ECID) in the intelligent customer service domain, comprising 1,170 cases from real-world application.
- Extensive experiments across various domains and models demonstrate the effectiveness and robustness of our proposed method.

2 Related Work

2.1 Reflection for Large Language Models

Large language models (LLMs) (Achiam et al., 2023; Yang et al., 2024; Dubey et al., 2024), despite their exceptional performance, still exhibit undesired behaviors such as unfaithful reasoning (Turpin et al., 2024), hallucination (Rawte et al., 2023), and toxic generation (Zhang et al., 2024a). *Reflection* techniques (Pan et al., 2023; Shinn et al., 2023; Madaan et al., 2024) address these issues by utilizing feedback to guide LLMs in refining their outputs. For instance, Self-Refine (Madaan et al., 2024) uses a single LLM to generate, critique, and refine outputs, while Reflexion (Shinn et al., 2023) employs memory mechanisms and LLM agents to reflect on generations and feedback. Renze and Guven (2024) demonstrated the effectiveness of various reflection types across different domains. Nevertheless, reflection techniques often require high-quality external feedback or golden labels, typically unavailable during deployment (Huang et al., 2024; Dou et al., 2024), and frequently involve multi-agent inference processes, incurring significant computational costs. While Dou et al. (2024) incorporates reflective information through self-training, its implicit incorporation leads to sub-optimal results. In this work, we propose Meta-Reflection, which incorporates reflective information into a learnable codebook, enhancing performance across various tasks.

2.2 Parameter-Efficient Fine-Tuning (PEFT)

Parameter-Efficient Fine-Tuning (PEFT) methods enable adaptation of large pretrained models to downstream applications while avoiding the computational costs of full parameter fine-tuning (Hu et al., 2023). These methods can be broadly categorized into two primary approaches: *adapter-based* and *prompt-based* methods. Adapter-based methods introduce additional trainable parameters to a frozen pretrained model, with notable implementations including LoRA (Hu et al., 2021) and Llama-Adapter (Zhang et al., 2023). Prompt-based methods transform the discrete optimization of identifying optimal hard prompts into a continuous optimization problem using soft prompts, exemplified by Prefix-Tuning (Li and Liang, 2021), Prompt-Tuning (Lester et al., 2021), and P-Tuning (Liu

et al., 2022). In this work, we propose a lightweight learnable codebook module capable of storing and retrieving question-specific reflections, thereby enhancing LLM performance across diverse tasks.

3 Method

In this section, we first present the process of LLM-based reflection generation in Section 3.1. Next, we describe our proposed implicit feedback-free reflection approach in Section 3.2. Subsequently, we introduce the concept of adaptive meta-reflection alignment in Section 3.3. Finally, the overall optimization stage and inference stage are outlined in Section 3.4. The pipeline of Meta-Reflection is illustrated in Figure 2.

3.1 LLM-based Reflection Generation

Formally, consider a dataset $U = \{(x, y)\}_{i=1}^{N}$, where x represents a question and y represents its corresponding answer. An actor LLM agent $\mathcal{M}$ is used to generate an initial output $\hat{y}_{\text{act}} = \mathcal{M}(x)$. However, this process may lead to unfaithful reasoning or hallucination (Pan et al., 2023). To address these issues, reflection methods (Shinn et al., 2023; Madaan et al., 2024) propose leveraging feedback from external environment or golden labels (Huang et al., 2024) to refine the initial output $\hat{y}_{\text{act}}$. This feedback, denoted as $e = \mathcal{E}(x, \hat{y}_{\text{act}})$ where $\mathcal{E}$ represents the environment, provides comprehensive assessment of the initial output. For instance, in programming tasks, feedback typically includes interpreter information or execution results, while for mathematical problems, it involves comparing outputs against correct answer y. Based on the feedback e, a reflector LLM agent $\mathcal{R}$ generates reflections $r = \mathcal{R}(x, e)$, which guide the actor model $\mathcal{M}$ to produce refined responses $\hat{y}_{\text{ref}} = \mathcal{M}(x, r)$. As shown in Figure 2(a), this iterative process of generation, reflection, and refinement aims to enhance the quality and accuracy of the actor model $\mathcal{M}$'s outputs, mitigating potential errors and improving overall performance (Pan et al., 2023). Throughout the reflection generation process, we systematically curate a new dataset $D_t = \{(x, r, \hat{y}_{\text{ref}})\}_{i=1}^{N'}$ containing reflection-question-answer triplets. Details and corresponding prompts are provided in the Appendix.

3.2 Implicit Feedback-free Reflection

As discussed in Section 1, reflection methodologies, while promising, are limited by their reliance on external feedback (Huang et al., 2024) and computationally intensive multi-agent inference processes, hindering practical deployment. Inspired by the adage "One never falls into the same ditch twice," which suggests that people learn from past mistakes without repeated feedback, we propose implicit feedback-free reflection. As shown in Figure 2(b), this approach uses a learnable meta-reflection codebook to store and retrieve reflective insights, enabling efficient, feedback-free inference.

Meta-Reflection Codebook. The meta-reflection codebook consists of implicit reflective units $\boldsymbol{P} \in \mathbb{R}^{K \times C}$, where K and C denote codebook length and feature dimension, respectively. The question x serves as the query to retrieve the relevant reflective units from the codebook. Previous studies have demonstrated that intermediate layer features can provide sufficient preliminary understanding of input samples (Xin et al., 2020; Zhang et al., 2024b). Leveraging this insight, we utilize query representations from intermediate LLM layers, which contain rich semantic information for effective retrieval. Specifically, we position the meta-reflection codebook at the L-th layer ($0 < L < N$), where N is the total layers of LLM, serving as a repository of reflective insights. To retrieve relevant reflective insights, the query is processed through the initial L layers, transforming it into hidden states $\boldsymbol{H}_{\text{query}}^{L}$. We subsequently employ mean pooling $\mathcal{P}_{\text{mean}}$ to derive sentence-level representation as follows:

$$\boldsymbol{h} = \mathcal{P}_{\text{mean}}(\boldsymbol{H}_{\text{query}}^{L}) \in \mathbb{R}^{1 \times C} \tag{1}$$

The representation of the query is utilized to compute relevance score through:

$$\mathbf{s} = \sigma\left(\frac{g(\boldsymbol{h}) f(\boldsymbol{P}^{T})}{\sqrt{K}}\right) \in \mathbb{R}^{1 \times K}, \tag{2}$$

where σ denotes the softmax function, and $g(\cdot)$ and $f(\cdot)$ represent transformation functions implemented as two-layer MLPs, which serve to stabilize the training process (Liu et al., 2022). The resulting score $\mathbf{s}$ quantifies the relevance between the question and reflective units from codebook, with higher scores indicating more applicable reflective units for the given query. Based on the score $\mathbf{s}$, we select the top-k relevant reflection units from the codebook to form the sequence $\hat{\boldsymbol{P}}_{\text{ref}} \in \mathbb{R}^{k \times C}$, maintaining their relative positions in the codebook. The concatenated sequence $\{\boldsymbol{H}_{\text{query}}^{L}; \hat{\boldsymbol{P}}_{\text{ref}}\}$ is fed into the remaining $(N - L)$ layers, incorporating

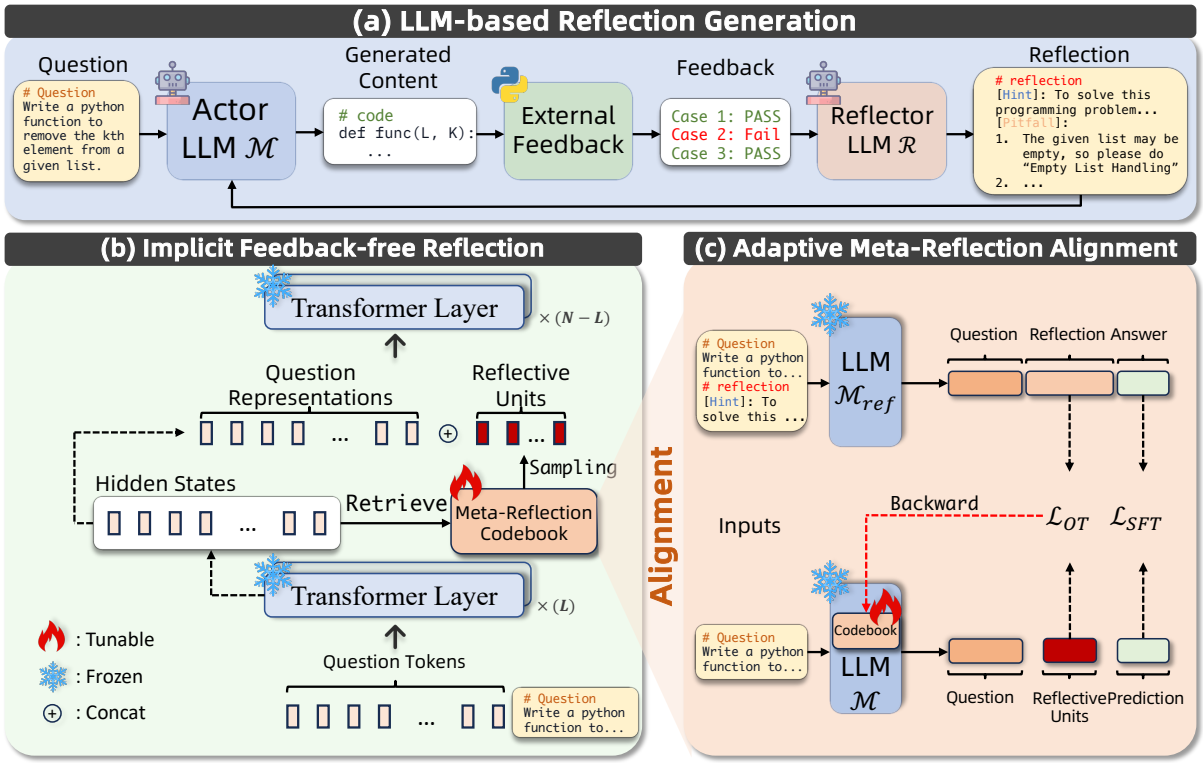


Figure 2: Overview of the Meta-Reflection framework: **(a)** LLM-based reflection generation through iterative processes; **(b)** Implicit feedback-free reflection, storing and retrieving reflective insights in a codebook; **(c)** Adaptive Meta-Reflection Alignment, incorporating reflective insights into the codebook.

question-specific reflective insights that guide the LLM's solution approach and enhance its performance. Notably, during the training phase, only the meta-reflection codebook is tunable while the backbone model remains frozen.

Sampling Strategy. To address the non-differentiable top-k function that impedes gradient back-propagation during training, and to enhance the sampling diversity, we employ Gumbel-Softmax technique (Jang et al., 2017) with additional tricks (Bengio et al., 2013) to derive the sampling process:

$$\hat{\mathbf{s}} = \sigma(\log(\mathbf{s}) + \epsilon_{\text{gumbel}}) \in \mathbb{R}^{1\times K},$$
$$I = \mathbb{1}_{i\in topk(\hat{s})} - sg[\hat{\mathbf{s}}] + \hat{\mathbf{s}} \in \mathbb{R}^{1\times K}, \quad (3)$$

where $\epsilon_{\text{gumbel}} \in \mathbb{R}^{1\times K}$ represents the Gumbel noise, $sg[\cdot]$ denotes the stop gradient operator and $\mathbb{1}_{i\in topk(\hat{s})}$ indicates whether an index belongs to the top-k indices. The resulting indicator vector I identifies the selected reflective units. This strategy ensures both differentiability during training and diverse sampling of reflective units.

3.3 Adaptive Meta-Reflection Alignment

After acquiring the dataset D_t as outlined in Section 3.1, our objective is to effectively leverage the information encapsulated within reflection r. As depicted in Figure 2(c), we employ a same frozen LLM but with different input as the teacher model $\mathcal{M}_{\text{ref}}$, to process the input sequence $\{x, r\}$ and extract the hidden states for each layer, $\{\boldsymbol{P}^l_{\text{que}}, \boldsymbol{P}^l_{\text{ref}}\}_{l=1}^{N}$, where $\boldsymbol{P}^l_{\text{que}}$ and $\boldsymbol{P}^l_{\text{ref}}$ denote the hidden states of query and reflection sequences, respectively. The reflective units selected from the codebook are integrated into the final $N-L$ layers, yielding $\{\hat{\boldsymbol{P}}^l_{\text{ref}}\}_{l=L}^{N}$, with the purpose of aligning $\{\boldsymbol{P}^l_{\text{ref}}\}_{l=L}^{N}$ and thereby embedding valuable information into the meta-reflection codebook. However, the dimensional variations and semantic misalignment between the ground-truth reflection $\boldsymbol{P}^l_{\text{ref}}$ and the reflective units $\hat{\boldsymbol{P}}^l_{\text{ref}}$ pose challenges for precise alignment between these sequences. To overcome this, we employ the optimal transport (OT) algorithm (Rubner et al., 2000; Liu et al., 2020; Zhang et al., 2020), which applies the earth mover's distance (EMD) to gauge the semantic discrepancy between these two sequences.

OT for Meta-Reflection Alignment. The EMD quantifies the distance between two discrete distributions as the minimum cost of transporting piles of dirt from "suppliers" to "demanders" (Zhu

et al., 2022), framed as a linear optimization problem. Specifically, at the l-th $(L < l < N)$ layer, we measure the distance required to transform $\hat{\boldsymbol{P}}^l_{\text{ref}} \in \mathbb{R}^{k' \times C}$ to $\boldsymbol{P}^l_{\text{ref}} \in \mathbb{R}^{k \times C}$. Let each unit $\hat{\boldsymbol{p}}_i \in \hat{\boldsymbol{P}}^l_{\text{ref}}$ possesses a total of $\boldsymbol{r}_i$ quantities to transport, and each unit $\boldsymbol{p}_j \in \boldsymbol{P}^l_{\text{ref}}$ requires $\boldsymbol{c}_j$ quantities, forming the transport prototype:

$$\Pi(\boldsymbol{r}, \boldsymbol{c}) = \{\boldsymbol{\Gamma} \in \mathbb{R}^{k' \times k} | \boldsymbol{\Gamma}\mathbf{1}_k = \boldsymbol{r}, \boldsymbol{\Gamma}^T\mathbf{1}_{k'} = \boldsymbol{c}\}, \tag{4}$$

where $\boldsymbol{r} \in \mathbb{R}^{k'}$ and $\boldsymbol{c} \in \mathbb{R}^k$ are marginal weights for transportation matrix $\boldsymbol{\Gamma}$ respectively. $\mathbf{1}$ is all-one vector with corresponding size, and $\Pi(\boldsymbol{r}, \boldsymbol{c})$ is the set of all possible distributions whose marginal weights are $\boldsymbol{r}$ and $\boldsymbol{c}$.

We define the cost per unit transported from supplier token $\hat{p}_i$ to demander token p_j as:

$$\mathbf{D}_{ij} = 1 - \frac{\hat{\boldsymbol{p}}_i^T \boldsymbol{p}_j}{||\hat{\boldsymbol{p}}_i|| ||\boldsymbol{p}_j||}, \tag{5}$$

where tokens with similar representations incur lower transport costs. Given this, we can define the linear optimization problem as follows:

$$\mathcal{R}_{\text{OT}}(\boldsymbol{r}, \boldsymbol{c}) = \min_{\boldsymbol{\Gamma} \in \boldsymbol{\Pi}(\boldsymbol{r}, \boldsymbol{c})} \sum_{i}^{k'} \sum_{j}^{k} \mathbf{D}_{ij} \boldsymbol{\Gamma}_{ij} \tag{6}$$

However, The exact minimization over $\boldsymbol{\Gamma}$ is solved in polynomial time and can be computationally intractable (Arjovsky et al., 2017; Genevay et al., 2018). Therefore, to find the optimal $\tilde{\boldsymbol{\Gamma}}$, we utilize *Sinkhorn Algorithm* (Cuturi, 2013) as an efficient approximation method. The detailed algorithm and the optimization process are shown in Appendix A. With optimal transportation matrix $\tilde{\boldsymbol{\Gamma}}$, the corresponding alignment loss for layer l is:

$$\mathcal{L}^l_{\text{OT}} = \langle \tilde{\boldsymbol{\Gamma}}, \mathbf{D} \rangle_{\text{F}}, \tag{7}$$

and the overall alignment loss is calculated as the mean across the last $N - L$ layers:

$$\mathcal{L}_{\text{OT}} = \frac{\sum_{l=L}^{N} \mathcal{L}^l_{\text{OT}}}{N - L} \tag{8}$$

The alignment loss quantifies the semantic gap (Li et al., 2020) between the reflective units from the meta-reflection codebook and the ground-truth reflection. In our scenario, by minimizing the $\mathcal{L}_{\text{OT}}$, the reflective insights from ground-truth reflection are incorporated into the codebook, enhancing the model $\mathcal{M}$'s capacity to handle complex tasks and improve overall performance.

3.4 Optimization and Inference

We delineate the overall optimization and inference stages as follows:

Progressive Optimization Stage. We employ a progressive optimization paradigm to enhance model performance. Initially, we utilize $\mathcal{L}_{\text{OT}}$ to align the reflective units from codebook with ground truth reflections, infusing reflective information into the codebook of the model $\mathcal{M}$. Subsequently, we leverage labels from dataset $\mathcal{D}_t$ to fine-tune the codebook using the vanilla supervised learning loss $\mathcal{L}_{\text{SFT}}$. This optimization paradigm ensures stable training progression and effective incorporation of reflective information, enhancing the model's ability to capture and utilize this knowledge while maintaining overall learning stability.

Inference Stage. During the inference stage, the input question x serves as query to retrieve pertinent reflective units from the meta-reflection codebook, guiding the LLM in addressing complex tasks. The retrieval process, elucidated in Section 3.2, is executed only once at the generation of the initial token. Leveraging the characteristics of causal language models, this inference stage can also utilize KV caching (Pope et al., 2023) to mitigate computational overhead.

4 Experiments

4.1 Datasets

We assess our method on diverse datasets across different domains: programming (*i.e.*, MBPP, HumanEval), mathematical reasoning (*i.e.*, GSM8K), and E-commerce customer intent detection (*i.e.*, ECID). Details can be found in Appendix B and C.

Programming. We evaluate our approach on two Python code generation benchmarks (MBPP (Austin et al., 2021) and HumanEval (Chen et al., 2021)), using Pass@k metric to measure the percentage of problems that successfully pass all unit tests within k attempts (Dou et al., 2024).

Mathematical Reasoning. For mathematical reasoning task, We employ the Grade School Math 8K (GSM8K) dataset (Cobbe et al., 2021) for evaluating Meta-Reflection. We utilize the Exact Match (EM) metric between the generated response and the correct answer (Madaan et al., 2024).

E-commerce Customer Intent Detection (ECID). Intelligent Customer Service (ICS) in e-commerce

Methods	*ref*	MBPP				HumanEval			
		LLaMA-3.1		CodeLlama		LLaMA-3.1		CodeLlama	
		Pass @ 1	Pass @ 3	Pass @ 1	Pass @ 3	Pass @ 1	Pass @ 3	Pass @ 1	Pass @ 3
Zero-Shot	✗	58.8	68.0	40.4	49.2	62.7	68.3	41.0	47.8
Few-Shot	✗	59.6	68.6	41.4	50.6	63.4	70.8	42.2	48.5
LoRA	✗	60.4	69.0	41.6	54.2	62.1	72.1	43.5	52.8
P-Tuning	✗	59.4	68.8	42.8	55.6	62.1	73.3	42.9	52.2
Llama-Adapter	✗	59.6	68.2	45.4	56.0	62.7	73.3	42.9	53.4
Re-ReST	✓	60.2	69.6	42.4	55.2	63.4	73.9	42.2	53.4
Reflection$_{(RAG)}$	✓	58.6	67.2	41.2	51.2	62.7	67.1	35.4	46.6
Ours	✓	**63.4**	**70.4**	**46.8**	**57.6**	**64.6**	**75.2**	**45.3**	**55.9**

Table 1: The experimental results on two programming benchmarks: MBPP (Austin et al., 2021) and HumanEval (Chen et al., 2021) datasets. We report the performance using Pass@1 and Pass@3 metrics. Here, *ref* indicates the utilization of reflection mechanism. The **boldface** and underline fonts denote the best and second-best performance, respectively.

Methods	*ref*	LLaMA-3.1	Qwen-2
Zero-Shot	✗	78.4	78.1
Few-Shot	✗	80.4	79.5
LoRA	✗	80.7	80.0
P-Tuning	✗	79.4	79.6
Re-ReST	✓	82.4	84.8
Reflection$_{(RAG)}$	✓	77.7	76.7
Ours	✓	**85.3**	**86.7**

Table 2: The experimental results on a mathematical reasoning benchmark: GSM8K (Cobbe et al., 2021).

Methods	*ref*	LLaMA-3.1	Qwen-2
Zero-Shot	✗	83.5	89.8
Few-Shot	✗	85.5	90.8
LoRA	✗	86.9	91.1
P-Tuning	✗	85.5	90.9
Re-ReST	✓	85.5	90.9
Reflection$_{(RAG)}$	✓	81.8	86.6
Ours	✓	**89.7**	**92.9**

Table 3: The experimental results on ECID dataset in E-commerce domain.

has emerged as a prominent application for Large Language Models (LLMs) (Kolasani, 2023). Research by Cheng et al. (2024) has highlighted that the key to enhancing ICS performance lies in accurately inferring customers' core service intent through the analysis of historical customer-agent interactions and corresponding order data. However, current LLMs struggle with precise intent detection, primarily due to semantic ambiguities inherent in diverse service requests. To evaluate the efficacy of our proposed approach in this domain, we introduce the E-commerce Customer Intent Detection (ECID) dataset. This dataset comprises meticulously cleaned and systematically labeled Chinese language data from Taobao online customer service interactions, resulting in 1,170 high-quality entries. Details of the ECID can be found in Appendix B.

4.2 Experimental Setup

Models. We evaluate Meta-Reflection across various open-source LLMs. For the actor models, we utilize `Qwen-2-7B-Instruct` (Yang et al., 2024), `Llama-3.1-8B-Instruct` (Dubey et al., 2024), and `CodeLlama-7B-Instruct` (Roziere et al., 2023). `Qwen-2-72B-Instruct` serves as the reflector model.

Baselines. To evaluate the effectiveness of our proposed method, we compare it with three types of baselines: **Common reasoning**: including Zero-Shot and Few-Shot approaches. **PEFT methods**: Adapter-based approaches such as LoRA (Hu et al.) and Llama-Adapter (Zhang et al., 2023), as well as Prompt-based methods like P-Tuning (Liu et al., 2022). **Reflection-based methods**: Re-ReST (Dou et al., 2024) for reflection-enhanced training. Additionally, we implement Reflection-RAG, which generates reflections on training data and employs Retrieval-Augmented Generation (RAG) (Gao et al., 2023) during inference to select the most relevant question-specific reflections. The details of baselines are in the Appendix D.

4.3 Main Results

Tables 1, 2, and 3 present the experimental results across three distinct domains: programming, mathematical reasoning, and e-commerce customer in-

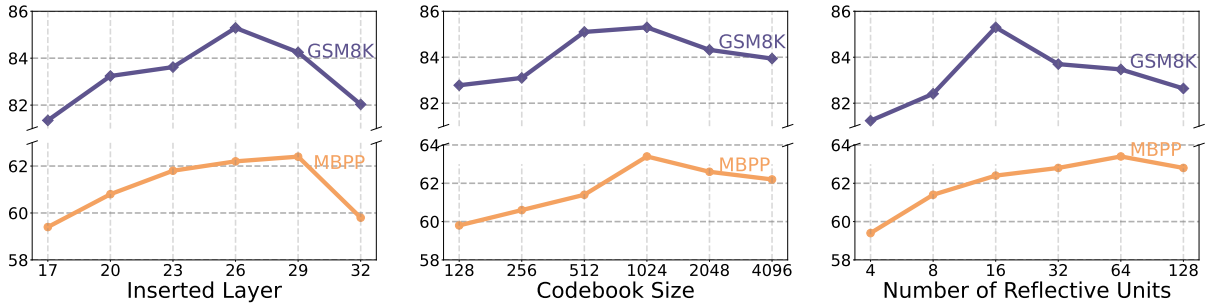


Figure 3: Sensitivity analysis of three critical hyper-parameters: **Left**: Insertion layer position of the codebook; **Middle**: Total number of reflective units in codebook; **Right**: Number of reflective units selected per inference.

tent detection.

Our empirical investigation reveals fundamental limitations in base LLMs' domain-specific capabilities, as demonstrated by CodeLlama's modest 40.4% performance on MBPP under the Pass@1 metric. This deficiency primarily stems from these models' **insufficient domain knowledge and capabilities**. While tuning with PEFT methods like LoRA demonstrate potential for improvement, the gains remain incremental—yielding mere 1.2% and 0.2% improvements in Zero-Shot and Few-Shot settings respectively. This suggests that current supervised learning paradigms, while domain knowledge internalization during finetuning, **fail to address the critical need for guidance during inference.**

Recent advances in reflection-based methodologies, particularly Re-ReST, have shown promise by implicitly incorporating reflective guidance through refined self-training data, evidenced by LLaMA-3.1's 1.7% performance improvement over LoRA on GSM8K. However, these approaches still **neglect the crucial aspect of explicit, granular guidance during the inference phase**. Although leveraging RAG-retrieved reflections as explicit guidance appears promising, empirical results on benchmarks like GSM8K and ECID demonstrate suboptimal performance even compared to common reasoning approaches. This degradation occurs because retrieved reflections, though relevant to source problems, often **lack precise applicability to similar cases and may introduce noise**, particularly in mathematical tasks requiring fine-grained guidance. Comprehensive case studies supporting these findings are presented in Appendix G. Our proposed methodology addresses these limitations by providing explicit, fine-grained reflective guidance during inference, significantly outperforming existing approaches across all baseline metrics.

Methods	**First Token Latency** (↓)		
	Retrieve	LLM Processing	Total
Zero-Shot	–	149 ms	149 ms
Few-Shot	–	545 ms	545 ms
Reflection$_{(\text{RAG})}$	642 ms	309 ms	951 ms
Ours	93 ms	153 ms	246 ms

Table 4: We analyze inference efficiency on the ECID dataset by measuring the first token (in milliseconds). The first token latency is decomposed into retrieval time and LLM processing time. All measurements are conducted using a 24-core Intel(R) Xeon(R) Platinum 8163 CPU @ 2.50GHz and 2 NVIDIA Tesla V100 GPUs.

4.4 Inference Efficiency Analysis

We evaluate the inference efficiency of Meta-Reflection, with results presented in Table 4. Compared to existing reflection-based methods like Reflection-RAG that require separate encoders and knowledge base retrieval, our approach leverages LLM's intermediate layer representations for retrieval. Furthermore, RAG-based methods store knowledge in a discrete format, necessitating a large-scale knowledge base. In contrast, Meta-Reflection captures knowledge and reflective insights in a dense format, enabling the construction of a smaller, more compact knowledge base, thereby significantly reducing computational overhead. Notably, our method achieves comparable first-token latency to common reasoning approaches while maintaining the benefits of reflection-based reasoning, demonstrating its practicality for real-world applications.

4.5 Sensitive Analysis

We perform sensitivity analysis on three critical hyper-parameters of Meta-Reflection: inserted layers, codebook size, and number of reflective units. The Experimental results are presented in Figure 3.

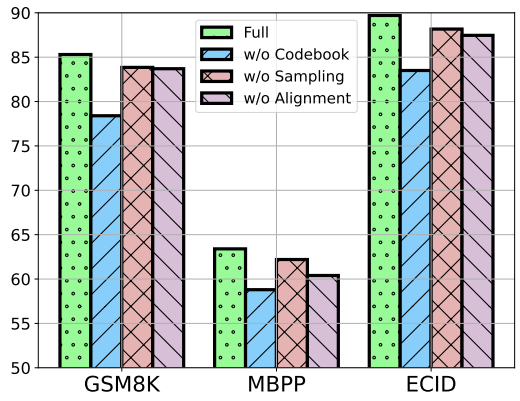


Figure 4: Experimental results of ablation study.

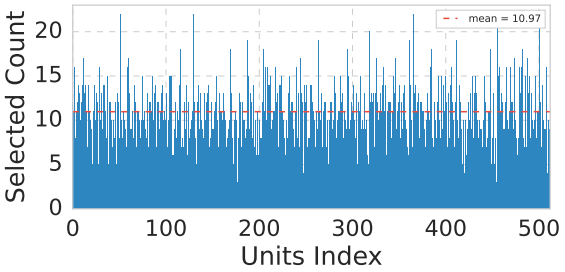


Figure 5: Visualization of reflective unit selection frequencies distribution in the ECID dataset. The **x-axis** represents the unit indices, while the **y-axis** shows their cumulative selection counts during inference. The codebook is configured with a size of 512 units, with 16 units selected per inference.

Inserted Layer. The positioning of the meta-reflection codebook layer critically influences the balance between retrieval quality and reflective information integration. Analysis from Figure 3 (Left) reveals that early-layer insertion results in insufficient semantic query encoding, while late-layer placement constrains the processing of retrieved reflective components. Our empirical results demonstrate that an intermediate-posterior position (*e.g.*, layer 26) achieves optimal performance.

Codebook Size. The codebook size, which represents the total number of reflective units, determines the capacity of the codebook. As shown in Figure 3 (Middle), we observe that a codebook size of 1024 yields optimal performance. Smaller sizes may lead to underfitting, while larger sizes can result in a sparse codebook, potentially causing training instability.

Number of Reflective Units. As illustrated in Figure 3 (Right), the optimal number of reflective units varies proportionally with task complexity. Notably, while base LLaMA-3.1 achieves a substantial 78.4% performance on GSM8K, the inherently more challenging MBPP dataset requires additional reflective insights to provide comprehensive guidance. This observation underscores the relationship between task complexity and the requisite quantity of reflective support.

4.6 Ablation Study

In this section, we conduct a comprehensive ablation study to evaluate the impact of various components in Meta-Reflection. We examine three key variants: 'w/o Codebook' (no meta-reflection codebook), 'w/o Sampling' (no sampling strategy defined in Equation 3), and 'w/o Alignment' (no alignment mechanism described in Equation 8). As illustrated in Figure 4, the meta-reflection codebook demonstrates significant effectiveness in storing and retrieving reflective units that guide LLMs through the problem-solving process. The ablation analysis further demonstrates that both the sampling strategy and meta-reflection alignment mechanism play crucial roles in maintaining solution diversity and incorporating reflective insights, respectively, thereby enhancing overall performance.

4.7 Visualization

We visualize the selection frequency distribution of reflective units within the meta-reflection codebook. As shown in Figure 5, the selection patterns of reflective units vary significantly. Notably, certain units exhibit higher selection frequencies, potentially reflecting commonly applicable insights, whereas others are selected less frequently, suggesting their specialized nature. Additional visualization results are provided in Appendix H.

5 Conclusion

In this paper, we introduce Meta-Reflection, a novel feedback-free reflection mechanism that operates with a single inference pass without requiring external feedback. Our approach incorporates reflective insights within a codebook structure, facilitating efficient storage, retrieval, and utilization of historical insights to guide LLMs in problem-solving tasks. To validate the practical applicability of our method, we propose a new industrial benchmark: E-commerce Customer Intent Detection (ECID). Comprehensive experiments conducted across diverse domains and the ECID benchmark demonstrate the effectiveness and efficiency of Meta-Reflection.

6 Limitations

This work introduces Meta-Reflection, a novel feedback-free reflection mechanism that operates with a single inference pass without requiring external feedback. However, Meta-Reflection is primarily applicable to parameter-accessible LLMs (*e.g.*, Qwen and LLaMA) and cannot be extended to models where parameters are inaccessible through API-only interfaces (*e.g.*, ChatGPT and Claude).

References

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

Martin Arjovsky, Soumith Chintala, and Léon Bottou. 2017. Wasserstein generative adversarial networks. In *International conference on machine learning*, pages 214–223. PMLR.

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, et al. 2021. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*.

Yoshua Bengio, Nicholas Léonard, and Aaron Courville. 2013. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS '20, Red Hook, NY, USA. Curran Associates Inc.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. 2021. Evaluating large language models trained on code. *Preprint*, arXiv:2107.03374.

Zhendong Cheng, Wenfang Fan, Bingjia Shao, Wenli Jia, and Yong Zhang. 2024. The impact of intelligent customer service agents' initial response on consumers' continuous interaction intention. *Journal of Retailing and Consumer Services*, 76:103585.

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

Marco Cuturi. 2013. Sinkhorn distances: Lightspeed computation of optimal transport. *Neural Information Processing Systems,Neural Information Processing Systems*.

Zi-Yi Dou, Cheng-Fu Yang, Xueqing Wu, Kai-Wei Chang, and Nanyun Peng. 2024. Reflection-reinforced self-training for language agents. *arXiv preprint arXiv:2406.01495*.

Yilun Du, Shuang Li, Antonio Torralba, Joshua B Tenenbaum, and Igor Mordatch. 2023. Improving factuality and reasoning in language models through multiagent debate. *arXiv preprint arXiv:2305.14325*.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, and Haofen Wang. 2023. Retrieval-augmented generation for large language models: A survey. *arXiv preprint arXiv:2312.10997*.

Aude Genevay, Gabriel Peyré, and Marco Cuturi. 2018. Learning generative models with sinkhorn divergences. In *International Conference on Artificial Intelligence and Statistics*, pages 1608–1617. PMLR.

Edward J Hu, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. Lora: Low-rank adaptation of large language models. In *International Conference on Learning Representations*.

Edward J Hu, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. 2021. Lora: Low-rank adaptation of large language models. In *International Conference on Learning Representations*.

Zhiqiang Hu, Lei Wang, Yihuai Lan, Wanyu Xu, Ee-Peng Lim, Lidong Bing, Xing Xu, Soujanya Poria, and Roy Lee. 2023. Llm-adapters: An adapter family for parameter-efficient fine-tuning of large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 5254–5276.

Jie Huang, Xinyun Chen, Swaroop Mishra, Huaixiu Steven Zheng, Adams Wei Yu, Xinying Song, and Denny Zhou. 2024. Large language models cannot self-correct reasoning yet. In *The Twelfth International Conference on Learning Representations*.

Eric Jang, Shixiang Gu, and Ben Poole. 2017. Categorical reparametrization with gumble-softmax. In *International Conference on Learning Representations (ICLR 2017)*. OpenReview. net.

Li Jing, Pascal Vincent, Yann LeCun, and Yuandong Tian. Understanding dimensional collapse in contrastive self-supervised learning. In *International Conference on Learning Representations*.

Saydulu Kolasani. 2023. Optimizing natural language processing, large language models (llms) for efficient customer service, and hyper-personalization to enable sustainable growth and revenue. *Transactions on Latest Trends in Artificial Intelligence*, 4(4).

Janet L Kolodner. 1992. An introduction to case-based reasoning. *Artificial intelligence review*, 6(1):3–34.

Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The power of scale for parameter-efficient prompt tuning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3045–3059.

Jianqiao Li, Chunyuan Li, Guoyin Wang, Hao Fu, Yuhchen Lin, Liqun Chen, Yizhe Zhang, Chenyang Tao, Ruiyi Zhang, Wenlin Wang, et al. 2020. Improving text generation with student-forcing optimal transport. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 9144–9156.

Xiang Lisa Li and Percy Liang. 2021. Prefix-tuning: Optimizing continuous prompts for generation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4582–4597.

Songtao Liu, Zeming Li, and Jian Sun. 2020. Self-emd: Self-supervised object detection without imagenet. *arXiv preprint arXiv:2011.13677*.

Xiao Liu, Kaixuan Ji, Yicheng Fu, Weng Tam, Zhengxiao Du, Zhilin Yang, and Jie Tang. 2022. P-tuning: Prompt tuning can be comparable to fine-tuning across scales and tasks. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 61–68.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, et al. 2024. Self-refine: Iterative refinement with self-feedback. *Advances in Neural Information Processing Systems*, 36.

Liangming Pan, Michael Saxon, Wenda Xu, Deepak Nathani, Xinyi Wang, and William Yang Wang. 2023. Automatically correcting large language models: Surveying the landscape of diverse self-correction strategies. *arXiv preprint arXiv:2308.03188*.

Reiner Pope, Sholto Douglas, Aakanksha Chowdhery, Jacob Devlin, James Bradbury, Jonathan Heek, Kefan Xiao, Shivani Agrawal, and Jeff Dean. 2023. Efficiently scaling transformer inference. *Proceedings of Machine Learning and Systems*, 5:606–624.

Dongqi Pu and Vera Demberg. 2023. Chatgpt vs human-authored text: Insights into controllable text summarization and sentence style transfer. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 4: Student Research Workshop)*, pages 1–18.

Vipula Rawte, Amit Sheth, and Amitava Das. 2023. A survey of hallucination in large foundation models. *arXiv preprint arXiv:2309.05922*.

Matthew Renze and Erhan Guven. 2024. Self-reflection in llm agents: Effects on problem-solving performance. *arXiv preprint arXiv:2405.06682*.

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

Yossi Rubner, Carlo Tomasi, and Leonidas J Guibas. 2000. The earth mover's distance as a metric for image retrieval. *International journal of computer vision*, 40:99–121.

Noah Shinn, Federico Cassano, Edward Berman, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2023. Reflexion: Language agents with verbal reinforcement learning. *Preprint*, arXiv:2303.11366.

Miles Turpin, Julian Michael, Ethan Perez, and Samuel Bowman. 2024. Language models don't always say what they think: unfaithful explanations in chain-of-thought prompting. *Advances in Neural Information Processing Systems*, 36.

Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, et al. 2022a. Emergent abilities of large language models. *arXiv preprint arXiv:2206.07682*.

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

Ji Xin, Raphael Tang, Jaejun Lee, Yaoliang Yu, and Jimmy Lin. 2020. DeeBERT: Dynamic early exiting for accelerating BERT inference. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2246–2251, Online. Association for Computational Linguistics.

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

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. 2024. Tree of thoughts: Deliberate problem solving with large language models. *Advances in Neural Information Processing Systems*, 36.

Chi Zhang, Yujun Cai, Guosheng Lin, and Chunhua Shen. 2020. Deepemd: Few-shot image classification with differentiable earth mover's distance and structured classifiers. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Jiang Zhang, Qiong Wu, Yiming Xu, Cheng Cao, Zheng Du, and Konstantinos Psounis. 2024a. Efficient toxic content detection by bootstrapping and distilling large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 21779–21787.

Renrui Zhang, Jiaming Han, Chris Liu, Peng Gao, Aojun Zhou, Xiangfei Hu, Shilin Yan, Pan Lu, Hongsheng Li, and Yu Qiao. 2023. Llama-adapter: Efficient fine-tuning of language models with zero-init attention. *arXiv preprint arXiv:2303.16199*.

Wenqiao Zhang, Tianwei Lin, Jiang Liu, Fangxun Shu, Haoyuan Li, Lei Zhang, He Wanggui, Hao Zhou, Zheqi Lv, Hao Jiang, et al. 2024b. Hyperllava: Dynamic visual and language expert tuning for multimodal large language models. *arXiv preprint arXiv:2403.13447*.

Yun Zhu, Jianhao Guo, Fei Wu, and Siliang Tang. 2022. Rosa: A robust self-aligned framework for node-node graph contrastive learning. *arXiv preprint arXiv:2204.13846*.

A Sinkhorn Algorithm and Optimal Transport

The vanilla optimization problem of optimal transport, as formulated in Equation 6, aims to find the optimal transportation matrix $\tilde{\mathbf{\Gamma}}$. Nevertheless, the exact minimization over $\tilde{\mathbf{\Gamma}}$ is generally computationally intractable (Arjovsky et al., 2017; Genevay et al., 2018; Li et al., 2020). To address this, the *Sinkhorn Algorithm* (Cuturi, 2013) is utilized to approximate $\tilde{\mathbf{\Gamma}}$. Specifically, the algorithm introduces a regularization term:

$$\min_{\mathbf{\Gamma}\in\mathbf{\Pi}(\mathbf{r},\mathbf{c})} \langle \mathbf{\Gamma}, \mathbf{D} \rangle_{\mathrm{F}} + \underbrace{\frac{1}{\lambda}\mathbf{\Gamma}(\log \mathbf{\Gamma} - 1)}_{\text{regularization term}}, \tag{9}$$

where $\langle , \rangle_{\mathrm{F}}$ denotes Frobenius inner product, and λ is a hyper-parameter that controls the strength of regularization.

With this regularization term, the optimal $\tilde{\Gamma}$ can be approximated as:

$$\tilde{\mathbf{\Gamma}} = diag(\boldsymbol{v})\mathbf{Q}diag(\boldsymbol{u}), \tag{10}$$

where $\mathbf{Q} = e^{-\lambda \mathbf{D}}$, and $\boldsymbol{v}$, $\boldsymbol{u}$ are two coefficient vectors whose values can be iteratively updated as:

$$\begin{aligned} \boldsymbol{v}_i^{t+1} &= \frac{\boldsymbol{r}_i}{\sum_{j=1}^{k} \mathbf{Q}_{ij}\boldsymbol{u}_j^t}, \\ \boldsymbol{u}_j^{t+1} &= \frac{\boldsymbol{c}_j}{\sum_{i=1}^{k'} \mathbf{Q}_{ij}\boldsymbol{v}_i^{t+1}} \end{aligned} \tag{11}$$

The critical aspect then lies in determining the marginal weights $\boldsymbol{r}$ and $\boldsymbol{c}$, which control the total supplying and demanding units, respectively. A larger weight indicates that the reflective unit exhibits semantic similarity to the ground truth reflection tokens. We define the weight as dot product between its embedding and the mean pooling embedding from the other set:

$$\begin{aligned} \boldsymbol{r}_i &= \max\{\boldsymbol{p'}_i^T \cdot \frac{\sum_{j=1}^{k} \boldsymbol{p}_j}{k}, 0\}, \\ \boldsymbol{c}_j &= \max\{\boldsymbol{p}_j^T \cdot \frac{\sum_{i=1}^{k'} \boldsymbol{p'}_i}{k'}, 0\} \end{aligned} \tag{12}$$

After obtaining the approximated optimal transportation matrix $\tilde{\Gamma}$, we can compute the loss as defined in Equation 7.

B E-commerce Customer Intent Detection (ECID) Benchmark

In the domain of Intelligent Customer Service (ICS) for e-commerce, effectively and efficiently discerning customers' core intentions when they contact ICS for assistance is critical to enhancing service quality (Cheng et al., 2024; Kolasani, 2023). In this work, we introduce an industrial benchmark, named E-commerce Customer Intent Detection (ECID) to evaluate our proposed method. This dataset is in Chinese, focusing on customer interactions within major Chinese e-commerce platforms. The following sections detail the construction of this dataset and elaborate on its specific tasks.

Task. The primary objective of the ECID dataset is to infer the core intention of customers seeking ICS assistance, based on previous communication records between customers and customer service platforms, customer purchase histories, and order information. The core intention refers to the customer's current concern or the problem they wish to resolve. Specifically, each data point in the dataset comprises input information from five fields:

- **Customer Question.** The specific issue or obstacle encountered by the customer.
- **Customer Request.** Customer requirements, encompassing all objectives or desired outcomes expressed during interactions with the ICS, sellers, and platform customer service representatives, as well as any proactively initiated request.
- **Solution.** Proposals offered by the platform or sellers to address the customer's issue.
- **Customer Attitude.** The customer's attitudes towards the proposed solutions, as expressed during communication.
- **Processing status.** PThe current state of the customer's submitted request.

ECID aims to match the aforementioned input information with the most appropriate intention from a predefined list. In real-world applications, we categorize intentions into 36 distinct types, each representing a specific issue customers seek to resolve. For the ECID dataset, a condensed list of six intentions is provided, from which the most relevant core intention must be selected. An illustrative example is presented in the accompanying Figure 6.

Data Processing. The ECID dataset is derived from customer service system records of the

Question:

1. 用户遇到的问题：1天前用户反馈收到的落地衣帽架质量太差，太薄。
2. 用户的诉求：1天前用户因质量问题申请退货退款，金额为*元，29分钟前用户询问退货运费承担者。
3. 平台或商家给出的解决方案：商家最初提出*元的补偿方案，随后建议用户调整置物架看看，并告知用户厂家品控不同。在用户坚持退货后，商家表示会加强品控并告知用户运费需要自行承担。
4. 用户对解决方案表达的态度：用户对商家最初*元补偿的解决方案表示不满，并明确表示不接受，坚持要退货。在商家告知需要自行承担运费后，用户表示强烈不满，并表示将把实际情况反馈在评价里，必要时请平台介入处理。
5. 处理状态：当前状态卖家已经同意退款，等待买家退货，运费卖家承担。1天前发货包裹已签收。

诉求清单列表：['A 退运费', 'B 争议处理规则', 'C 预约上门取件', 'D 退货款', 'E 填写退货快递单号', 'F 投诉物流服务问题']

Answer: A

Figure 6: An example of the ECID dataset.

Taobao e-commerce platform, collected over a single day. From this collection, we randomly sampled 30,000 data points in an unbiased manner. Each data point comprises information from various sources, including customer-service representative chat logs, customer-seller communications, customer order details, and ongoing request processing records. We employed a fine-tuned LLM, specifically `Qwen2-7B-Instruct`, to extract the aforementioned five fields of information from the diverse sources.

We initially applied a rule-based method to eliminate incomplete or inconsistent data (such as newly registered users without any purchase history), resulting in approximately 4,000 refined data points. Subsequently, we utilized `GPT-4-turbo-128k` and `Qwen2-72B-Instruct` for data labeling. Using a voting system, we selected the most appropriate intention from a predefined list of 36 intentions, along with five secondary matching intentions, to create a set of candidate intentions and answers for each data point. To ensure high data quality, we discarded instances where the highest voting rate was below 80%. We also implemented human evaluation, randomly sampling and verifying the accuracy of answers. This rigorous process yielded **1,170** high-quality data points, each accompanied by a Chain-of-Thought (CoT) reasoning process. The dataset was partitioned into a 7:3 ratio for training and testing.

We conducted data anonymization to remove sensitive information from the dataset. Personal identifiable information, including customer names, addresses, and contact details, was redacted. Additionally, all monetary values within the dataset were masked using asterisks (*) to ensure confidentiality.

C Public Datasets

We evaluate our method across three public datasets spanning diverse domains: two programming benchmarks (MBPP and HumanEval) and one mathematical reasoning dataset (GSM8K).

Programming. For evaluating our method on programming tasks, we utilize two Python code programming benchmarks: MBPP (Austin et al., 2021) and HumanEval (Chen et al., 2021). The MBPP dataset consists of approximately 1,000 Python programming problems, while HumanEval encompasses 161 problems, each accompanied by comprehensive unit test cases. We adhere to the official train-test split for MBPP, employing its training set for model training. As HumanEval provides only a test set, we use it exclusively for evaluation purposes. Following Dou et al. (2024), we employ the Pass@k metric, which quantifies the percentage of problems where the model successfully passes all unit tests within k attempts. During the code generation process, in line with previous work by Roziere et al. (2023), the actor model is provided with the unit test cases.

Mathematical Reasoning. For mathematical reasoning evaluation, we employ the Grade School Math 8K (GSM8K) dataset (Cobbe et al., 2021), a comprehensive benchmark containing approximately 8,000 grade school mathematics word problems. This dataset is particularly valuable due to its linguistic diversity and high-quality annotations, featuring detailed human-curated solution trajectories and precise answers for each problem (Madaan et al., 2024). Following standard practices, we strictly adhere to the official train-test split (7,473 for training, 1,319 for testing) in our experimental setup. Performance is evaluated using the Exact Match (EM) metric, which assesses the precise correspondence between model-generated responses and ground-truth answers (Madaan et al., 2024), providing a rigorous measure of mathematical rea-

soning capabilities.

D Baselines

We evaluate our method against three categories of baselines: **Common Reasoning**, **Parameter-Efficient Fine-Tuning (PEFT)**, and **Reflection-Based** approaches. The specifics of these baseline implementations are detailed below:

Common Reasoning Approaches. For common reasoning approaches, we evaluate both **Zero-Shot** and **Few-Shot** (2-shots) (Brown et al., 2020) strategies. In both settings, we employ the Chain-of-Thought (CoT) (Wei et al., 2022b) reasoning methodology to facilitate structured generation processes.

Parameter-Efficient Fine-Tuning (PEFT) Approaches. We implement three widely-adopted PEFT methods for model tuning: **LoRA** (Hu et al.), **P-Tuning** (Liu et al., 2022), and **Llama-Adapter** (Zhang et al., 2023). Through extensive hyper-parameter grid search: For LoRA, we augment the query, key, and value matrices with adapter matrices of rank {8, 16}. For P-Tuning, we experiment with prompt lengths of {16, 32, 64} and implement the MLP-based re-parameterization function (Liu et al., 2022). For Llama-Adapter, we explore adapter lengths of {32, 64} and position them within the final 15 layers of the LLM (Zhang et al., 2023).

Reflection-Based Approaches. We implement two reflection-based approaches as our primary baselines:

Re-ReST (Dou et al., 2024) implements a self-reflection mechanism to optimize self-training data quality. The method operates in two phases: first refining the training dataset through reflective incorporation, then conducting model fine-tuning on the enhanced data. This approach enables implicit integration of reflective insights, allowing for improved performance during single-pass inference. We employ their official implementation[1], adapting it to our experimental settings with corresponding datasets and base LLMs.

Reflection-RAG implements a Retrieval Augmented Generation (RAG) framework (Gao et al., 2023) for reflection-based reasoning. The method stores reflections generated from the training dataset as described in Section 3.1. During inference, it retrieves relevant reflections based on question similarity, leveraging the intuition that similar questions often share comparable solution strategies and hints. The retrieval process consists of two phases for enhanced accuracy: First, we employ `BGE-m3`[2], a widely-adopted text embedding model for RAG systems, to identify the top-6 similar question-reflection pairs. Subsequently, we utilize `BGE-reranker-v2-m3`[3] to re-rank these candidates and select the reflection whose associated question exhibits the highest relevance to the input query. The selected reflection then serves as guidance for the LLM's problem-solving process. To optimize retrieval efficiency, we cache question embedding matrices in GPU memory, significantly reducing retrieval latency.

E Implementations Details

Models. To evaluate our proposed approach, we employ three widely used base LLMs as Actor LLMs: `Qwen-2-7B-Instruct`[4] (Yang et al., 2024), `Llama-3.1-8B-Instruct`[5] (Dubey et al., 2024), and `CodeLlama-7B-Instruct`[6] (Roziere et al., 2023). Additionally, we utilize `Qwen-2-72B-Instruct`[7] as the Reflector Model in our experiments.

Implementations Details. In the reflection generation phase, we set a maximum of 4 iteration steps, discarding data that fails to solve the problem correctly after 4 action-reflection loops. To ensure certainty, we set the reflector LLM's temperature to 0, eliminating sampling variability.

For codebook tuning, we employ grid search to identify optimal hyper-parameters across various tasks. The codebook size is selected from {512, 1024}, positioned at either the last 3rd, 6th, or 9th layer. The number of selected reflective units is chosen from {16, 32, 64}.

We implement a progressive optimization paradigm to enhance model performance. During meta-reflection alignment, we set the epoch to either 1 or 2 with a learning rate of 1e-4. We utilize the Sinkhorn Algorithm to approximate the

[1] https://github.com/PlusLabNLP/Re-ReST

[2] https://huggingface.co/BAAI/bge-m3

[3] https://huggingface.co/BAAI/bge-reranker-v2-m3

[4] https://huggingface.co/Qwen/Qwen2-7B-Instruct

[5] https://huggingface.co/meta-llama/Llama-3.1-8B-Instruct

[6] https://huggingface.co/meta-llama/CodeLlama-7b-Instruct-hf

[7] https://huggingface.co/Qwen/Qwen2-72B-Instruct

transportation matrix, with 10 iterations to ensure accurate approximation (details in Appendix A).

For supervised fine-tuning (SFT), we explore either 2 or 3 epochs with learning rates selected from {1e-4, 5e-5, 1e-5} for tuning.

F Prompts

In this section, we present the domain-specific prompt templates utilized in our approach for various task domains. We emphasize that the '{reflection}' component is only integrated into the prompt after the actor LLMs' first attempt. Initial trials are executed without any reflective guidance to establish baseline performance.

F.1 Prompts for Programming Tasks

Prompt for Actor LLMs:
You are an AI that only responds with python code, NOT ENGLISH. You will be given a function signature and its docstring by the user. Write your full implementation (restate the function signature, the class definition, or the necessary libraries).
[Function signature]: {func_sign}
[Your code should pass these tests]: {unit tests}
[Hint or past experience that may guide you]: {reflection}

Prompt for Reflector LLMs:
You are a Python programming assistant, your task is to instruct a student on correcting a mistake in a programming question. You will be given:
1. A function signature.
2. The student's implementation
3. A series of unit tests for the implementation.
Your goal is to write a few sentences to provide a corrective solution that can solve not only this question but also a series of similar questions. Remember point out the common pitfalls or easily misunderstood aspects of this problem based on the student's incorrect implementation. Then the student need this as a hint when he/she try again later. Only provide the few sentence description in your answer, not the implementation.
Example output: 'The hint to this programming problem is ...'
[Function signature]: {func_sign}
[Function impl]: {fun_impl}
[Unit test results]: {test results}

F.2 Prompts for Mathematical Reasoning Task

Prompt for Actor LLMs:
You are an AI assitant, you are required to solve mathematical question.
[Question]: {question}
[Hint or past experience that may guide you]: {reflection}

Prompt for Reflector LLMs:
You are a mathematical expert, your task is to instruct a student on correcting a mistake in a math question. Note that you should ONLY provide a corrective solution that can solve not only this question but also a series of similar questions, and you must not reveal the answer to prevent leaking. Your output should only contain the solution without any explanation.
Example output: 'For this question, you should first calculate...'
[Question]: {question}
[Student response]: {response}

F.3 Prompts for E-commerce Customer Intent Detection Task

Prompt for Actor LLMs:
你是一个来自电商平台的AI客服智能助手，你的输入分为两部分：
用户需求以及订单的信息，分为以下五个字段内容： 1. 用户遇到的问题，即用户遭遇到的异常情况或障碍； 2. 用户的诉求，即用户所有的在与助手、商家和平台人工客服沟通过程中表达的想要实现的目的或达成的内容以及主动发起的申请，包括退款申请、投诉申请、赔偿申请等； 3. 平台或商家给出的解决方案； 4. 用户对解决方案表达的态度； 5. 处理状态；
定义好的诉求清单，用列表作为输入，其中一共有6个诉求，诉求由字母+诉求文字表示（比如 'B 退运费'）

你现在需要根据以上信息从诉求清单列表中选择出最匹配的用户诉求，你的输出应该包括： 1.你的思考过程 2.诉求清单中最为匹配的诉求对应的字母，有且仅有一个。
[问题]: {question}
[一些可能对你有用的提示和来自过去的错误经验]: {reflection}

Prompt for Reflector LLMs:
你是一个智能AI助手，现在需要你解决一些电商智能助手在推断用户诉求时存在的问题。 目前输入分为三部分内容:
用户需求以及订单的信息，分为以下五个字段内容： 1. 用户遇到的问题，即用户遭遇到的异常情况或障碍； 2. 用户的诉求，即用户所有的在与客服、商家和平台人工客服沟通过程中表达的想要实现的目的或达成的内容以及主动发起的申请，包括退款申请、投诉申请、赔偿申请等； 3. 平台或商家给出的解决方案； 4. 用户对解决方案表达的态度； 5. 处理状态;
定义好的诉求清单，用列表作为输入，其中一共有6个诉求，诉求由字母+诉求文字表示（比如 'A 退运费'），核心任务是根据用户需求和订单信息选择出最匹配的诉求
一段错误的匹配过程，其中包括思考过程和预测的诉求
现在需要你对上述错误的匹配过程的进行反思，并提供正确的解决方案，以指导再次遇到类似订单情况下能够找出最匹配的诉求。注意，你的输出不应该包括正确答案（防止出现答案泄漏），应该给出如何思考从而指导下一次的匹配过程，并且保证通用性（对相似问题也可以提供帮助）。"'
[问题]: {question}
[匹配过程]: {response}

G Case Study

We conduct a case study on the GSM8K dataset. As illustrated in Figure 7, we compare three distinct methodologies. The base LLM, under Zero-Shot settings, demonstrates a lack of domain-relevant knowledge and fails to solve the problem without external guidance. The Reflection-RAG approach retrieves similar problems from the training knowledge base and leverages their associated reflections as guidance. However, despite the high similarity of retrieved problems, their reflection guidance processes often deviate significantly from the required reasoning path of the given problem. This misalignment prevents fine-grained guidance and introduces noise, resulting in suboptimal performance. In contrast, our proposed method achieves superior performance by incorporating reflective insights into the codebook and retrieving question-specific reflective insights during inference, enabling precise step-by-step guidance for the LLM to successfully solve the problem.

H Visualization

We visualize the selection frequency distribution of reflective units in the meta-reflection codebook across three additional benchmark datasets from programming and mathematical reasoning domains. As shown in Figure 8, the reflective units exhibit significant variations, consistent with the findings in Section 4.7. This distribution pattern indicates that the retrieval process adaptively selects different reflective units based on the specific questions, thereby providing tailored guidance for LLMs in problem-solving tasks.
Additionally, we visualize the feature distributions of reflective units in the meta-reflection codebook. Each reflective unit is first reduced to one dimension through dimensionality reduction and subsequently normalized. As shown in Figure 9, the results demonstrate diverse distributions across reflective units, indicating their ability to capture varied semantic information without feature space collapse (Jing et al.).

Question

Every 2 miles a car drives the tires rotate 725 times. Jeremy drives 400 miles a month. If each tire can undergo 10,440,000 rotations how many years before the tire needs to be replaced?

Zero-Shot

To find out how many years before the tire needs to be replaced, we need to calculate the total number of rotations the tire will undergo in a year, and then divide that by the total number of rotations the tire can handle ... Now, let's divide the total number of rotations the tire can handle (10,440,000) by the number of rotations per year (2400):10,440,000 rotations / 2400 rotations/year = 4346.67 years. So, the tire will need to be replaced in approximately 4347 years.

Reflection-RAG

[Similar Question]: Nina loves to travel. She tries to travel at least 400 kilometers in one month outside of her home country. Every second month she does twice that distance. If she were able to keep up with her resolution, how many kilometers would she travel during 2 years?

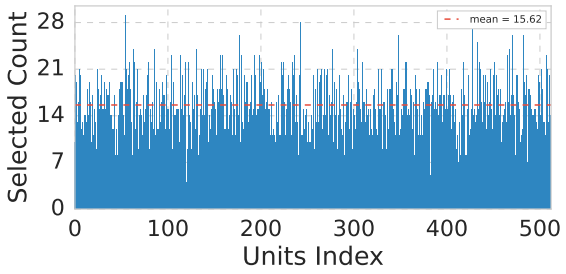
[Reflection]: For this question, you should first calculate the total distance Nina travels in one month and then consider the pattern of her travel distance every second month. After that, calculate the total distance she would travel in 2 years by considering the pattern of her travel distance.

To solve this problem, we need to follow these steps: ##Step 1: Calculate the total distance Jeremy travels in one month. Jeremy drives 400 miles a month. ## Step 2: ... The number of years before the tire needs to be replaced is 10440000 / 3480000 = 3. The final answer is: 3.

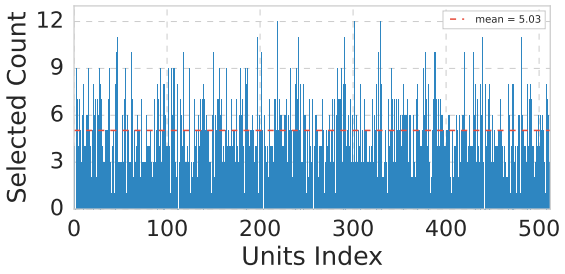
Meta-Reflection

To solve this problem, we need to follow these steps: 1. Calculate the total number of rotations per mile. 2. Calculate the total number of rotations per 400 miles.3. Calculate the total number of years before the tire needs to be replaced. Step 1: Calculate the total number of rotations per mile. The car drives the tires 725 times every 2 miles. To find the number of rotations per mile, we divide 725 by 2:725 / 2 = 362.5 rotations per mile. Step 2: Calculate the total number of rotations per 400 miles. Since Jeremy drives 400 miles a month, we multiply the number of rotations per mile by 400:362.5 * 400 = 145,000 rotations per month. Step 3: Calculate the total number of years before the tire needs to be replaced. We know that each tire can undergo 10,440,000 rotations. To find the number of months before the tire needs to be replaced, we divide the total number of rotations by the number of rotations per month:10,440,000 / 145,000 = 72 months. To convert the number of months to years, we divide by 12:72 / 12 = 6 years. Therefore, the tire needs to be replaced in 6 years. #### 6.

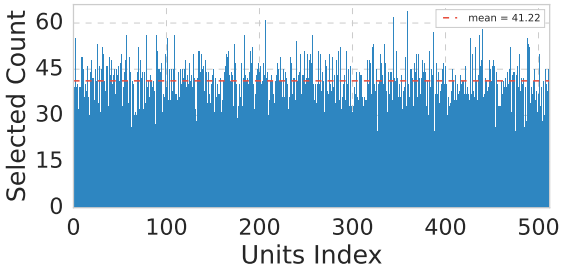
Figure 7: A case study on the GSM8K dataset.



(a) MBPP

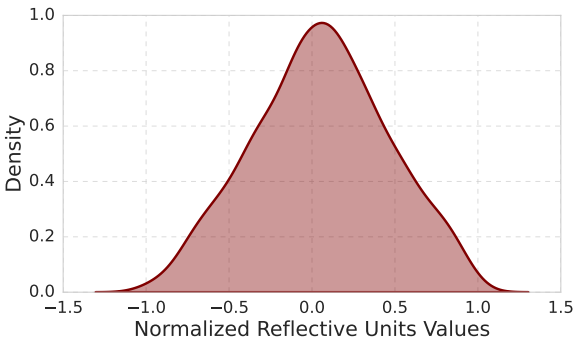


(b) HumanEval

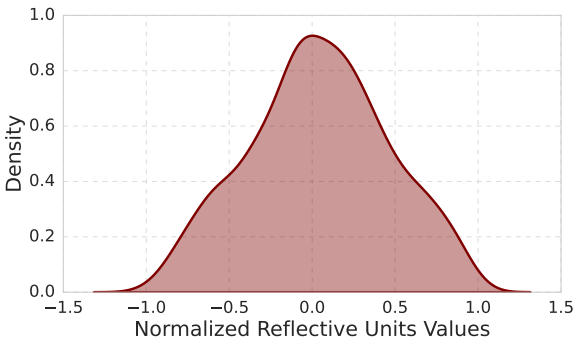


(c) GSM8K

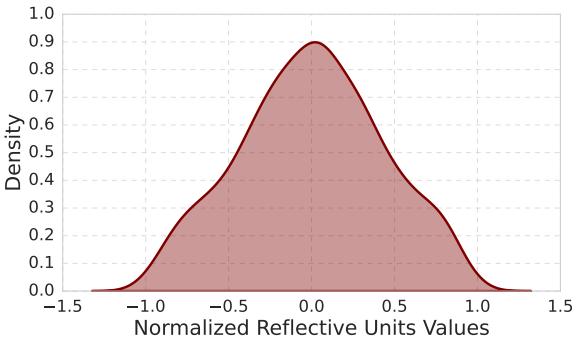
Figure 8: Visualization of reflective unit selection frequency distributions across three benchmark datasets in programming and mathematical reasoning domains. A meta-reflection codebook of size 512 is uniformly maintained, with 16 units uniformly selected per inference.



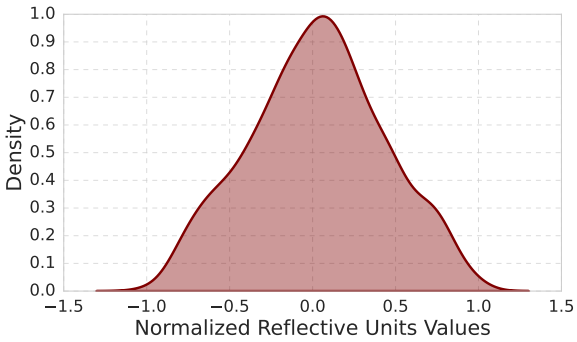
(a) MBPP



(b) HumanEval



(c) GSM8K



(d) ECID

Figure 9: Visualization of feature distributions for reflective units in the meta-reflection codebook.