SALA: Semantic-Aware Logical Alignment for Complex Reasoning in In-Context Learning

Zhao Ji[1,2], Wenqing Chen[1,2]*, Zhixuan Chu[3], Jianxing Yu[4,5], Jingping Liu[1,2], Shanhe Zhao[6], Zibin Zheng[1,2]

[1]School of Software Engineering, Sun Yat-sen University, Zhuhai, China
[2]Zhuhai Key Laboratory of Trusted Large Language Models, Zhuhai, China
[3]The State Key Laboratory of Blockchain and Data Security, Zhejiang University, Hangzhou, China
[4]School of Artificial Intelligence, Sun Yat-sen University, Zhuhai, China
[5]Key Laboratory of Sustainable Tourism Smart Assessment Technology, Ministry of Culture and Tourism, Zhuhai, China
[6]Merchants Union Consumer Finance Company Limited, Shenzhen, China

Abstract

Effective in-context learning (ICL) for complex reasoning relies on selecting the right demonstrations. Traditional retrieval methods based on surface similarity fail to capture the underlying problem-solving logic. Recent logic-based methods address this by matching predefined reasoning steps, but the rigid rules and exact-match criteria is improper to handle flexible or diverse reasoning processes. To address the problem, we propose **SALA**, a **S**emantic-**A**ware **L**ogical **A**lignment framework. Instead of relying on a fixed inventory, SALA automatically learns task-specific reasoning operations. It then embeds these operations into a continuous semantic space and uses dynamic time warping (DTW) to align the reasoning sequences. This approach allows for soft, flexible matching of reasoning logic while remaining highly interpretable. Experiments across four reasoning benchmarks and three LLMs demonstrate that SALA outperforms existing demonstration selection methods. Further analysis confirms the roles of the operation induction and the logical semantic alignment.

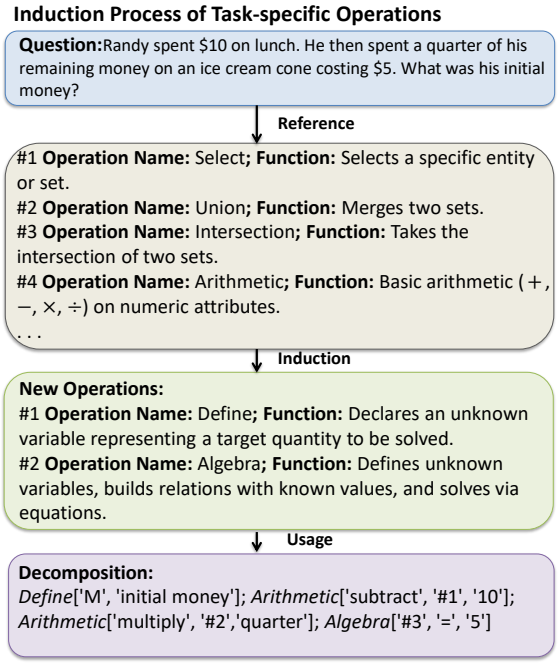

Figure 1: An example of the induction process of task-specific operations.

1 Introduction

Large Language Models (LLMs) (Ouyang et al., 2022; Achiam et al., 2023; Touvron et al., 2023; Bai et al., 2023; Ye et al., 2023b; Bahrini et al., 2023) have shown strong in-context learning (ICL) ability (Dong et al., 2024; Wies et al., 2023; Luo et al., 2024; Bhattamishra et al., 2024), enabling them to solve complex reasoning tasks from a few labeled demonstrations. ICL performance largely depends on demonstration selection, since useful examples can improve accuracy (Ye et al., 2023a), whereas mismatched or noisy examples may introduce reasoning biases and degrade performance (Zhang et al., 2025c). The importance of selecting useful examples is further enhanced in recent agentic LLM systems, where reasoning experiences are stored as memories and retrieved to guide future tasks (Ouyang et al., 2025).

Existing studies have improved demonstration selection in general domain (Rubin et al., 2022; Li et al., 2023; Ye et al., 2023a; Yang et al., 2023; Qin et al., 2024). However, most retrieval models rely on embedding similarity and the reasoning logic can be obscured by surface semantics.

The key question is how to represent demonstrations for reasoning tasks. Recent reasoning-

*Corresponding author.

Findings of the Association for Computational Linguistics: EMNLP 2026, pages 21440–21456
October 24-29, 2026 ©2026 Association for Computational Linguistics

aware methods construct intermediate representations before retrieval, including generated reasoning paths (Qin et al., 2024), reasoning patterns (Zhang et al., 2025b), latent reasoning skills (Xu et al., 2024), and reasoning graphs (Lin et al., 2025). These methods suggest that raw question embeddings are often insufficient, since reasoning-relevant signals can be diluted by topics, entities, and surface semantics. Problem-solving logic (PSL) takes a more symbolic route by representing each problem as a sequence of predefined reasoning operations and matching demonstrations at the operation level (Ma et al., 2025). Such operation-based representations reduce the influence of surface wording and provide a more explicit view of the problem-solving process.

Despite these advances, existing representations still struggle to make reasoning logic both comparable and adaptable. Free-form rationales or reasoning paths can express diverse reasoning processes, but their natural-language form often makes the representation noisy and unstable. Symbolic operation sequences, as explored by PSL, make reasoning steps easier to compare, but fixed operation spaces and exact matching rules limit their ability to generalize across task-specific and semantically equivalent reasoning patterns. This motivates a logical alignment framework that represents reasoning explicitly while aligning it semantically.

To this end, we propose **SALA**, a **S**emantic-**A**ware **L**ogical **A**lignment framework for reasoning-oriented demonstration selection. SALA constructs a task-adaptive operation space by inducing reasoning operations from downstream data, enabling demonstrations to be represented with problem-solving units beyond the predefined operation set. Figure 1 illustrates how SALA induces additional task-specific operations when the predefined operation set cannot fully express the reasoning logic of a downstream question. Then we embed operation descriptions into a continuous semantic space and applies dynamic time warping (DTW) to align reasoning-operation sequences (Sakoe and Chiba, 1978). In this way, we keep the reasoning representation explicit while enabling soft semantic alignment beyond exact symbolic correspondence.

We conduct experiments on four reasoning benchmarks across three LLMs. SALA achieves better average performance against similarity-based, learning-based, and reasoning-aware demonstration selection baselines. Ablation studies further show that both task-adaptive operation construction and semantic sequence alignment contribute to the improvement. The key contributions of this work are as follows:

- We propose SALA, a reasoning-oriented demonstration selection framework that represents problem-solving logic with explicit operation sequences and aligns them in semantic space.
- We introduce a task-adaptive operation space that extends predefined operations with reusable reasoning units induced from downstream data, enabling more expressive problem-solving representations without manual operation design.
- We design a semantic DTW-based alignment strategy to measure logical similarity across reasoning-operation sequences with different lengths and decomposition granularities.
- We validate SALA on four reasoning benchmarks and three LLMs, showing consistent average gains over recent state-of-the-art baselines and confirming the roles of operation construction and semantic alignment through ablations.

2 Related Work

Existing work on ICL demonstration selection (Dong et al., 2024) has studied this problem from lexical, semantic, learned, diversity-aware, and reasoning-aware perspectives. We review the most relevant lines of work in general and reasoning domains.

2.1 General Demonstration Selection

Early demonstration selection methods retrieve examples according to lexical overlap or sentence-level semantic similarity. Sparse retrieval methods such as BM25 (Robertson and Zaragoza, 2009) estimate query-example word overlap, while embedding-based methods use pretrained encoders such as BERT (Devlin et al., 2019) to retrieve semantically similar demonstrations. Later studies improve this basic retrieval paradigm by considering supportiveness, diversity, and compositionality. For example, support-example selection chooses demonstrations according to their usefulness for the target query (Li and Qiu, 2023). DPP-based selection balances relevance and diversity (Yang et al., 2023), and compositional exemplar selection

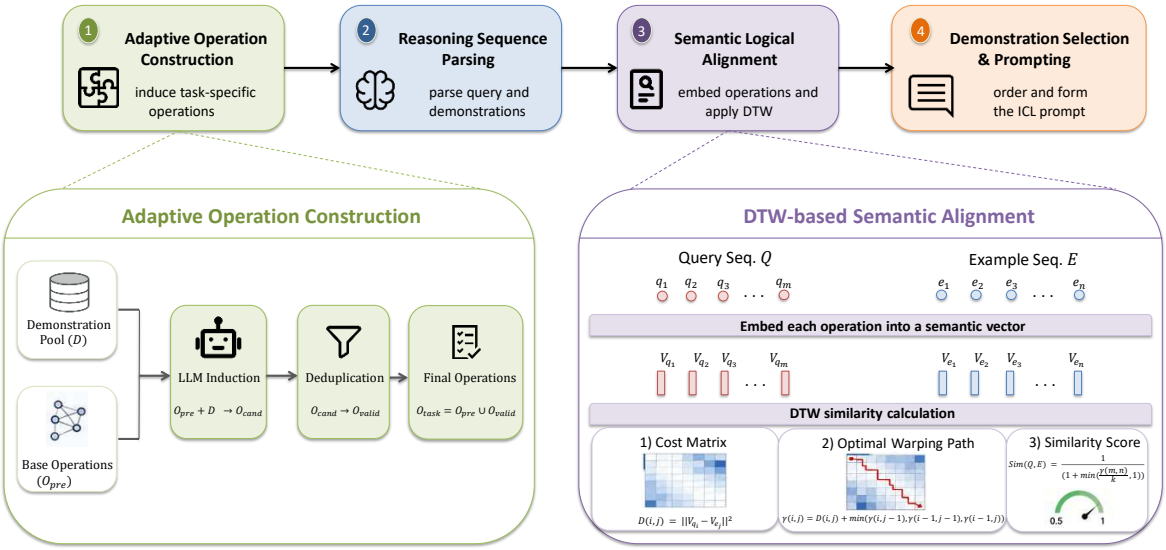

Figure 2: Overview of the SALA framework. SALA constructs a task-adaptive operation set, parses queries and demonstrations into reasoning-operation sequences, and applies DTW-based semantic alignment to retrieve logically similar demonstrations for ICL prompting.

models interactions among demonstrations beyond nearest-neighbor retrieval (Ye et al., 2023a). Iterative selection further shows that demonstration retrieval can be treated as a multi-step process rather than a one-shot nearest-neighbor search (Qin et al., 2024).

Another line of work formulates demonstration selection as a learnable retrieval or optimization problem. EPR trains a retriever from language-model feedback (Rubin et al., 2022), while UDR learns a unified retriever for cross-task demonstration selection (Li et al., 2023). More recent methods use model preference, gradient matching, reinforcement learning, or many-shot optimization to improve selection quality (Zhang et al., 2025c,a; Wang et al., 2025; Purohit et al., 2025). These methods improve the retrieval of useful demonstrations, but their selection signals are still mostly derived from input similarity or model-level feedback rather than the reasoning process itself. For complex reasoning tasks, a helpful demonstration should not only be topically or semantically related to the query, but also follow a compatible problem-solving logic.

2.2 Reasoning Demonstration Selection

Recent work has started to construct reasoning-oriented representations. One direction uses natural-language reasoning descriptions. Skill-KNN rewrites inputs into skill-based descriptions before applying embedding-based retrieval (An et al., 2023). Luo et al. (2023) extend retrieval-based ICL to chain-of-thought (CoT) prompting, and iterative demonstration selection uses generated reasoning paths to guide example retrieval (Qin et al., 2024). These methods keep the representation flexible, but the reasoning signal is still expressed in free-form text, which can make the retrieval noisy or unstable.

A second direction introduces more structured reasoning representations. Reasoning-pattern methods select demonstrations according to task-specific reasoning patterns (Zhang et al., 2025b), LaRS learns latent reasoning skills from CoT rationales (Xu et al., 2024), and RGER represents intermediate reasoning steps as reasoning graphs for exemplar retrieval (Lin et al., 2025). For mathematical reasoning, LMS3 further shows that useful demonstrations should balance semantic similarity and inference stability (Liu et al., 2025). These methods provide stronger reasoning-oriented signals than raw question embeddings, but latent skills are less directly interpretable, and graph-based representations require more complex structure matching across examples.

PSL-guided ICL is the most closely related symbolic approach (Ma et al., 2025). It represents each problem as a sequence of predefined QDMR-style reasoning operations (Wolfson et al., 2020) and selects demonstrations through operation-level exact matching. While this shows the value of explicit operation sequences for representing problem-solving logic, it still relies on a fixed operation space and

rigid symbolic matching. SALA moves beyond symbolic matching by combining operation-level reasoning representations with semantic alignment. It constructs a task-adaptive operation space from downstream data and aligns operation sequences in semantic space with DTW, allowing explicit reasoning representations to be compared flexibly across diverse reasoning processes.

3 Methodology

As shown in Figure 2, SALA follows a four-stage pipeline for reasoning-oriented demonstration selection, with adaptive operation construction and DTW-based semantic alignment serving as the two key mechanisms.

3.1 Problem Formulation

Let $\mathcal{D} = \{(q_i, y_i)\}_{i=1}^{N}$ denote the demonstration pool derived from the training set of a downstream task, where q_i and y_i are the question and its answer, respectively. Given a test query q^*, the goal is to select k demonstrations from $\mathcal{D}$ that support the target LLM in solving q^*.

For reasoning tasks, we model the problem-solving logic of a question as an ordered sequence of reasoning operations. Given an operation set $\mathcal{O}$, a parser maps a question q into

$$\pi(q; \mathcal{O}) = [o_1, \ldots, o_m], \quad o_j \in \mathcal{O}. \tag{1}$$

After constructing the task-adaptive operation set $\mathcal{O}_{task}$, we denote the query sequence and the candidate sequence as $Q^* = \pi(q^*; \mathcal{O}_{task})$ and $E_i = \pi(q_i; \mathcal{O}_{task})$, respectively. Demonstration selection is then formulated as:

$$\mathcal{D}_k(q^*) = \mathrm{TopK}_{(q_i, y_i) \in \mathcal{D}} \, Sim(Q^*, E_i), \tag{2}$$

where $Sim(Q^*, E_i)$ measures the alignment between the problem-solving logic of the query and that of the candidate demonstration. SALA focuses on constructing the task-adaptive operation set $\mathcal{O}_{task}$ and defining $Sim(\cdot, \cdot)$ through semantic DTW alignment.

3.2 Adaptive Construction of Reasoning Operation Sets

To address the limited transferability of fixed predefined reasoning operation sets, we construct a task-adaptive reasoning operation set $\mathcal{O}_{task}$ by extending the 13 predefined QDMR reasoning operations $\mathcal{O}_{pre}$ (Wolfson et al., 2020) (see Appendix B for the full list) with operations induced from downstream training data. The construction process consists of candidate operation induction followed by two-stage deduplication.

3.2.1 Candidate Operation Induction

For each training question q_i in the candidate demonstration pool $\mathcal{D}$, we condition an LLM on the predefined QDMR operation set $\mathcal{O}_{pre}$ and prompt it to identify additional operations only when the existing set is insufficient to cover the problem-solving logic of q_i. If no extension is needed, the model returns an empty result; otherwise, it outputs one or more candidate operation descriptions in the predefined format. Collecting the non-empty outputs over all questions yields a candidate operation pool $\mathcal{O}_{cand} = \{o_1, o_2, \ldots, o_M\}$, where M is the number of candidate new operations. The full prompt used for this step is provided in Appendix C.1.

3.2.2 Two-Stage Deduplication of New Operations

To avoid redundancy and ensure that newly induced operations truly complement the predefined ones, we apply a two-stage deduplication procedure.

Stage 1: Heuristic Name-Based Deduplication. We first perform a lightweight heuristic deduplication over the names of candidate operations. For each candidate operation $o_k \in \mathcal{O}_{cand}$, we extract its operation name, denoted as n_k. We then normalize the name by lowercasing it and removing spaces and underscore characters, yielding a normalized form $\tilde{n}_k$. Candidate entries whose extracted names contain obvious formatting artifacts or malformed markers are discarded at this stage. Let $\mathcal{O}_{rep}$ denote the retained representative operation set, initialized as an empty set. We process the candidates sequentially and compare the normalized name of the current candidate with the normalized operation names already stored in $\mathcal{O}_{rep}$.

For a current candidate o_k with normalized name $\tilde{n}_k$, and an existing representative operation $\bar{o}_i \in \mathcal{O}_{rep}$ with normalized name $\tilde{n}_{\bar{o}_i}$, we apply the following deterministic rules:

- If $\tilde{n}_k \subseteq \tilde{n}_{\bar{o}_i}$, we replace $\bar{o}_i$ with o_k.
- If $\tilde{n}_{\bar{o}_i} \subseteq \tilde{n}_k$, we discard o_k.
- Otherwise, we keep both entries unchanged.

If no containment relation is found between $\tilde{n}_k$ and any existing representative in $\mathcal{O}_{rep}$, we append o_k to $\mathcal{O}_{rep}$. After all candidates are processed,

$\mathcal{O}_{rep} = \{\bar{o}_1, \bar{o}_2, \ldots, \bar{o}_P\}$ becomes the representative operation set produced by Stage 1. This heuristic removes near-duplicate operation names caused by minor formatting differences or partially overlapping name variants, while preserving the original full operation descriptions for subsequent processing. Because the procedure depends only on normalized names, containment checks, and candidate order, it is reproducible given the same candidate operation pool.

Stage 2: Function Duplication Detection Based on LLM Judgment. When $\mathcal{O}_{rep} = \{\bar{o}_1, \bar{o}_2, \ldots, \bar{o}_P\}$ is obtained after Stage 1, we further remove functional redundancy through a sequential judgment process. We initialize the task-adaptive operation set as $\mathcal{O}_{task}^{(0)} = \mathcal{O}_{pre}$. For each representative operation $\bar{o}_i \in \mathcal{O}_{rep}$, we construct a function comparison prompt that includes the core function description of $\bar{o}_i$ and the core function descriptions of all operations in the current operation set $\mathcal{O}_{task}^{(i-1)}$. The LLM outputs a binary judgment result $J(\bar{o}_i, \mathcal{O}_{task}^{(i-1)}) \in \{0, 1\}$, where $J(\bar{o}_i, \mathcal{O}_{task}^{(i-1)}) = 1$ means that $\bar{o}_i$ functionally overlaps with at least one operation in $\mathcal{O}_{task}^{(i-1)}$, and $J(\bar{o}_i, \mathcal{O}_{task}^{(i-1)}) = 0$ means no overlap.

The operation set is then updated recursively:

$$\mathcal{O}_{task}^{(i)} = \begin{cases} \mathcal{O}_{task}^{(i-1)} \cup \{\bar{o}_i\}, & \text{if } J(\bar{o}_i, \mathcal{O}_{task}^{(i-1)}) = 0, \\ \mathcal{O}_{task}^{(i-1)}, & \text{otherwise.} \end{cases} \tag{3}$$

After all representative operations are examined, the final task-adaptive reasoning operation set is $\mathcal{O}_{task} = \mathcal{O}_{task}^{(P)}$.

This sequential process ensures that each accepted operation complements both the predefined operations and the previously accepted induced operations. The prompts for candidate induction (Section C.1) and deduplication (Section C.3) are fully documented in Appendix C.

3.3 Construction of the Reasoning Operation Embedding Library

To capture the functional semantic associations of operations and support semantic alignment, we convert discrete operations into continuous semantic embeddings and construct a reasoning operation embedding library, providing the foundation for subsequent similarity calculations.

First, we generate semantic description texts for each operation $o \in \mathcal{O}_{task}$, denoted as $desc(o)$. In our implementation, each $desc(o)$ describes the core function of the corresponding operation, and the description texts are used as inputs to a pretrained embedding model.

Next, we use a pretrained text embedding model as the semantic encoder, denoted as $f_{emb}(\cdot) : str \to \mathbb{R}^d$, where d is the embedding dimension. Specifically, the implementation uses a Hugging Face embedding model to encode the operation description text and obtains the initial semantic embedding vector through mean pooling over the hidden states:

$$u_o^{init} = f_{emb}(desc(o)) \in \mathbb{R}^d \tag{4}$$

We apply ℓ_2 normalization to the initial vector before DTW matching:

$$v_o = \frac{u_o^{init}}{\|u_o^{init}\|_2} \tag{5}$$

where $v_o \in \mathbb{R}^d$ and $\|v_o\|_2 = 1$ is the final semantic embedding vector of operation o.

Finally, the mapping relationship, $\{(o, v_o) \mid o \in \mathcal{O}_{task}\}$, is used to form the reasoning operation embedding library L_{emb}. In practice, the embeddings are precomputed once for all operations in $\mathcal{O}_{task}$ and then reused during retrieval.

3.4 DTW-Based Semantic Alignment

In this section, we propose a semantic alignment strategy that maps reasoning operation sequences to semantic embedding sequences and uses DTW to calculate sequence-level similarity (Sakoe and Chiba, 1978), enabling soft logical alignment across sequences of different lengths.

3.4.1 Reasoning Operation Sequence Parsing

Based on the constructed task-adaptive reasoning operation set $\mathcal{O}_{task}$, we use an LLM to map each query and candidate demonstration into an ordered reasoning operation sequence. Each sequence represents the problem-solving process as a series of operations drawn from $\mathcal{O}_{task}$. The parsing prompt is provided in Appendix C.2.

3.4.2 Semantic Sequence Generation

We retrieve the semantic embedding of each reasoning operation in the parsed reasoning operation sequence from the embedding library, converting discrete reasoning operation sequences into continuous semantic embedding sequences. For the query sequence $Q^* = [o_1^*, o_2^*, \ldots, o_m^*]$, its operation embedding sequence is $\mathbf{Q}^* = [v_{o_1^*}, v_{o_2^*}, \ldots, v_{o_m^*}]$, where $v_{o_i^*}$ is the semantic embedding of the i-th query operation o_i^*. For a candidate demonstration sequence E_i, we simplify it to $E = [o_1^E, o_2^E, \ldots, o_n^E]$

when computing DTW, and its semantic sequence is $\mathbf{E} = [v_{o_1^E}, v_{o_2^E}, \ldots, v_{o_n^E}]$, where $v_{o_j^E}$ is the semantic embedding of the j-th candidate operation o_j^E.

3.4.3 DTW-based Similarity Calculation

The DTW algorithm calculates the semantic similarity between the candidate semantic sequence $\mathbf{E}$ and the query semantic sequence $\mathbf{Q}^*$ through three steps. First, we construct a pairwise distance matrix $D \in \mathbb{R}^{m \times n}$, where the element $D(i, j)$ represents the semantic distance between the i-th vector $v_{o_i^*}$ in $\mathbf{Q}^*$ and the j-th vector $v_{o_j^E}$ in $\mathbf{E}$. Consistent with the implementation, we use Euclidean distance to measure this cost:

$$D(i, j) = \|v_{o_i^*} - v_{o_j^E}\|_2. \tag{6}$$

where $\| \cdot \|_2$ denotes the Euclidean norm. Since the operation embeddings are precomputed in a fixed d-dimensional space, this distance directly measures the semantic discrepancy between two reasoning operations.

Next, we search for the optimal warping path. The warping path $W = [w_1, w_2, \ldots, w_k]$ is a sequence of coordinate pairs (i, j) that satisfies three constraints to ensure a valid alignment: the boundary constraint (starting at $(1, 1)$ and ending at (m, n) to cover the entire problem-solving logic), the monotonicity constraint (avoiding reverse matching to maintain consistency with the reasoning process), and the continuity constraint (ensuring continuous paths without jumps to avoid missing key reasoning steps). The optimal warping path is the one with the minimum cumulative cost among all feasible paths, where the cumulative cost $\gamma(i, j)$ from the starting point $(1, 1)$ to the point (i, j) is calculated recursively as:

$$\begin{aligned} &\gamma(i, j) = D(i, j) + \\ &\min\{\gamma(i-1, j), \gamma(i, j-1), \gamma(i-1, j-1)\}. \end{aligned} \tag{7}$$

with the initial condition $\gamma(1, 1) = D(1, 1)$.

Finally, we calculate the sequence similarity. To reduce the impact of sequence length differences, we divide the cumulative DTW distance by the length of the optimal alignment path to obtain the average alignment distance, and then transform it into a bounded similarity score:

$$Sim(Q^*, E_i) = \frac{1}{1 + \min\left(\frac{\gamma(m,n)}{k}, 1\right)}. \tag{8}$$

where k denotes the length of the optimal warping path. Here, $\gamma(m, n)/k$ is the average DTW alignment distance per step. Since operation embeddings are ℓ_2-normalized unit vectors, their pairwise Euclidean distance lies in $[0, 2]$, and the $\min(\cdot, 1)$ clipping bounds the effective distance to $[0, 1]$. Therefore, $Sim(Q^*, E_i) \in [0.5, 1]$, and a value closer to 1 indicates higher semantic similarity and greater consistency in problem-solving logic between the demonstration and the query.

We rank all demonstrations by DTW semantic similarity in descending order and select the top-k most similar ones. Then, following an easy-to-hard curriculum (Ma et al., 2025), we sort the selected demonstrations in ascending order of reasoning operation sequence length.

4 Experiments

4.1 Experimental Setup

4.1.1 Datasets

We evaluate SALA on four reasoning benchmarks covering diverse task types and difficulty levels. SVAMP (Patel et al., 2021) contains 1,000 arithmetic word problems. GSM8K (Cobbe et al., 2021) consists of 1,319 high-quality grade-school math problems that typically require multi-step reasoning. CommonsenseQA (Talmor et al., 2019) is a multiple-choice commonsense reasoning benchmark with 10,881 questions, each associated with five answer options. StrategyQA (Geva et al., 2021) is an open-domain question-answering benchmark with 2,290 questions. Detailed dataset statistics and split information are provided in Appendix A.

4.1.2 Baselines

We compare SALA with seven representative ICL baselines, including **Random** sampling, **EPR** with a contrastive retriever (Rubin et al., 2022), **BM25** retrieval (Robertson and Zaragoza, 2009), **TopK-BERT** retrieval (Devlin et al., 2019), and **DPP-BERT**, which combines BERT-based retrieval with a Determinantal Point Process to balance relevance and diversity (Chen et al., 2018; Yang et al., 2023). We also include two reasoning-aware baselines: **PSL**, which selects demonstrations using problem-solving logic guidance (Ma et al., 2025), and **LMS3**, which selects demonstrations based on semantic similarity and inference stability (Liu et al., 2025).

4.1.3 Implementation Details

We evaluate SALA on three LLMs: Llama3-8B-Instruct (Grattafiori et al., 2024), Qwen2.5-7B-In-

Method	GSM8K	SVAMP	CMSQA	StrQA	Avg
Llama3-8B-Instruct					
Random	80.91	85.40	72.91	82.27	80.37
EPR	79.45	84.33	69.94	83.99	79.43
BM25	79.88	85.53	69.26	83.49	79.54
TopK-BERT	80.97	85.00	71.25	84.13	80.34
DPP-BERT	79.65	84.20	70.81	82.91	79.39
PSL	82.03	84.67	72.89	84.28	80.97
LMS3	**82.65**	85.40	73.89	84.89	81.71
SALA	82.59	**87.13**	**74.07**	**87.25**	**82.76**
Qwen2.5-7B-Instruct					
Random	89.25	82.73	71.84	83.29	81.78
EPR	87.95	88.00	74.12	81.95	83.01
BM25	87.08	86.53	73.84	81.74	82.28
TopK-BERT	87.26	85.33	**75.18**	82.97	82.69
DPP-BERT	88.07	83.86	73.59	83.05	82.14
PSL	89.76	89.33	73.05	80.20	83.09
LMS3	**91.52**	83.40	73.23	81.98	82.53
SALA	91.16	**91.13**	74.73	**83.90**	**85.23**
DeepSeek-V4-Pro					
Random	94.54	90.33	74.45	90.25	87.39
BM25	93.87	92.73	76.59	91.59	88.70
TopK-BERT	96.94	92.20	77.61	92.37	89.78
DPP-BERT	95.45	93.00	75.67	92.58	89.18
PSL	96.48	91.86	76.71	93.68	89.68
SALA	**97.12**	**94.67**	**79.28**	**94.61**	**91.42**

Table 1: Main results across three LLMs. EPR and LMS3 are unavailable on DeepSeek-V4-Pro (API-only access). CMSQA denotes CommonsenseQA and StrQA denotes StrategyQA. Bold indicates the best result per model–benchmark.

struct (Yang et al., 2024), and DeepSeek-V4-Pro (DeepSeek-AI, 2026) (accessed via API). For each benchmark, demonstrations are retrieved from the training set. To induce task-specific reasoning operations, we use Seed-OSS-36B-Instruct (ByteDance Seed Team, 2025) on the training set of each benchmark. Using a separate LLM keeps operation construction independent of the evaluated target LLMs and ensures a fair comparison under the same operation space. For semantic matching, we use bert-base-uncased (Devlin et al., 2019) to encode reasoning operations. All experimental results are averaged over five runs. The number of selected examples for all baselines is based on the settings in PSL (Ma et al., 2025) for different benchmarks. Detailed configurations can be found in Appendix D.

4.2 Main Results

Table 1 summarizes the performance of SALA and the baseline methods. On Llama3-8B-Instruct, SALA achieves the best overall average and the top results on SVAMP, CommonsenseQA, and StrategyQA, while remaining competitive on GSM8K. On Qwen2.5-7B-Instruct, SALA again yields the best average, with the strongest results on SVAMP and StrategyQA. On DeepSeek-V4-Pro (DeepSeek-AI, 2026), EPR and LMS3 are omitted as they require model training or internal hidden states inaccessible through API calls. SALA achieves the best results across all four benchmarks with an average of 91.42%. These results show that SALA consistently improves ICL performance across models of different scales and architectures.

Variant	GSM8K	SVAMP	CMSQA	StrQA	Avg
Llama3-8B-Instruct					
w/o DTW+OPs	81.65	85.00	72.40	83.41	80.62
w/o OPs	82.20	87.06	73.73	86.20	82.30
w/o DTW	82.61	86.87	74.01	85.09	82.15
Full	**82.59**	**87.13**	**74.07**	**87.25**	**82.76**
Qwen2.5-7B-Instruct					
w/o DTW+OPs	88.63	86.67	73.55	80.49	82.34
w/o OPs	90.54	87.73	74.22	82.27	83.69
w/o DTW	89.20	90.20	73.02	81.16	83.40
Full	**91.16**	**91.13**	**74.73**	**83.90**	**85.23**

Table 2: Ablation study results. "w/o DTW" uses exact prefix matching similar to PSL; "w/o OPs" uses only the 13 predefined QDMR operations. CMSQA denotes CommonsenseQA and StrQA denotes StrategyQA.

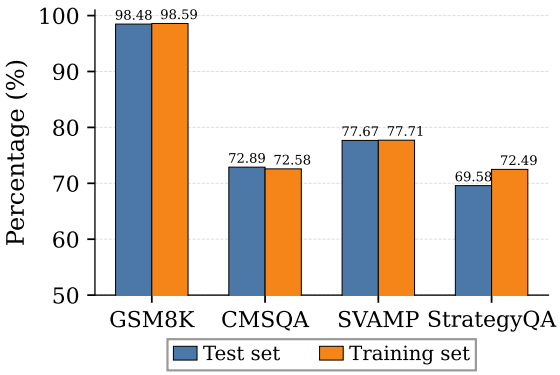

Figure 3: The percentage of samples using task-specific reasoning operations in the training and test sets.

4.3 Ablation Study

To further demonstrate the effectiveness of the SALA framework, we conduct an ablation study on its key modules.

As shown in Table 2, removing either component leads to a consistent performance drop on both models, indicating that both task-adaptive operation construction and DTW-based semantic alignment are necessary for SALA. When the induced operations are removed, SALA falls back to the 13 predefined QDMR operations. This reduces the average accuracy by 0.5 points on Llama3-8B-Instruct and 1.5 points on Qwen2.5-7B-Instruct, showing that the additional operations help repre-

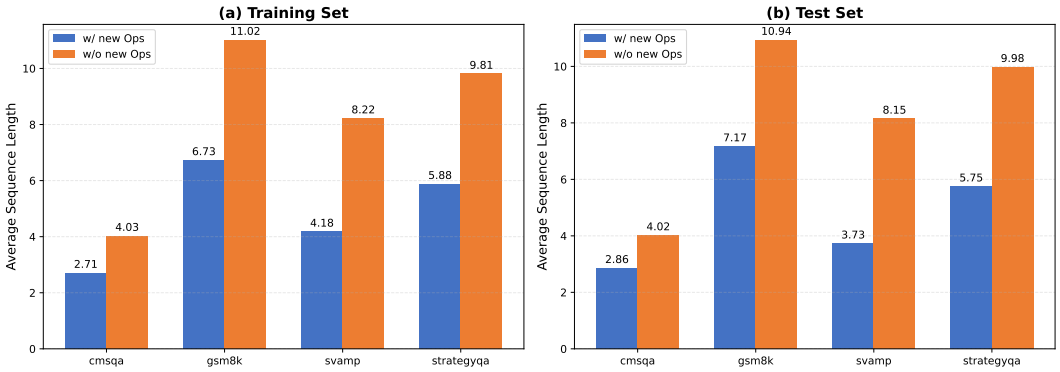

Figure 4: Average reasoning operation sequence length with and without task-specific operations across four benchmarks. (a) Average reasoning sequence length of training set samples; (b) Average reasoning sequence length of test set samples.

sent task-specific reasoning patterns that are not sufficiently covered by the predefined operation set. When DTW is removed and exact prefix matching is used instead, the average accuracy drops by 0.6 points on Llama3-8B-Instruct and 1.8 points on Qwen2.5-7B-Instruct. This suggests that semantic sequence alignment is important for handling semantically related operations and different decomposition granularities. Removing both components causes the largest degradation, with average drops of 2.1 and 2.9 points on the two models, respectively.

4.4 SALA Analysis

4.4.1 Necessity of Inducing Task-Specific Operations

To examine the role of task-specific reasoning operations, we measure the proportion of samples whose reasoning sequences require such operations in both the training and test sets. As shown in Figure 3, this proportion exceeds 68% across all datasets. In GSM8K, it is above 98% for both the training and test sets, whereas for the remaining datasets it generally falls between 70% and 80%. These findings suggest that task-specific reasoning operations are widely involved in reasoning decomposition across diverse tasks. Some representative induced operations are listed in Appendix E.

4.4.2 Qualitative Case Analysis

To further understand how each module contributes to demonstration selection, we present four case studies. Cases 1–2 isolate the effect of task-specific operations (Module 1), while Cases 3–4 isolate DTW semantic alignment (Module 2).

Cases 1–2: Task-Specific Operations Condense Reasoning Sequences Figure 4 shows that adding task-specific operations consistently reduces average reasoning sequence length across all benchmarks. This compaction is critical for accurate demonstration matching. In Case 1 (Table 7 in Appendix F.1), with Modulo, the query sequence shrinks from 9 to 7 operations, and the matched demonstration is another remainder problem; without Modulo, the inflated division-multiplication-subtraction sequence attracts a proportion problem with an incompatible reasoning pattern. In Case 2 (Table 8 in Appendix F.1), with Algebra, the sequence shrinks from 13 to 7 operations; the compact Define–Algebra pattern matches another equation-solving demonstration, while the 13-step QDMR-only decomposition attracts a multi-item subtraction problem.

Cases 3–4: DTW Overcomes Limitations of Prefix Matching Cases 3–4 use only the 13 QDMR operations to isolate Module 2. Case 3 (Table 9 in Appendix F.2) shows that DTW selects a percentage-markup demonstration that matches the query's reasoning pattern despite different third operations, while the longest valid prefix demonstration shares 6 of 7 operations but encodes a subtract-discount pattern incompatible with the query's add-insurance requirement. Case 4 (Table 10 in Appendix F.2) shows that the longest valid prefix demonstration covers 5 of 7 operations but lacks any percentage step, causing the model to miss the tax computation entirely; DTW instead selects a semantically aligned percentage demonstration. Full case details with query examples and demonstration sequences are provided in Appendix F.

5 Conclusion and Future Work

We propose SALA, a reasoning-oriented demonstration selection framework that combines task-

adaptive operation construction with semantic DTW-based alignment. SALA represents problem-solving logic as explicit reasoning-operation sequences, while allowing flexible matching in semantic space. Experiments on four reasoning benchmarks and three LLMs show that SALA consistently outperforms strong demonstration selection baselines. In future work, we plan to extend SALA to broader reasoning domains and explore richer reasoning structures beyond linear operation sequences.

Limitations

SALA uses LLMs to induce task-adaptive operations and parse questions into reasoning-operation sequences. While this reduces the need for manual operation engineering, the induced operation set and parsed sequences may vary with the induction model and prompting strategy. In this work, we use a separate LLM and fixed prompts to keep the construction process consistent across experiments.

SALA currently represents problem-solving logic as a linear sequence of operations. This representation works well for the reasoning benchmarks studied in this paper, but more complex tasks may benefit from richer structures, such as hierarchical or graph-based reasoning representations. We leave the extension of SALA to these settings for future work.

Acknowledgements

This work was supported by the National Natural Science Foundation of China (62306344, 62276279), the Guangdong Basic and Applied Basic Research Foundation (2024A1515010253, 2026A1515011800, 2024B1515020032), the Open Research Fund of the State Key Laboratory of Blockchain and Data Security, Zhejiang University (Grant No. A2537), and the Guangdong S&T Programme Key-Area Research and Development Program of Guangdong Province (2026B0101100004).

Generative AI tools were used during the preparation and revision of this manuscript to polish the language, revise selected passages, check the consistency of terminology and reported numerical values across sections, and assist with LaTeX formatting. All AI-assisted changes incorporated into the manuscript were reviewed and verified by the authors, who take full responsibility for the paper's content.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, and 1 others. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Shengnan An, Bo Zhou, Zeqi Lin, Qiang Fu, Bei Chen, Nanning Zheng, Weizhu Chen, and Jian-Guang Lou. 2023. Skill-based few-shot selection for in-context learning. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 13472–13492, Singapore. Association for Computational Linguistics.

Aram Bahrini, Mohammadsadra Khamoshifar, Hossein Abbasimehr, Robert J. Riggs, Maryam Esmaeili, Rastin Mastali Majdabadkohne, and Morteza Pasehvar. 2023. Chatgpt: Applications, opportunities, and threats. In *2023 Systems and Information Engineering Design Symposium (SIEDS)*, pages 274–279.

Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, and 1 others. 2023. Qwen technical report. *arXiv preprint arXiv:2309.16609*.

Satwik Bhattamishra, Arkil Patel, Phil Blunsom, and Varun Kanade. 2024. Understanding in-context learning in transformers and LLMs by learning to learn discrete functions. In *The Twelfth International Conference on Learning Representations*.

ByteDance Seed Team. 2025. Seed-oss open-source models. https://github.com/ByteDance-Seed/seed-oss.

Laming Chen, Guoxin Zhang, and Eric Zhou. 2018. Fast greedy map inference for determinantal point process to improve recommendation diversity. In *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, and 1 others. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

DeepSeek-AI. 2026. Deepseek-v4: Towards highly efficient million-token context intelligence.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Qingxiu Dong, Lei Li, Damai Dai, Ce Zheng, Jingyuan Ma, Rui Li, Heming Xia, Jingjing Xu, Zhiyong Wu,

Baobao Chang, Xu Sun, Lei Li, and Zhifang Sui. 2024. A survey on in-context learning. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 1107–1128, Miami, Florida, USA. Association for Computational Linguistics.

Mor Geva, Daniel Khashabi, Elad Segal, Tushar Khot, Dan Roth, and Jonathan Berant. 2021. Did aristotle use a laptop? a question answering benchmark with implicit reasoning strategies. *Transactions of the Association for Computational Linguistics*, 9:346–361.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, and 1 others. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Xiaonan Li, Kai Lv, Hang Yan, Tianyang Lin, Wei Zhu, Yuan Ni, Guotong Xie, Xiaoling Wang, and Xipeng Qiu. 2023. Unified demonstration retriever for in-context learning. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4644–4668, Toronto, Canada. Association for Computational Linguistics.

Xiaonan Li and Xipeng Qiu. 2023. Finding support examples for in-context learning. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 6219–6235, Singapore. Association for Computational Linguistics.

Yukang Lin, Bingchen Zhong, Shuoran Jiang, Joanna Siebert, and Qingcai Chen. 2025. Reasoning graph enhanced exemplars retrieval for in-context learning. In *Proceedings of the 31st International Conference on Computational Linguistics*, pages 9737–9759, Abu Dhabi, UAE. Association for Computational Linguistics.

Jiayu Liu, Zhenya Huang, Chaokun Wang, Xunpeng Huang, ChengXiang Zhai, and Enhong Chen. 2025. What makes in-context learning effective for mathematical reasoning. In *Forty-second International Conference on Machine Learning*.

Man Luo, Xin Xu, Zhuyun Dai, Panupong Pasupat, Mehran Kazemi, Chitta Baral, Vaiva Imbrasaite, and Vincent Y Zhao. 2023. Dr.ICL: Demonstration-Retrieved In-Context Learning. In *NeurIPS 2023 Workshops: R0-FoMo*.

Man Luo, Xin Xu, Yue Liu, Panupong Pasupat, and Mehran Kazemi. 2024. In-context learning with retrieved demonstrations for language models: A survey. *Transactions on Machine Learning Research*. Survey Certification.

Xuetao Ma, Wenbin Jiang, and Hua Huang. 2025. Problem-solving logic guided curriculum in-context learning for LLMs complex reasoning. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 8394–8412, Vienna, Austria. Association for Computational Linguistics.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. 2022. Training language models to follow instructions with human feedback. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, NIPS '22, Red Hook, NY, USA. Curran Associates Inc.

Siru Ouyang, Jun Yan, I-Hung Hsu, Yanfei Chen, Ke Jiang, Zifeng Wang, Rujun Han, Long T. Le, Samira Daruki, Xiangru Tang, Vishy Tirumalashetty, George Lee, Mahsan Rofouei, Hangfei Lin, Jiawei Han, Chen-Yu Lee, and Tomas Pfister. 2025. Reasoningbank: Scaling agent self-evolving with reasoning memory. *CoRR*, abs/2509.25140.

Arkil Patel, Satwik Bhattamishra, and Navin Goyal. 2021. Are NLP models really able to solve simple math word problems? In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2080–2094, Online. Association for Computational Linguistics.

Kiran Purohit, Venktesh V, Sourangshu Bhattacharya, and Avishek Anand. 2025. Sample efficient demonstration selection for in-context learning. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pages 49959–49982. PMLR.

Chengwei Qin, Aston Zhang, Chen Chen, Anirudh Dagar, and Wenming Ye. 2024. In-context learning with iterative demonstration selection. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 7441–7455, Miami, Florida, USA. Association for Computational Linguistics.

Stephen Robertson and Hugo Zaragoza. 2009. The probabilistic relevance framework: Bm25 and beyond. *Found. Trends Inf. Retr.*, 3(4):333–389.

Ohad Rubin, Jonathan Herzig, and Jonathan Berant. 2022. Learning to retrieve prompts for in-context learning. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2655–2671, Seattle, United States. Association for Computational Linguistics.

H. Sakoe and S. Chiba. 1978. Dynamic programming algorithm optimization for spoken word recognition. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 26(1):43–49.

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. 2019. CommonsenseQA: A question answering challenge targeting commonsense knowledge. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for*

Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), pages 4149–4158, Minneapolis, Minnesota. Association for Computational Linguistics.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, and 1 others. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Xubin Wang, Jianfei Wu, Yichen Yuan, Deyu Cai, Mingzhe Li, and Weijia Jia. 2025. Demonstration selection for in-context learning via reinforcement learning. In *Proceedings of the 42nd International Conference on Machine Learning*, ICML'25. JMLR.org.

Noam Wies, Yoav Levine, and Amnon Shashua. 2023. The learnability of in-context learning. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, NIPS '23, Red Hook, NY, USA. Curran Associates Inc.

Tomer Wolfson, Mor Geva, Ankit Gupta, Matt Gardner, Yoav Goldberg, Daniel Deutch, and Jonathan Berant. 2020. Break it down: A question understanding benchmark. *Transactions of the Association for Computational Linguistics*, 8:183–198.

Zifan Xu, Haozhu Wang, Dmitriy Bespalov, Xian Wu, Peter Stone, and Yanjun Qi. 2024. LaRS: Latent reasoning skills for chain-of-thought reasoning. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 3624–3643, Miami, Florida, USA. Association for Computational Linguistics.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, and 22 others. 2024. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*.

Zhao Yang, Yuanzhe Zhang, Dianbo Sui, Cao Liu, Jun Zhao, and Kang Liu. 2023. Representative demonstration selection for in-context learning with two-stage determinantal point process. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 5443–5456, Singapore. Association for Computational Linguistics.

Jiacheng Ye, Zhiyong Wu, Jiangtao Feng, Tao Yu, and Lingpeng Kong. 2023a. Compositional exemplars for in-context learning. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 39818–39833. PMLR.

Junjie Ye, Xuanting Chen, Nuo Xu, Can Zu, Zekai Shao, Shichun Liu, Yuhan Cui, Zeyang Zhou, Chao Gong, Yang Shen, Jie Zhou, Siming Chen, Tao Gui, Qi Zhang, and Xuanjing Huang. 2023b. A comprehensive capability analysis of gpt-3 and gpt-3.5 series models. *Preprint*, arXiv:2303.10420.

Jianfei Zhang, Bei Li, Jun Bai, Rumei Li, Yanmeng Wang, Chenghua Lin, and Wenge Rong. 2025a. Selecting demonstrations for many-shot in-context learning via gradient matching. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 11686–11704, Vienna, Austria. Association for Computational Linguistics.

Yufeng Zhang, Xuepeng Wang, Lingxiang Wu, and Jinqiao Wang. 2025b. Enhancing chain of thought prompting in large language models via reasoning patterns. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 25985–25993.

Zheng Zhang, Shaocheng Lan, Lei Song, Jiang Bian, Yexin Li, and Kan Ren. 2025c. Learning to select in-context demonstration preferred by large language model. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 11345–11360, Vienna, Austria. Association for Computational Linguistics.

A Dataset Statistics

Table 3 summarizes the training and test splits of the four benchmark datasets used in this paper.

Dataset	Train	Test
GSM8K	7,473	1,319
SVAMP	700	300
CommonsenseQA	9,741	1,140
StrategyQA	1,603	687

Table 3: Training and test split statistics for the datasets used in our experiments.

B Predefined QDMR Reasoning Operations

SALA extends the 13 predefined QDMR reasoning operations (Wolfson et al., 2020) with task-specific operations. Table 4 lists all 13 predefined operations with their core functions, example questions, and example sequences.

C Prompt Templates

We provide the full prompt templates used in each step of the SALA pipeline. All prompts were used in English during our experiments.

C.1 Operation Induction Prompt

This prompt instructs the LLM to induce reasoning operations not covered by the predefined 13.

Operation	Core Function	Example Question	Example Sequence
Select	Selects a specific entity or set	How many touchdowns were scored overall?	SELECT['touchdowns']; AGGREGATE['number', '#1']
Filter	Selects a subset matching conditions	I would like a flight from Toronto to San Diego please.	SELECT['flights']; FILTER['#1', 'from Toronto']; FILTER['#2', 'to San Diego']
Arithmetic	Basic arithmetic (+, −, ×, ÷) on numeric attributes	How many more red objects are there than blue objects?	SELECT['red objects']; SELECT['blue objects']; PROJECT['number', '#1']; PROJECT['number', '#2']; ARITHMETIC['difference', '#3', '#4']
Comparative	Selects subset greater/less than a threshold	Who are the authors with more than 500 papers?	SELECT['authors']; PROJECT['papers', '#1']; GROUP['number', '#2', '#1']; COMPARATIVE['#1', '#3', 'more than 500']
Superlative	Selects the entity with the extreme value	What is the keyword contained by the most papers?	SELECT['papers']; PROJECT['keywords', '#1']; GROUP['number', '#1', '#2']; SUPERLATIVE['#2', '#3', 'highest']
Aggregate	Computes mathematical properties (count, avg) of a set	How many states border Colorado?	SELECT['Colorado']; PROJECT['border states', '#1']; AGGREGATE['number', '#2']
Union	Merges two sets	Tell me the president and vice-president.	SELECT['president']; SELECT['vice-president']; UNION['#1', '#2']
Intersection	Takes the intersection of two sets	Show parties with representatives in both New York and Pennsylvania.	SELECT['representatives']; FILTER['#1', 'in New York']; FILTER['#1', 'in Pennsylvania']; INTERSECTION['parties', '#2', '#3']
Project	Obtains a specific attribute of an entity	Who is the head coach of the Los Angeles Lakers?	SELECT['Los Angeles Lakers']; PROJECT['head coach', '#1']
Sort	Arranges elements by a specified rule	Find student addresses sorted by monthly rental.	SELECT['students']; PROJECT['addresses', '#1']; PROJECT['monthly rental', '#2']; SORT['#2', '#3']
Group	Computes properties per group element	How many female students per club?	SELECT['clubs']; FILTER['#1', 'female students']; GROUP['number', '#2', '#1']
Discard	Selects subset not satisfying a condition	Find professors not playing Canoeing.	SELECT['professors']; FILTER['#1', 'playing Canoeing']; DISCARD['#1', '#2']
Boolean	Determines if entity satisfies a condition	Were Scott Derrickson and Ed Wood of the same nationality?	SELECT['Scott Derrickson']; SELECT['Ed Wood']; PROJECT['nationality', '#1']; PROJECT['nationality', '#2']; BOOLEAN['#3', 'the same as', '#4']

Table 4: The 13 predefined QDMR reasoning operations used as the base inventory in SALA.

Task: Given an input question and the 13 existing reasoning operations (listed below), induce any new reasoning operations needed to solve the problem.

Existing reasoning operations: {operations}

Requirements: Supplement with new operations. Identify any new reasoning operations from the input question that are not covered by the existing ones. Describe the core functionality, provide an example question, example decomposition, and corresponding reasoning operation sequence in the same format as above. Do not output any extraneous content. If you determine that the existing operations are sufficient to solve the input question, simply output "No new operation needed."

Input question: {problem}

C.2 Sequence Analysis Prompt

This prompt decomposes a question into an operation sequence using the extended operation set.

Decompose the input question into a reasoning operation sequence using the given reasoning operations.

Given reasoning operations: {operations}

Example Question: what flights are available tomorrow from denver to philadelphia? Reasoning operation sequence: ['SELECT['flights']', 'FILTER['#1', 'from denver']', 'FILTER['#2', 'to philadelphia']', 'FILTER['#3', 'if available']'] Operation name list: ['select', 'filter', 'filter', 'filter']

Follow the example format strictly and provide the reasoning operation sequence and corresponding operation name list for the input question below.

Input question: {question}

C.3 LLM Deduplication Prompt

This prompt checks whether a newly induced operation overlaps with existing ones (Stage 2 of deduplication).

Task: Compare whether the functionality of the new reasoning operation overlaps with any existing reasoning operation, and determine whether the new operation is a composite operation.

Existing reasoning operations: {old_batch}

Requirements: If the new operation's functionality overlaps with any existing operation, or if the new operation can be decomposed into existing operations (i.e., it is composite), output '<1>'. Otherwise, output '<0>'.

New reasoning operation: {new}

C.4 ICL Prompt Template

This template formats the retrieved demonstrations and query for the final ICL inference.

Before answering my question, there are some examples you can use for reference:

<example>{examples} </example>

Here is my question: {question}

Please answer it:

C.5 Evaluation Prompts

We use three dataset-specific evaluation prompts to compare candidate answers against ground-truth answers.

Numeric Evaluation (used for GSM8K, SVAMP):

> Your task is to compare the Candidate Answer with the Correct Answer and determine whether they are consistent.
>
> **Question:** {question}
>
> **Candidate Answer:** {candidate}
>
> **Correct Answer:** {correct}
>
> **Criteria:** - Numerical values are consistent if they represent the same quantity despite different formats (e.g., 0.88 is consistent with 88%). Return <1>. - Numerical values are also considered consistent if rounding leads to the same result (e.g., if the correct answer is 8 and the candidate is 7.96, return <1>).
>
> **Output:** Compare whether the answers are consistent. If consistent, output <1>; otherwise, output <0>.

Text Evaluation (used for CommonsenseQA):

> Your task is to compare the Candidate Answer with the Correct Answer and determine whether they are consistent.
>
> **Question:** {question}
>
> **Candidate Answer:** {candidate}
>
> **Correct Answer:** {correct}
>
> **Criteria:** - Evaluate whether the Candidate Answer and Correct Answer share the same meaning in the context of the given question. If they use different words but convey the same meaning, consider them consistent.
>
> **Output:** If consistent, output <1>; otherwise, output <0>.

Boolean Evaluation (used for StrategyQA):

> Your task is to compare the Candidate Answer with the Correct Answer and determine whether they are consistent.
>
> **Question:** {question}
>
> **Candidate Answer:** {candidate}
>
> **Correct Answer:** {correct}
>
> **Criteria:** - The Candidate Answer may be verbose while the Correct Answer is very concise. If the final conclusion of the Candidate Answer matches the boolean value (True or False) of the Correct Answer, consider them consistent.
>
> **Output:** If consistent, output <1>; otherwise, output <0>.

D Hyperparameter Settings

Table 5 lists the key hyperparameters used in our experiments.

Parameter	Value
Operation Induction	
LLM for induction & deduplication	Seed-OSS-36B-Instruct
Temperature	0.01
Max tokens	4,096
Semantic Embedding	
Embedding model	bert-base-uncased
Embedding dimension	768
Max input length	1,024 tokens
DTW Retrieval	
Retrieval top-k	8
Inference & Evaluation	
Target LLMs	Llama3-8B-Instruct, Qwen2.5-7B-Instruct, DeepSeek-V4-Pro
Temperature	0.01
Max tokens	4,096
Evaluation LLM	GPT-4

Table 5: Key hyperparameter settings for SALA.

E Induced Reasoning Operation Examples across Downstream Tasks

Table 6 presents representative reasoning operations induced by SALA on each downstream benchmark through the adaptive operation set construction process.

F Case Study — Detailed Tables

This appendix provides the full detailed tables for the four case studies summarized in Section 4.4 (Table 11). Cases 1–2 isolate Module 1 (task-specific operations) and Cases 3–4 isolate Module 2 (DTW semantic alignment). All experiments use Llama3-8B-Instruct.

F.1 Effect of Task-Specific Operations on Reasoning Sequence Compactness

Module 1 induces task-specific operations that condense reasoning sequences which would otherwise require long chains of QDMR operations. This compactness directly affects which demonstrations are retrieved and whether the model produces the correct answer.

F.1.1 Case 1: Modulo Condenses Remainder Computation

F.1.2 Case 2: Algebra Condenses Variable Solving

F.2 DTW Semantic Alignment vs. Prefix Subsequence Matching

To isolate Module 2, both methods in this subsection use **only the 13 QDMR operations**.

Operation	Core Function	Example Question	Example Sequence
Define	Declares an unknown variable representing a target quantity to be solved	Randy spent $10 on lunch. He then spent a quarter of his remaining money on an ice cream cone costing $5. What was his initial money?	DEFINE['M', 'initial money']; ARITHMETIC['subtract', '#1', '10']; ARITHMETIC['quarter', '#2']; ALGEBRA['#3', '=', '5']
Algebra	Defines unknown variables, builds relations with known values, and solves via equations	Yvonne brings a box of chocolates to school. Half have nuts and half do not. The students eat 80% of the ones with nuts and half of the ones without nuts. If there are 28 left, how many were originally in the box?	DEFINE['total chocolates', '#x']; ARITHMETIC['division', '#x', 2]; ARITHMETIC['division', '#x', 2]; ARITHMETIC['multiply', '#2', '0.2']; ARITHMETIC['multiply', '#3', '0.5']; ARITHMETIC['addition', '#4', '#5']; ALGEBRA['#x', '#6', '=', '28']
Modulo	Computes the remainder after dividing one number by another	Emma's bank account has $100. Each day of the week she spends $8. At the end of the week, she withdraws as many $5 bills as possible. How many dollars remain?	SELECT['initial amount']; SELECT['daily spending']; SELECT['days in a week']; ARITHMETIC['product', '#2', '#3']; ARITHMETIC['difference', '#1', '#4']; MODULO['#5', '5']
		SVAMP	
Constant	Obtains a fixed numeric value given directly or implicitly in the problem	Mary is baking a cake. The recipe calls for 5 cups of sugar and 14 cups of flour. She already put in 11 cups of flour. How many more cups of sugar than cups of flour does she need to add now?	SELECT['recipe']; PROJECT['sugar', '#1']; PROJECT['flour', '#1']; SELECT['flour']; PROJECT['already added', '#1']; CONSTANT['0']; ARITHMETIC['subtract', '#2', '#6']; ARITHMETIC['subtract', '#3', '#5']; ARITHMETIC['subtract', '#7', '#8']
Literal	Retrieves a specific numeric or attribute value directly stated in the problem	If 479 students suggested adding mashed potatoes while 489 suggested adding bacon, how many more students suggested bacon than mashed potatoes?	LITERAL['mashed potato', '479']; LITERAL['bacon', '489']; ARITHMETIC['difference', '#1', '#2']
Retain	Preserves the original attribute value of an entity when no change to that attribute is described	Ed had 12 more marbles than Doug. Ed lost 20 marbles. If Ed now has 17, how many marbles does Doug have now?	SELECT['Edś current marbles']; SELECT['Edś lost marbles']; ARITHMETIC['addition', '#1', '#2']; SELECT['Ed-Doug gap']; ARITHMETIC['subtract', '#3', '#4']; RETAIN['Dougś marbles', '#5']
		StrategyQA	
EntityLinking	Resolves a referring expression to its corresponding real-world entity	Was historical Dracula from a town in Bucharest?	ENTITYLINKING["historical Dracula"]; PROJECT["birthplace", "#1"]; BOOLEAN["#2", "in", "Bucharest"]
Relate	Obtains the value of a specific relationship between two or more entities	What is the distance between Dusseldorf and Stonehenge?	SELECT['Dusseldorf']; SELECT['Stonehenge']; RELATE['distance', '#1', '#2']
Deductive	Derives a conclusion through logical reasoning from multiple factual premises	Aristotle died in 322 BC. The Model Parliament was held in 1295. The House of Lords grew out of the Model Parliament. Was Aristotle a member of the House of Lords?	SELECT['Aristotle']; PROJECT['death year', '#1']; SELECT['Model Parliament']; PROJECT['held year', '#3']; SELECT['House of Lords']; PROJECT['origin', '#5']; BOOLEAN['#2', 'earlier than', '#4']; DEDUCTIVE['#1 is a member of #5', '#6', '#7']
		CommonsenseQA	
Contrast	Obtains the opposite or contrasting concept of a given concept or attribute	The troublemaker had been hoping for a soft punishment, but the ruling handed down was quite what?	SELECT['soft punishment']; CONTRAST['opposite', '#1']; SELECT['the ruling']; PROJECT['#2', '#3']
Similar	Finds entities or collections that are similar in type or features to a specified entity	A hurricane is similar to what other wind event?	SELECT['hurricane']; SIMILAR['wind events', '#1']
Source	Locates where or how a specific attribute of an entity can be found or accessed	If I wanted to find out the hours that Marmot was open, I might look where?	SELECT['Marmot']; SOURCE['open hours', '#1']

Table 6: Representative reasoning operations induced by SALA on four downstream benchmarks.

F.2.1 Case 3: Granularity Mismatch — Best Structural Match Excluded by Prefix Constraint

F.2.2 Case 4: Incomplete Reasoning — Longest Valid Prefix Omits Percentage Computation

Table 11 summarizes the four cases.

Query	
Emma has $100 in her bank account. She spends $8 each day for a full week. At the end of the week, she withdraws as many $5 bills as possible. How many dollars remain in her account?	
Ground Truth	$4
w/ new OPs	**$4** ✓ (with Modulo)
w/o new OPs	**$2** × (QDMR only)
With New Operations (Modulo)	
Query sequence (len=7)	[SELECT, PROJECT, SELECT, PROJECT, ARITHMETIC, ARITHMETIC, MODULO]
Matched demo	"A factory packs 125 toys per day. Each box holds 8 toys. After filling full boxes, how many toys are left?"
Demo sequence (len=4)	[SELECT, PROJECT, ARITHMETIC, MODULO]
Analysis	This is another remainder problem; the model correctly performs the modulo computation.
Without New Operations (QDMR Only)	
Query sequence (len=9)	[SELECT, PROJECT, SELECT, PROJECT, ARITHMETIC, ARITHMETIC, ARITHMETIC, ARITHMETIC, ARITHMETIC] To simulate modulo, the sequence appends division → multiplication → subtraction.
Matched demo	"A recipe needs 3 cups of flour per 5 servings. How many cups for 20 servings?"
Demo sequence (len=7)	[SELECT, PROJECT, SELECT, PROJECT, ARITHMETIC, ARITHMETIC, ARITHMETIC]
Analysis	This is a proportional reasoning problem, not a remainder problem. The model is misled by the similar tail of consecutive ARITHMETIC steps (5 in the query, 3 in the demo).

Table 7: Case 1: With Modulo, the compact sequence attracts another remainder problem. Without it, the inflated division-multiplication-subtraction chain attracts a proportion problem that shares the same surface pattern but has a different reasoning structure.

Query	
Yvonne brings a box of chocolates to school. Half have nuts and half do not. The students eat 80% of the ones with nuts and half of the ones without nuts. If there are 28 chocolates left, how many chocolates were originally in the box?	
Ground Truth	80
w/ new OPs	**80** ✓ (with Algebra)
w/o new OPs	**90** × (QDMR only)
With New Operations (Algebra)	
Query sequence (len=7)	[DEFINE, ARITHMETIC, ARITHMETIC, ARITHMETIC, ARITHMETIC, ARITHMETIC, ALGEBRA]
Matched demo	"Randy spent $10 on lunch and a quarter of the remaining money on an ice cream cone costing $5. What was his initial money?"
Demo sequence (len=5)	[DEFINE, ARITHMETIC, ARITHMETIC, ARITHMETIC, ALGEBRA]
Analysis	Both queries define an unknown variable and solve for it via an equation. The model recovers the algebraic structure and correctly outputs 80.
Without New Operations (QDMR Only)	
Query sequence (len=13)	[SELECT, SELECT, PROJECT, PROJECT, ARITHMETIC, ARITHMETIC, PROJECT, PROJECT, ARITHMETIC, ARITHMETIC, ARITHMETIC, ARITHMETIC, ARITHMETIC] Without DEFINE and ALGEBRA, each chocolate category must be separately selected and projected; remaining amounts are computed stepwise.
Matched demo	"Danny has 94 guppies, 76 angelfish, 89 tiger sharks, and 58 Oscar fish. If he sells 30, 48, 17, and 24 respectively, how many fish remain?"
Demo sequence	[SELECT, PROJECT, SELECT, PROJECT, SELECT, PROJECT, SELECT, PROJECT, ARITHMETIC, ...]
Analysis	This is a multi-item subtraction problem whose SELECT–PROJECT repetition is structurally similar to the inflated query, but its reasoning logic (per-item subtraction then aggregation) is unrelated to variable solving.

Table 8: Case 2: With Algebra, the compact DEFINE–...–ALGEBRA pattern matches equation-solving demonstrations. QDMR-only decomposition produces a long chain of SELECT–PROJECT pairs that attracts structurally similar but logically unrelated multi-item problems.

Query	
Janet buys a brooch for her daughter. She pays $500 for the material and another $800 for the jeweler to construct it. After that, she pays 10% of that to get it insured. How much did she pay?	
Query sequence (len=7)	[SELECT, PROJECT, SELECT, PROJECT, ARITHMETIC, ARITHMETIC, ARITHMETIC]
Ground Truth	$1,430
DTW	**$1,430** ✓
Prefix	**$1,170** ×
DTW-Selected Demonstration	
Demo	"John commissions a drawing. A black and white costs $160. Color is 50% more. How much for both?"
Demo sequence (len=5)	[SELECT, PROJECT, ARITHMETIC, ARITHMETIC, ARITHMETIC]
Reasoning	Extract base cost → apply percentage markup → sum.
Prefix match	**Not a valid prefix.** c_3 = ARITHMETIC $\neq$ q_3 = SELECT, so the demo's entire sequence does not match the query's prefix. Excluded.
Prefix-Selected Demonstration (longest valid prefix = 6)	
Demo	"A book costs $20 and a notebook costs $5. With a 10% total discount, final price?"
Demo sequence (len=6)	[SELECT, PROJECT, SELECT, PROJECT, ARITHMETIC, ARITHMETIC]
Reasoning	Sum two items → apply percentage → **subtract** (discount).
Prefix match	Entire demo sequence $c_{1:6}$ = $q_{1:6}$ ✓. Valid prefix.
Analysis	
	The DTW-selected demo shares the query's "base cost → percentage" reasoning but is excluded by prefix matching because $c_3 \neq q_3$. The prefix-selected demo matches the first 6 operations exactly, but its final reasoning step is "subtract discount" while the query requires "add insurance." The model, misled by the subtraction pattern, outputs $1,170 ($1,300 − $130) instead of $1,430 ($1,300 + $130).

Table 9: Case 3: DTW selects a structurally aligned demo (single-item percentage markup) despite being excluded by prefix matching due to a granularity mismatch at position 3. Prefix matching instead selects a two-item discount demo whose entire sequence matches the query's prefix, but whose final reasoning step (subtract discount) contradicts the query's requirement (add insurance).

Query	
Alice bought a $50 dress and a $30 bag. The store applies 8% sales tax. What is the total?	
Query sequence (len=7)	[SELECT, PROJECT, SELECT, PROJECT, ARITHMETIC, ARITHMETIC, ARITHMETIC]
Ground Truth	$86.40
DTW	**$86.40** ✓
Prefix	**$80** ×
DTW-Selected Demonstration	
Demo	"John commissions a drawing. A black and white costs $160. Color is 50% more. How much for both?"
Demo sequence (len=5)	[SELECT, PROJECT, ARITHMETIC, ARITHMETIC, ARITHMETIC]
Reasoning	Extract base cost → apply percentage → sum.
Prefix match	**Not a valid prefix.** c_3 = ARITHMETIC $\neq$ q_3 = SELECT. Excluded.
Prefix-Selected Demonstration (longest valid prefix = 5)	
Demo	"A Toyota costs $20,000 and a Honda costs $25,000. What is the total cost?"
Demo sequence (len=5)	[SELECT, PROJECT, SELECT, PROJECT, ARITHMETIC]
Reasoning	Extract two item costs → sum them.
Prefix match	Entire demo sequence $c_{1:5}$ = $q_{1:5}$ ✓. Valid prefix.
Analysis	
	The DTW-selected demo teaches the model to apply a percentage after obtaining a base value, which matches the query's "add tax" logic. It is excluded by prefix matching because $c_3 \neq q_3$. The prefix-selected demo is a valid prefix covering 5 of 7 operations, but its reasoning stops at "sum two costs" with no percentage step. The model follows this incomplete pattern, outputs $80 (the pre-tax subtotal), and misses the 8% tax entirely.

Table 10: Case 4: DTW selects a demonstration whose reasoning pattern (base value → percentage → total) matches the query, despite not being a valid prefix. Prefix matching selects a structurally simpler demo that is a valid prefix, but whose reasoning stops at "sum two costs" without any percentage computation, causing the model to omit the tax step.

Case	What is compared	Key insight
1	13 QDMR vs. + Modulo	Without Modulo, remainder is simulated via 3 extra ARITHMETIC steps, attracting proportion problems instead of remainder problems.
2	13 QDMR vs. + Algebra	Without Algebra, variable solving expands to 13-step SELECT–PROJECT chains, attracting multi-item subtraction problems.
3	Prefix vs. DTW	Prefix matching excludes the best structural match because $c_3 \neq q_3$; DTW spans the granularity gap. The longest valid prefix demo shares 6 of 7 operations but ends with "subtract" instead of "add," misleading the model.
4	Prefix vs. DTW	The longest valid prefix demo matches 5 of the query's 7 operations but has no percentage step, causing the model to miss the tax computation. DTW selects a semantically aligned demo ($c_3 \neq q_3$) that includes percentage reasoning.

Table 11: Summary of the four case studies and their key insights.