Process In-Context Learning: Enhancing Mathematical Reasoning via Dynamic Demonstration Insertion

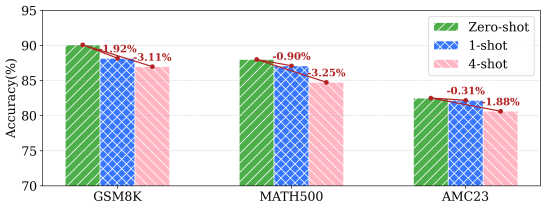
Anonymous ACL submission

Abstract

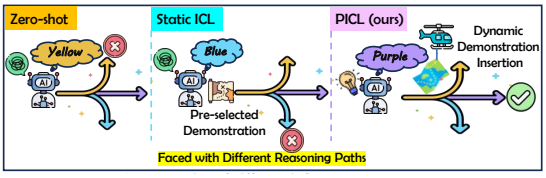
In-context learning (ICL) has proven highly effective across diverse large language model (LLM) tasks. However, its potential for enhancing tasks that demand step-by-step logical deduction, such as mathematical reasoning, remains underexplored. A core limitation of existing ICL approaches is their static use of demonstrations: examples are pre-selected before inference and remain fixed, failing to adapt to the dynamic confusion points that often arise during multi-step reasoning such as ambiguous calculations or logical gaps. These unresolved confusion points can lead to cascading errors that degrade final accuracy. To tackle this issue, we propose Process In-Context Learning (PICL), a dynamic demonstration integration framework designed to boost mathematical reasoning by responding to real-time inference needs. PICL operates in two stages: 1) it identifies potential confusion points by analyzing semantics and entropy in the reasoning process and summarizes their core characteristics; 2) upon encountering these points, it retrieves relevant demonstrations from the demonstration pool that match the confusion context and inserts them directly into the ongoing reasoning process to guide subsequent steps. Experiments show that PICL outperforms baseline methods by mitigating mid-inference confusion, highlighting the value of adaptive demonstration insertion in complex mathematical reasoning.

1 Introduction

Mathematical reasoning, a cornerstone of AI intelligence assessment, requires step-by-step logical deduction and remains challenging for even state-of-the-art models (Zhang et al., 2019; Liu et al., 2025). In-context learning (ICL) has emerged as a pivotal paradigm to address this, enabling models to improve reasoning via task-related demonstrations without parameter updates (Dong et al., 2024). However, recent studies have revealed that ICL's effectiveness in mathematical reasoning is not guaranteed. Simply prepending a set of demonstrations before inference can even yield negative effects, especially in multi-step logical tasks (Liu et al., 2025; Zheng et al., 2025; Cheng et al., 2025).



(a) Accuracy comparison of zero-shot and few-shot.



(b) Illustration of different inference methods.

Figure 1: Performance of static ICL and the conceptual framework of PICL. (a) Performance comparison of Deepseek-R1-Distilled-Qwen-7B under zero-shot, 1-shot and 4-shot ICL settings on three math benchmarks. (b) A schematic illustration comparing three inference paradigms when encountering a reasoning path selection challenge. Left: Zero-shot prompting. Middle: Conventional static few-shot ICL. Right: Our proposed PICL framework, which inserts targeted demonstrations to provide adaptive guidance.

Existing ICL methods pre-select demonstrations upfront and fix them throughout inference (Wies et al., 2023; Min et al., 2022), but such static integration fails to align with the dynamic nature of mathematical reasoning. In multi-step deduction, models often hit "confusion points" such as hesitating between calculation paths or misidentifying logical transitions (Tang et al., 2025). Static demonstrations not only fail to resolve these but may exacerbate errors by introducing irrelevant reasoning patterns, leading to cascading mistakes in subsequent steps (Liu et al., 2025). Worse, even

advanced demonstration selection strategies such as focusing on similarity or influence overlook in-process reasoning dynamics, making them ineffective or counterproductive in complex mathematical tasks (Cheng et al., 2025; Zheng et al., 2025). Our experiments reinforce this observation: static ICL approaches offer limited benefits for mathematical reasoning and lead to performance degradation in many cases. As shown in Fig. 1a, static few-shot demonstrations fail to outperform the zero-shot baseline, and simply increasing their number can further degrade performance, underscoring the fundamental limitations of this fixed approach. This highlights an urgent need. Instead of fixed pre-inference demonstrations, we must inject targeted examples during the reasoning process to provide adaptive guidance when models need it most.

To bridge this gap, we propose PICL (Process In-Context Learning), a novel method whose advantages over static methods are illustrated in Fig. 1b. This figure depicts a common failure mode in multi-step reasoning: a "confusion point" where the model must select the correct logical path. As shown, traditional approaches such as zero-shot learning and static ICL struggle to handle this scenario: the former lacks guidance entirely, while the latter's fixed demonstrations, which lack targeting, prove unhelpful. Both ultimately lead the model down an erroneous reasoning path. In contrast, PICL operates differently: it actively detects points of uncertainty in the reasoning process and dynamically intervenes by injecting highly relevant examples. This adaptive demonstration insertion provides the precise guidance needed to resolve confusion, redirecting the model back to the correct reasoning path.

PICL consists of two core stages, tightly coupled with the inference process: First, the *Confusion Point Detection and Summarization* stage: During multi-step reasoning, PICL detects the model's intermediate outputs and identifies "confusion points" using semantics and entropy indicators of prediction uncertainty (Wang et al., 2025). It then summarizes the core characteristics of these confusion points (such as ambiguities in algebraic transformation or misalignments of logical premises) to determine the specific guidance required. Second, the *Confusion-Based Demonstration Selection & Insertion* stage: Based on the summarized characteristics of the confusion points, PICL retrieves the most relevant demonstrations from a pre-constructed example pool. These demonstrations are designed to address the identified issues (such as resolving the same algebraic ambiguity) and are inserted directly into the ongoing reasoning process. This provides targeted guidance to correct the model's reasoning trajectory and alleviate confusion. In summary, our work makes three key contributions:

- **A novel dynamic demonstration insertion paradigm:** We move beyond static ICL by introducing process-aware demonstration adaptation, enabling real-time guidance tailored to the model's in-process confusion point and addressing the rigidity of existing methods.
- **A confusion point detection mechanism:** We propose using semantics and entropy to detect reasoning uncertainty and summarize confusion characteristics, providing a concrete basis for targeted demonstration retrieval.
- **Empirical validation across benchmarks:** Experiments on mathematical reasoning datasets show PICL outperforms state-of-the-art static ICL methods, with significant gains in multi-step tasks where confusion points are most prevalent.

2 Related Work

In-Context Learning. In-context learning (ICL) enables LLMs to perform tasks without parameter updates by conditioning on demonstrations (Brown et al., 2020). Prior work studies the impact of scale, demonstration format, and data distribution (Dong et al., 2024; Chen et al., 2021; Min et al., 2022; Wei et al., 2022). Demonstration selection is critical for ICL: unsupervised methods use criteria like similarity, diversity and uncertainty (Liu et al., 2022; Levy et al., 2023; Wu et al., 2023; Nguyen and Wong, 2023), while supervised methods rely on learning-based policies (Rubin et al., 2022; Ye et al., 2023; Zhang et al., 2022; Wang et al., 2024a; Zhang et al., 2025). However, most demonstration selection methods adopt a static paradigm, using the same exemplars throughout the process, which can limit adaptability in complex reasoning.

Mathematical Reasoning Enhancement. Math reasoning is difficult for LLMs due to multi-step inference and numerical precision. Recent methods improve performance via prompting and augmentation. Structured prompting uses Chain-of-Thought to guide reasoning (Yao et al., 2023). Code-based

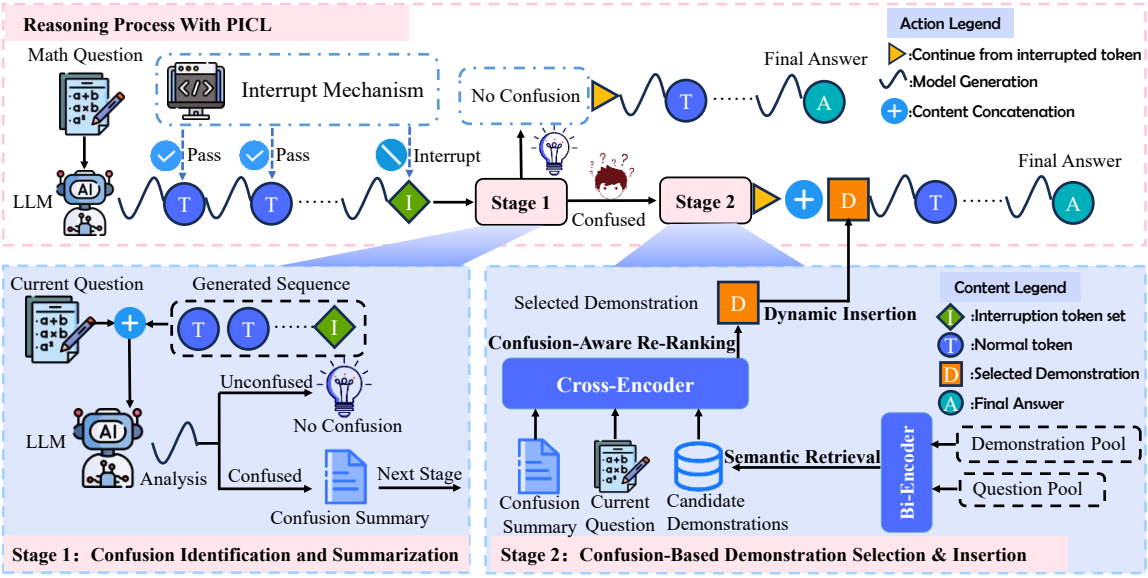


Figure 2: Process In-Context Learning (PICL) for enhancing mathematical reasoning of language models. It features two stages: an Interrupt Mechanism detects confusion via semantics and entropy in reasoning. Upon detection, relevant demonstrations are dynamically retrieved from a pool and inserted, refining reasoning process.

method alleviate errors by prompting code generations (Gao et al., 2023; Chen et al., 2021). Others augment models with retrieval, web search, tools (Lewis et al., 2020; Nakano et al., 2021; Schick et al., 2023; Li et al., 2025), and math-focused fine-tuning with process supervision (Yu et al., 2023; Lewkowycz et al., 2022; Cobbe et al., 2021b; Lightman et al., 2023). Our work targets ICL for better mathematical reasoning support.

ICL for Mathematical Reasoning. Most mathematical ICL methods remains static: exemplars are chosen beforehand using criteria such as influence functions (Guo et al., 2024), mutual information (Sorensen et al., 2022), perplexity/entropy (Gonen et al., 2023; Lu et al., 2022), or semantic similarity via BM25/BERT/BGE-M3 (Robertson et al., 2009; Devlin et al., 2019; Chen et al., 2024a). However, recent evidence suggests static ICL can be ineffective or even harmful for math reasoning (Cheng et al., 2025; Zheng et al., 2025). Motivated by these limitations, we propose PICL, a dynamic framework that inserts demonstrations at uncertainty moments during reasoning, enabling more targeted and adaptive problem solving.

3 Problem Formulation

In-context learning (ICL) is a powerful paradigm that helps LLMs solve new problems by conditioning on demonstrations. This avoids the need for parameter fine-tuning.

The conventional approach, which we term static ICL, operates in a single, fixed pass. Given a new query q and a demonstration pool $\mathcal{D} = \{d_1, \ldots, d_N\}$, a set of k demonstrations is selected before the model begins its reasoning process. Each demonstration d_i typically consists of an example problem x_i and its corresponding solution or reasoning process y_i, i.e., $d_i = (x_i, y_i)$. This selection is usually based on the metrics between the new query q and the problem component x_i of each demonstration. These selected demonstrations, $\{d_{s_1}, \ldots, d_{s_k}\}$, are then prepended to the query to form a single, static prompt:

$$\text{Prompt}_{\text{static}} = d_{s_1} \circ d_{s_2} \circ \cdots \circ d_{s_k} \circ q.$$

Here, $\circ$ denotes the concatenation operator that sequentially appends demonstrations and the query to form a single prompt. The model M conditions on this entire prompt to generate the answer in one uninterrupted sequence.

The defining limitation of this static paradigm is its rigidity, which is especially acute in tasks requiring complex, multi-step reasoning such as mathematics. This motivates our work on PICL, a dynamic framework that operates on the same demonstration pool $\mathcal{D}$. However, instead of statically pre-selected demonstrations, PICL dynamically inserts demonstrations $\{d_{s_1}, \ldots, d_{s_k}\}$ into the model's reasoning process precisely when and where confusion is detected.

4 PICL: The Proposed Method

We propose Process In-Context Learning (PICL), a novel framework designed to enhance the mathematical reasoning capabilities of language models by dynamically inserting demonstrations during the inference process.

4.1 Method Overview

As illustrated in Fig. 2, PICL operates in two primary stages: *Stage 1: Confusion Point Identification and Summarization*, where it identifies potential confusion points by analyzing semantics and entropy during the reasoning process and summarizes the context of this confusion; and *Stage 2: Confusion-Based Demonstration Selection & Insertion*, where upon encountering these points, PICL retrieves relevant demonstrations from a pre-constructed demonstration pool, tailored to address the specific confusion, and inserts them into the reasoning process to improve performance on mathematical tasks. We detail these two stages in the following subsections.

4.2 Confusion Identification and Summarization

The first stage of PICL is to detect moments of uncertainty in the model's reasoning process. This is achieved by identifying confusion points, which are instances where the model exhibits signals of hesitation or deviates from a confident reasoning trajectory.

Our detection starts from an Interrupt Mechanism, which is triggered by a predefined set of interruption tokens, denoted as $\mathcal{V}_{\text{int}}$. The selection of this vocabulary is based on a dual criterion. First, from a **semantic** perspective, we deliberately choose tokens that linguistically signal reflection, transition, or hesitation. Words such as "wait" or "maybe" are not arbitrary; they explicitly mark moments where a reasoning process is being re-evaluated or redirected. By monitoring for these specific tokens, we target junctures that are most likely to represent complex reasoning steps or potential points of confusion.

Second, this semantic intuition is empirically substantiated by a key statistical property: these chosen tokens consistently exhibit high **entropy**. At each step t when the model is about to generate the next token, it first computes a logit vector $\boldsymbol{l}_t \in \mathbb{R}^{|\mathcal{V}|}$ over its vocabulary $\mathcal{V}$. This vector is then transformed into a probability distribution $\boldsymbol{p}_t$ via the softmax function. The model's local uncertainty at this specific step is then quantified by the Shannon entropy of this probability distribution:

$$H(\boldsymbol{p}_t) = -\sum_{v \in \mathcal{V}} p_t(v) \log p_t(v).$$

A high value of $H(\boldsymbol{p}_t)$ indicates that the model's prediction is diffuse across many tokens, indicating greater uncertainty.

When the model generates a token $y_t \in \mathcal{V}_{int}$, the generation process is temporarily paused. Following this interruption, PICL initiates a self-reflection step to determine if a meaningful confusion exists. The model is prompted to analyze its preceding output $Y_{<t}$ in the context of the query q. If and only if a specific point of confusion is identified, the model is tasked with generating a concise summary of the issue. We formalize this process as:

$$\mathcal{C} = M(\text{Prompt}_{\text{detect}}, q, Y_{<t}),$$

where M is the large language model itself, and $\text{Prompt}_{\text{detect}}$ is a specialized instruction that guides the model to articulate its confusion. The resulting confusion summary, $\mathcal{C}$, might describe issues like "ambiguity in applying a specific formula" or "uncertainty about the next logical step." If no confusion is found, $\mathcal{C}$ is considered empty, and no further action is taken. This step identifies the points of confusion in the model's reasoning process, specifies the locations for dynamic demonstration insertion, and generates a confusion summary to facilitate the selection of demonstrations.

4.3 Confusion-Based Demonstration Selection & Insertion

Once a confusion summary $\mathcal{C}$ is generated, PICL initiates a three-phase process to select the most relevant demonstrations to resolve the identified issue. This process leverages the query, the confusion summary, and the candidate demonstrations.

Phase 1: Semantic Retrieval. Given the current question q and a demonstration pool $\mathcal{D} = \{d_1, \ldots, d_N\}$, the initial stage aims to efficiently narrow down the vast demonstration pool $\mathcal{D}$ to a smaller set of semantically relevant candidate demonstrations. We employ a computationally efficient Bi-Encoder architecture for this task. The problem query q and each demonstration $d_i \in \mathcal{D}$ are independently encoded into dense vector representations using a pre-trained sentence transformer like BERT:

$$\boldsymbol{h}_q = f_{\text{bi-enc}}(q), \quad \boldsymbol{h}_{d_i} = f_{\text{bi-enc}}(d_i).$$

Algorithm 1 Process In-Context Learning (PICL)

Input: Large Language model M, query q, demonstration pool $\mathcal{D}$.
Parameter: Interruption tokens $\mathcal{V}_{\text{int}}$, insertion count k, max interventions r, retrieval candidates N.
Output: The final answer A.

Initialize $Y \leftarrow \emptyset$, $intervention_cnt \leftarrow 0$
while **not** `IsEndOfSequence`(Y) **and** $intervention_cnt < r$ **do**
  $y_{\text{next}} \leftarrow$ `GenerateNextToken`(M, q, Y)
  if $y_{\text{next}} \in \mathcal{V}_{\text{int}}$ **then**
    $intervention_cnt \leftarrow intervention_cnt + 1$
    $\mathcal{C} \leftarrow$ `GenerateConfusionSummary`(M, q, Y)
    if $\mathcal{C}$ is not empty **then**
      $\mathcal{D}_N \leftarrow$ `RetrieveCandidates`$(\mathcal{D}, q, N)$
      $\mathcal{D}_{\text{top-k}} \leftarrow$ `RerankCandidates`$(\mathcal{D}_N, q, \mathcal{C}, k)$
      $Y \leftarrow Y \oplus \mathcal{D}_{\text{top-k}}$ {Insert demonstrations}
    else
      $Y \leftarrow Y \oplus y_{\text{next}}$
      {No confusion, continue normally}
    end if
  else
    $Y \leftarrow Y \oplus y_{\text{next}}$
  end if
end while
while not `IsEndOfSequence`(Y) **do**
  $y_{\text{next}} \leftarrow$ `GenerateNextToken`(M, q, Y)
  $Y \leftarrow Y \oplus y_{\text{next}}$
end while
$A \leftarrow$ `ExtractAnswerFrom`(Y)
return A

We then compute the cosine similarity between the query vector $\boldsymbol{h}_q$ and each demonstration vector $\boldsymbol{h}_{d_i}$:

$$\text{sim}(q, d_i) = \frac{\boldsymbol{h}_q \cdot \boldsymbol{h}_{d_i}}{\|\boldsymbol{h}_q\| \|\boldsymbol{h}_{d_i}\|}.$$

The top-N demonstrations with the highest similarity scores are selected to form a candidate set $\mathcal{D}_{\mathcal{N}}$, reducing the search space for the next stage.

Phase 2: Confusion-Aware Re-Ranking. The second stage performs a fine-grained re-ranking of the candidate demonstrations in $\mathcal{D}_{\mathcal{N}}$, this time explicitly incorporating the confusion summary $\mathcal{C}$. We utilize a more powerful cross-encoder model, specifically the BGEM3-Reranker (Chen et al., 2024b), which can capture deeper contextual interactions. For each candidate demonstration $d_i \in \mathcal{D}_{\mathcal{N}}$, we construct a composite input by concatenating the original query q, the confusion summary $\mathcal{C}$, and the demonstration d_i:

$$\text{Input}_i = \texttt{[CLS]}\, q\, \texttt{[SEP]}\, d_i\, \texttt{[SEP]}\, C\, \texttt{[SEP]}.$$

This combined input is fed into the cross-encoder, which outputs a relevance score s_i that quantifies the relevance of the demonstration d_i with respect to the query q and the confusion $\mathcal{C}$:

$$s_i = f_{\text{cross-enc}}(\text{Input}_i).$$

The candidate demonstrations are then re-ranked based on these scores in descending order.

Phase 3: Demonstration Insertion. Following the confusion-aware re-ranking, the final step is to dynamically augment the model's reasoning context. From the re-ranked list, we select the top-k demonstrations, where k is a tunable hyperparameter that controls the amount of guidance inserted at each intervention.

These top-k selected demonstrations, denoted as $d_{s_1}, \ldots, d_{s_k}$, are then injected directly into the inference stream at the point of interruption. This alters the context for subsequent token generation. The modified generation path can be visualized as:

$$q \rightarrow y_1, \ldots, y_{t-1}, \underbrace{d_{s_1}, \ldots, d_{s_k}}_{\text{inserted}}, y_t,$$

where $y_1, \ldots, y_{t-1}$ represents the sequence generated before the pause, and y_t is the next token generated by the model, now conditioned on the newly provided examples. This targeted intervention provides immediate guidance to help the model navigate the specific point of confusion and correct its reasoning trajectory.

To ensure the process remains computationally efficient and does not excessively disrupt the reasoning flow, we introduce a crucial control mechanism. We define a hyperparameter, r, which sets the maximum number of times the confusion detection and insertion cycle can be triggered for a single problem-solving instance. This cap prevents the model from entering protracted loops of self-correction and ensures that PICL provides targeted assistance without incurring prohibitive computational overhead. This entire iterative process of confusion detection, retrieval, re-ranking, and insertion, which can be repeated up to r times, is formally detailed in Algorithm 1. We also provide a case study in Appendix A to illustrate PICL.

5 Experiments

This section evaluates the performance of PICL and other ICL approaches across multiple mathematical datasets. Our code is available at https://anonymous.4open.science/r/PICL-EB65.

5.1 Experimental Settings

5.1.1 Datasets.

We conducted experiments on four mainstream datasets covering problems of varying difficulty levels and types. The selected datasets include

Method	*Deepseek-R1-Distilled-Qwen-7B*					*Deepseek-R1-Distilled-Llama-8B*				
	GSM8K	MATH500	AMC23	AIME24	Avg.	GSM8K	MATH500	AMC23	AIME24	Avg.
Zero-shot	90.1	88.0	82.5	50.0	77.7	75.5	66.4	82.5	46.7	67.8
Random	89.8	87.4	82.5	46.7	76.6	73.6	66.0	72.5	36.7	62.2
Best-validate	80.5	88.4	72.5	46.7	72.0	73.4	65.2	82.5	40.0	65.3
Similarity	88.4	87.4	77.5	40.0	73.3	76.0	65.4	82.5	43.3	66.8
BM25	89.7	85.0	87.5	46.7	77.2	75.2	67.2	72.5	33.3	62.1
BGEM3	88.3	87.0	85.0	50.0	77.6	73.7	67.6	75.0	46.7	65.8
Perplexity	89.4	86.2	87.5	40.0	75.8	73.4	67.4	80.0	36.7	64.4
Influence	88.5	88.4	82.5	40.0	74.8	74.9	68.0	80.0	46.7	67.4
LMS3	90.8	87.0	82.5	43.3	75.9	75.4	66.6	85.0	46.7	68.4
PICL (Ours)	**91.7**	**90.2**	**90.0**	**60.0**	**83.0**	**76.9**	**70.6**	**90.0**	**53.3**	**72.7**

Table 1: Accuracy (%) comparison on four datasets across different backbones. Avg. is computed over GSM8K, MATH500, AMC23, and AIME24. All few-shot baselines utilize one-shot reasoning. For PICL, parameters are set to $k = 1$ and $r = 1$. The best and runner-up methods are highlighted in bold and underlined, respectively.

Stage 1	Stage 2	GSM8K	MATH500	AMC23	AIME24
Deepseek-R1-Distilled-Qwen-7B					
✗	✓	91.1	89.0	82.5	53.3
✓	✗	90.9	87.8	82.5	50.0
✓	✓	**91.7**	**90.2**	**90.0**	**60.0**
Deepseek-R1-Distilled-Llama-8B					
✗	✓	75.8	64.2	82.5	43.3
✓	✗	76.6	64.6	80.0	50.0
✓	✓	**76.9**	**70.6**	**90.0**	**53.3**

Table 2: Ablation studies of two core stages in PICL.

GSM8K (Cobbe et al., 2021a), MATH500 (Lightman et al., 2023), AMC23 (MAA, 2023) and AIME24 (MAA, 2024), which span elementary to competition-level difficulty. We further conducted experiments on multiple-choice benchmarks and broader NLP benchmarks beyond mathematics. The corresponding results are reported in Appendix B and Appendix C, respectively.

5.1.2 Models.

We adopt the open-source DeepSeek-R1 reasoning model series as the backbone. The experiments are carried out on Deepseek-R1-Distilled-Qwen-7B and Deepseek-R1-Distilled-Llama-8B (DeepSeek-AI, 2025). To further demonstrate the generalizability of our method, we additionally report results on models beyond the DeepSeek series, including Qwen3-8B (Yang et al., 2025) and Llama3-8B (Meta, 2024), in the appendix D.

5.1.3 Baselines.

We compare our approach with traditional ICL demonstration selection methods and select several representative methods as baselines:

- **Zero-shot** makes predictions directly from the input without any demonstrations.
- **Random** serves as a baseline by randomly selecting the demonstrations.
- **Best-validate** selects the demonstration with the highest accuracy, which is evaluated on a validation set.
- **BM25** (Robertson et al., 2009) uses BM25 retrieval model to select demostrations.
- **Similarity** (Liu et al., 2022) selects demonstrations based on their semantic similarity to the current query.
- **BGEM3** (Chen et al., 2024a) uses a unified embedding space integrated by multiple information retrieval capabilities.
- **Perplexity** (Gonen et al., 2023) selects demonstrations based on the calculated perplexity of each demonstration.
- **Influence** (Nguyen and Wong, 2023) determines demonstrations by comparing validation accuracy differences between subsets with and without the demonstration.
- **LMS3** (Liu et al., 2025) balances demonstrations' semantic similarity and inference stability, incorporated with a rejection mechanism.

5.1.4 Evaluation.

For evaluation, we employ a sampling strategy with a temperature of 0.7 and top_p of 0.8. We use problem-solving accuracy as evaluation metric.

5.1.5 Implementation Details.

We evaluate the average performance of various few-shot methods with varying numbers of shots,

comparing them to the zero-shot baseline. As shown in Fig. 1a, it reveals that for traditional static ICL methods, increasing the number of examples from one-shot to four-shot leads to a decrease in average accuracy. Therefore, all traditional ICL methods are evaluated in a one-shot setting. For fairness and a direct comparison, our PICL method is also configured with parameters set to $r = 1$ and $k = 1$. This means PICL performs only a single round of confusion detection and selects at most one demonstration, making it comparable to the one-shot baseline setup.

5.2 Main Experiment

5.2.1 Main Results.

The results in Table 1 demonstrate the consistent superiority of PICL across different model architectures. On *R1-Qwen-7B* and *R1-Llama-8B*, PICL achieves average accuracies of 83.0% and 72.7%, surpassing the strongest respective baselines by significant margins of 5.3 and 4.3 percentage points.

A phenomenon observed across both backbones is the instability of conventional static ICL methods. For current reasoning models, static few-shot baselines frequently underperform the simple Zero-shot setting. This trend is particularly pronounced on the *R1-Qwen-7B* backbone, where Zero-shot surpasses nearly all static retrieval methods. This indicates that fixed or heuristically retrieved demonstrations often introduce noise or irrelevant patterns that disrupt the model's intrinsic reasoning process, making them more harmful than helpful.

In contrast, PICL demonstrates a consistent advantage over all baselines, particularly on challenging benchmarks. On AIME24, PICL establishes a clear lead, exceeding runner-up methods by 10.0 points on *R1-Qwen-7B* and 6.6 points on *R1-Llama-8B*. This consistent effectiveness across diverse backbones validates PICL's dynamic mechanism in identifying mid-inference confusion points, proving that adaptive demonstration insertion is essential for unlocking the full potential of the reasoning models on mathematical tasks.

Finally, to provide a comprehensive evaluation, we analyze the computational efficiency of PICL compared to static baselines during reasoning process. Detailed results are provided in Appendix F.

5.3 Ablation Studies

To evaluate the contributions of the core components within PICL, we conducted ablation studies.

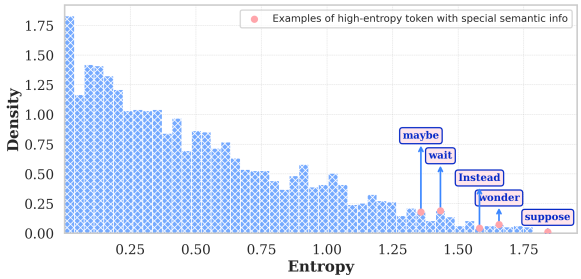


Figure 3: Density distribution of token entropy across reasoning tasks, highlighting high-entropy tokens (e.g., "maybe", "wait") that align with semantic markers of the model's uncertainty and cognitive effort.

5.3.1 Impact of Stage 1.

To examine the contribution of Stage 1, we remove the confusion identification and summarization module and instead insert a semantically retrieved demonstration at every interruption point, regardless of whether the model is actually confused. As shown in Table 2, this variant underperforms the full PICL model, which suggests that the effectiveness of demonstrations depends on being triggered at the right moments. When demonstration insertions are applied indiscriminately, the model may receive guidance even when it does not require external support, in which case the injected examples can be distracting. Therefore, Stage 1 is crucial for determining *when* to provide assistance.

5.3.2 Impact of Stage 2.

For the Stage 2 ablation, we retain Stage 1 to detect confusion points but disable the retrieval and re-ranking procedure in Stage 2, replacing it with random demonstration insertion once confusion is detected. As reported in Table 2, removing Stage 2 leads to a clear and consistent degradation across all benchmarks, with the largest drops on more challenging datasets (e.g., AMC23 and AIME24 on the *R1-Qwen-7B* setting). These results indicate that identifying confusion is not sufficient on its own. Improvements require resolving the confusion with confusion-based selected examples.

5.4 Analysis of Confusion Points and Entropy

To quantitatively identify moments of uncertainty in the model's reasoning process, we employed token entropy as our primary metric. As depicted in Fig. 3, we analyzed the entropy of over 150,000 tokens generated across various reasoning tasks. The resulting density distribution is prominently right-skewed, indicating that the model generates

the vast majority of its tokens with high certainty (low entropy). However, a critical "long tail" exists, composed of a small fraction of high-entropy tokens. These points represent rare but significant moments when the model faces substantial uncertainty in its decision-making.

To interpret these moments of statistical uncertainty, we examined their semantic content. Specifically, we analyzed an example set of "interruption tokens" that linguistically signify hesitation, reflection, or self-correction (e.g., "maybe", "wait"). The analysis revealed an alignment: these semantically significant tokens consistently fall within the high-entropy region of the distribution.

This establishes an intrinsic link between the model's statistical uncertainty (high entropy) and its linguistic expression of cognitive effort. It therefore provides an empirical validation for using these interruption tokens as reliable indicators of confusion points. We further provide a robustness analysis of the interruption-token mechanism in Appendix E.

5.5 Sensitivity Analysis of PICL

We analyze the sensitivity of PICL to two key hyperparameters: the maximum number of interventions r and the number of demonstrations inserted per intervention k, aiming to characterize their effect on accuracy and identify robust settings.

5.5.1 Impact of Max Interventions r.

Fig. 4 compares PICL with Zero-shot while varying r from 1 to 4. Across datasets, performance is generally best at small r (typically $r = 1$; occasionally comparable at $r = 2$) and degrades as r increases. In particular, results at $r = 3$ and $r = 4$ are consistently lower than those at $r \leq 2$. This suggests that a small number of targeted interventions is beneficial, whereas frequent interventions can interrupt the model's reasoning continuity and introduce additional noise, hurting final accuracy.

5.5.2 Impact of Insertion Count k.

We further vary k from 1 to 4 (Fig. 5). PICL performs best when k is small, peaking at $k = 1$. Notably, $k = 1$ achieves the strongest results on GSM8K (91.7%), MATH500 (90.2%), AMC23 (90.0%), and AIME24 (60.0%). Increasing k to 3 or 4 leads to a frequent drop across tasks. Overall, these results indicate that effective guidance is primarily driven by relevance rather than quantity: inserting one or two well-matched demonstrations is sufficient for course correction, while larger k may introduce competing patterns that dilute the most useful signal.

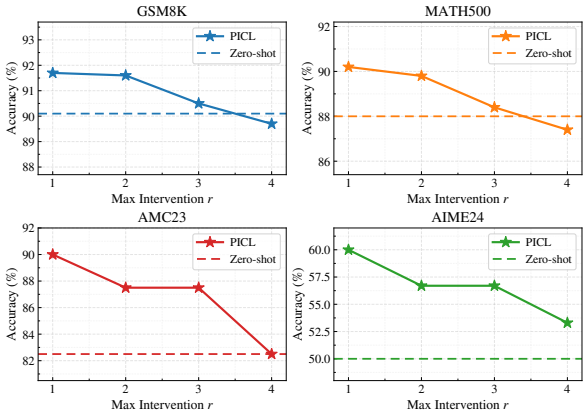


Figure 4: Performance comparison between PICL and Zero-shot with r ranging from 1 to 4.

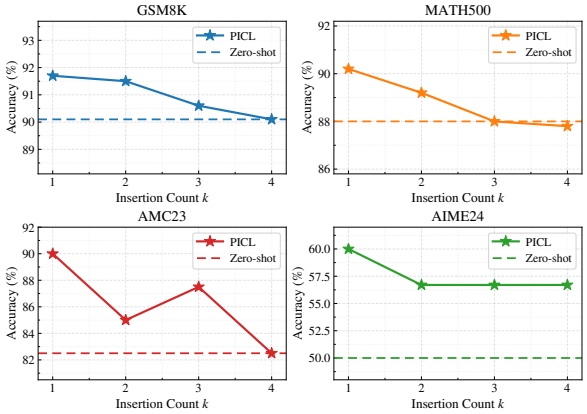


Figure 5: Performance comparison between PICL and Zero-shot with k ranging from 1 to 4.

6 Conclusion

This work addresses a key limitation of existing In-Context Learning (ICL) methods in mathematical reasoning: their static use of pre-selected demonstrations, which fail to adapt to dynamic confusion points during multi-step reasoning and often cause cascading errors. We propose Process In-Context Learning (PICL), a dynamic framework that detects confusion via semantic and entropy analysis, then retrieves and inserts relevant demonstrations to guide subsequent reasoning. Experiments show PICL outperforms baselines by mitigating mid-reasoning confusion, validating the value of adaptive demonstration insertion in complex mathematical reasoning. This work advances ICL from static to dynamic demonstration usage, with implications for other step-by-step reasoning tasks.

Limitations

Our proposed method PICL has the following limitations. First, we validate the proposed method on mathematical and commonsense reasoning benchmarks. It is still an open question whether the observed improvements will extend to more complex domains such as planning or more open-ended problem solving. Second, the method's effectiveness is tied to the quality and coverage of the example pool: when the pool is sparse, domain-mismatched, or contains noisy solutions, the retrieved demonstrations may be less helpful and can sometimes be misleading. Finally, our current strategy for addressing model "confusion" relies on an external, inference-time intervention mechanism rather than internalizing this capability within the model itself, leaving internalization through training or distillation to future work.

References

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, and 1 others. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Jianlyu Chen, Shitao Xiao, Peitian Zhang, Kun Luo, Defu Lian, and Zheng Liu. 2024a. M3-embedding: Multi-linguality, multi-functionality, multi-granularity text embeddings through self-knowledge distillation. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 2318–2335, Bangkok, Thailand. Association for Computational Linguistics.

Jianlyu Chen, Shitao Xiao, Peitian Zhang, Kun Luo, Defu Lian, and Zheng Liu. 2024b. M3-embedding: Multi-linguality, multi-functionality, multi-granularity text embeddings through self-knowledge distillation. In *Findings of the Association for Computational Linguistics ACL 2024*, pages 2318–2335.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, and 1 others. 2021. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*.

Xiang Cheng, Chengyan Pan, Minjun Zhao, Deyang Li, Fangchao Liu, Xinyu Zhang, Xiao Zhang, and Yong Liu. 2025. Revisiting chain-of-thought prompting: Zero-shot can be stronger than few-shot. *arXiv preprint arXiv:2506.14641*.

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. 2021a. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

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

DeepSeek-AI. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *Preprint*, arXiv:2501.12948.

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.

Qingxiu Dong, Lei Li, Damai Dai, Ce Zheng, Jingyuan Ma, Rui Li, Heming Xia, Jingjing Xu, Zhiyong Wu, Baobao Chang, and 1 others. 2024. A survey on in-context learning. In *Proceedings of the 2024 conference on empirical methods in natural language processing*, pages 1107–1128.

Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. 2023. Pal: Program-aided language models. In *International Conference on Machine Learning*, pages 10764–10799. PMLR.

Hila Gonen, Srini Iyer, Terra Blevins, Noah A Smith, and Luke Zettlemoyer. 2023. Demystifying prompts in language models via perplexity estimation. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 10136–10148.

Qi Guo, Leiyu Wang, Yidong Wang, Wei Ye, and Shikun Zhang. 2024. What makes a good order of examples in in-context learning. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 14892–14904.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021. Measuring massive multitask language understanding. *Proceedings of the International Conference on Learning Representations (ICLR)*.

Itay Levy, Ben Bogin, and Jonathan Berant. 2023. Diverse demonstrations improve in-context compositional generalization. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1401–1422.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, and 1 others. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances*

in neural information processing systems, 33:9459–9474.

Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, and 1 others. 2022. Solving quantitative reasoning problems with language models. *Advances in neural information processing systems*, 35:3843–3857.

Xiaoxi Li, Jiajie Jin, Guanting Dong, Hongjin Qian, Yongkang Wu, Ji-Rong Wen, Yutao Zhu, and Zhicheng Dou. 2025. Webthinker: Empowering large reasoning models with deep research capability. *arXiv preprint arXiv:2504.21776*.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2023. Let's verify step by step. In *The Twelfth International Conference on Learning Representations*.

Jiachang Liu, Dinghan Shen, Yizhe Zhang, William B Dolan, Lawrence Carin, and Weizhu Chen. 2022. What makes good in-context examples for gpt-3? In *Proceedings of Deep Learning Inside Out (DeeLIO 2022): The 3rd workshop on knowledge extraction and integration for deep learning architectures*, pages 100–114.

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*.

Yao Lu, Max Bartolo, Alastair Moore, Sebastian Riedel, and Pontus Stenetorp. 2022. Fantastically ordered prompts and where to find them: Overcoming few-shot prompt order sensitivity. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8086–8098.

MAA. 2023. American mathematics competitions. In *American Mathematics Competitions*.

MAA. 2024. American invitational mathematics examination - aime. In *American Invitational Mathematics Examination - AIME*.

AI Meta. 2024. Introducing meta llama 3: The most capable openly available llm to date. *Meta AI*, 2(5):6.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. 2018. Can a suit of armor conduct electricity? a new dataset for open book question answering. In *EMNLP*.

Sewon Min, Xinxi Lyu, Ari Holtzman, Mikel Artetxe, Mike Lewis, Hannaneh Hajishirzi, and Luke Zettlemoyer. 2022. Rethinking the role of demonstrations: What makes in-context learning work? In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 11048–11064.

Reiichiro Nakano, Jacob Hilton, Suchir Balaji, Jeff Wu, Long Ouyang, Christina Kim, Christopher Hesse, Shantanu Jain, Vineet Kosaraju, William Saunders, and 1 others. 2021. Webgpt: Browser-assisted question-answering with human feedback. *arXiv preprint arXiv:2112.09332*.

Tai Nguyen and Eric Wong. 2023. In-context example selection with influences. *arXiv preprint arXiv:2302.11042*.

Stephen Robertson, Hugo Zaragoza, and 1 others. 2009. The probabilistic relevance framework: Bm25 and beyond. *Foundations and Trends® in Information Retrieval*, 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.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2023. Toolformer: Language models can teach themselves to use tools. *Advances in Neural Information Processing Systems*, 36:68539–68551.

Taylor Sorensen, Joshua Robinson, Christopher Rytting, Alexander Shaw, Kyle Rogers, Alexia Delorey, Mahmoud Khalil, Nancy Fulda, and David Wingate. 2022. An information-theoretic approach to prompt engineering without ground truth labels. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 819–862.

Qiaoyu Tang, Hao Xiang, Le Yu, Bowen Yu, Hongyu Lin, Yaojie Lu, Xianpei Han, Le Sun, and Junyang Lin. 2025. Refcritic: Training long chain-of-thought critic models with refinement feedback. *arXiv preprint arXiv:2507.15024*.

Liang Wang, Nan Yang, and Furu Wei. 2024a. Learning to retrieve in-context examples for large language models. In *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1752–1767.

Shenzhi Wang, Le Yu, Chang Gao, Chujie Zheng, Shixuan Liu, Rui Lu, Kai Dang, Xionghui Chen, Jianxin Yang, Zhenru Zhang, and 1 others. 2025. Beyond the 80/20 rule: High-entropy minority tokens drive effective reinforcement learning for llm reasoning. *arXiv preprint arXiv:2506.01939*.

Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, and 1 others. 2024b. Mmlu-pro: A more robust and challenging multi-task language understanding benchmark. *Advances in Neural Information Processing Systems*, 37:95266–95290.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, and 1 others. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.

Noam Wies, Yoav Levine, and Amnon Shashua. 2023. The learnability of in-context learning. *Advances in Neural Information Processing Systems*, 36:36637–36651.

Zhiyong Wu, Yaoxiang Wang, Jiacheng Ye, and Lingpeng Kong. 2023. Self-adaptive in-context learning: An information compression perspective for in-context example selection and ordering. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1423–1436.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, and 1 others. 2025. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. 2023. Tree of thoughts: Deliberate problem solving with large language models. *Advances in neural information processing systems*, 36:11809–11822.

Jiacheng Ye, Zhiyong Wu, Jiangtao Feng, Tao Yu, and Lingpeng Kong. 2023. Compositional exemplars for in-context learning. In *International Conference on Machine Learning*, pages 39818–39833. PMLR.

Longhui Yu, Weisen Jiang, Han Shi, Jincheng YU, Zhengying Liu, Yu Zhang, James Kwok, Zhenguo Li, Adrian Weller, and Weiyang Liu. 2023. Metamath: Bootstrap your own mathematical questions for large language models. In *The Twelfth International Conference on Learning Representations*.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. Hellaswag: Can a machine really finish your sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*.

Changshuo Zhang, Ang Gao, Xiao Zhang, Yong Liu, and Deyang Li2 Fangchao Liu2 Xinyu Zhang. 2025. Reward mixology: Crafting hybrid signals for reinforcement learning driven in-context learning. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 4373–4383.

Dongxiang Zhang, Lei Wang, Luming Zhang, Bing Tian Dai, and Heng Tao Shen. 2019. The gap of semantic parsing: A survey on automatic math word problem solvers. *IEEE transactions on pattern analysis and machine intelligence*, 42(9):2287–2305.

Yiming Zhang, Shi Feng, and Chenhao Tan. 2022. Active example selection for in-context learning. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*.

Tianshi Zheng, Yixiang Chen, Chengxi Li, Chunyang Li, Qing Zong, Haochen Shi, Baixuan Xu, Yangqiu Song, Ginny Y Wong, and Simon See. 2025. The curse of cot: On the limitations of chain-of-thought in in-context learning. *arXiv preprint arXiv:2504.05081*.

A Case Study

Question:Josh decides to try flipping a house. He buys a house for $80,000 and then puts in $50,000 in repairs. This increased the value of the house by 150%. How much profit did he make?
Label:Totalcost:80,000+50,000=$<<80000+50000=130000>>\Increased value of house: 80,000*1.5=<<80000*1.5=120000>>\n New value of the house:120,000+80,000=$<<120000+80000=200000>>\nSo he made a profit of 200,000-130,000=$<<200000-130000=70000>>\n#### 70000

Wrong answer performed by Zero-shot & few-shot:

Solution: ……let me break this down step by step.\n\nFirst, I need to figure out the total cost Josh incurred. He spent $80,000 on buying the house and an additional $50,000 on repairs. So, adding those together, that's $80,000 + $50,000 = $130,000. Now, it says that the house's value increased by 150%. **Wait, percentage increases can sometimes be tricky**…… The question says he buys the house for $80,000 and then puts in $50,000 in repairs. So, the total cost is indeed $130,000.Therefore, the value increased by 150% of $130,000. To calculate that, I multiply $130,000 by 1.5 Let's do that: $130,000 * 1.5. **Wait a second, but does the problem specify whether the 150% increase is based on the original purchase price or the total cost including repairs?** …… sticking with the initial calculation, the value went up by 150% of $130,000, which is $195,000. Subtracting his total cost of $130,000, his profit is $65,000.\n **(wrong answer)**
Mistakes: mistakenly calculates the 150% increase based on the total investment ($130,000), rather than the original value of the house ($80,000)

Correct answer performed by PICL:

Solution: ……let me break down what's given. The original price of the house is $80,000. Josh spent an additional $50,000 on repairs. So, the total amount he invested before any appreciation would be the sum of these two amounts. Let me write that down:\n\nTotal Investment = Purchase Price + Repair Costs\nTotal Investment = $80,000 + $50,000\n\nHmm, adding those together gives me $130,000. **Wait //interrupt here and begin confusion identification**
Confusion Summary: {It may be confused about whether the 150% increase applies to ***the original purchase*** price or ***the total investment***.}
Selected Demonstration:
"question": "Tommy's home is worth 25% more than he bought it for. He sells it and buys a new house that costs $500,000. If he had to take a loan for the 75% he can't pay, how much did he buy his first house for?"
"answer": "He paid 25% of the new home price because 100 - 75 = <<100-75=25>>25\nHe paid $125,000 towards the new home because 500,000 x .25 = <<500000*.25=125000>>125,000\nTherefore he paid $100,000 for his first home because 125,000 / 1.25 =<<125000/1.25=100000>>100,000"
(Demonstration that can shed light on percentage calculation).
Continue Reasoning after Demonstration Insertion: ……Back to Josh's problem. So, the value increased by 150%. **I think that means the house is now worth 150% more than its original purchase price, not 150% of the total investment.** …… So, recapping:\n\nOriginal Purchase Price: $80,000\nRepair Costs: $50,000\nTotal Investment: $130,000\nAppreciation: 150% of $80,000 = $120,000\nNew Value: $80,000 + $120,000 = $200,000\n\Profit: $200,000 - $130,000 = $70,000. **(correct answer)**

Figure 6: A case study comparing reasoning processes.

To demonstrate how PICL mitigates ambiguity in mathematical reasoning, we analyze a house-flipping profit problem, shown in Fig. 6. In zero-shot and static few-shot settings, the model's reasoning hinges on an ambiguous reference point: whether the stated 150% appreciation is applied to the purchase price or to the total expenditure including repairs (highlighted in yellow). This misspecified baseline causes the model to follow an incorrect computation path and yield an erroneous profit.

A more fine-grained comparison with a static retrieval baseline further clarifies this failure mode. In this case, a static ICL selector based on similarity retrieves the demonstration:

"*Sonny received 45 boxes of cookies from his friend yesterday. He gave 12 to his brother, 9 to his sister, and 7 to his cousin. How many boxes of cookies were left for him?*"

Method	MMLU	MMLU_Pro
Zero-shot	<u>80.0</u>	<u>55.2</u>
Random	69.0	45.4
Best-validate	76.0	50.3
Similarity	70.0	41.9
BM25	64.0	42.3
BGEM3	73.0	42.6
Perplexity	68.0	44.9
Influence	57.0	45.9
LMS3	71.0	43.5
PICL (Ours)	**85.0**	**57.9**

Table 3: Accuracy (%) comparasion on MMLU and MMLU_Pro, two multiple-choice reasoning benchmarks. All few-shot baselines utilize one-shot reasoning. For PICL, parameters are set to k = 1 and r = 1. The best and runner-up methods are highlighted in bold and underlined, respectively

Although this example is superficially math-related, it only involves additive and subtractive bookkeeping and does not instantiate the percentage operator required by the target problem. As a result, even with static ICL support, the model still fails, because the retrieved example does not address the actual reasoning bottleneck—namely, how to interpret the percentage relationship and determine the correct base quantity.

In contrast, PICL detects this uncertainty as a confusion point and retrieves a closely matched demonstration that directly matches the operator required by the target problem:

"*Tommy's home is worth 25% more than he bought it for. He sells it and buys a new house that costs $500,000. If he had to take a loan for the 75% he can't pay, how much did he buy his first house for?*"

This demonstration explicitly operationalizes percentage reasoning with respect to the original value, including the correct treatment of the base quantity. Injecting this demonstration at inference time resolves the ambiguity, leading the model to ground the 150% increase on the purchase price and compute the correct profit. This case study suggests that PICL can localize mid-inference ambiguities and provide targeted corrective context to recover valid reasoning trajectories.

B Generalization to Multiple-Choice Benchmarks

Although PICL is introduced for open-ended mathematical reasoning with short-answer supervision, we further test its robustness under a format shift to multiple-choice evaluation. Specifically, we evaluate on MMLU (Hendrycks et al., 2021) and MMLU_Pro (Wang et al., 2024b), whose tasks and output space (option selection) differ substantially from the short-answer benchmarks used in our main experiments. We carry out this experiment on Deepseek-R1-Distilled-Qwen-7B.

Table 3 shows that PICL achieves the best performance on both datasets, improving over zero-shot by +5.0 on MMLU (80.0→85.0) and +2.7 on MMLU_Pro (55.2→57.9). Notably, several retrieval- or selection-based baselines underperform zero-shot, suggesting that statically chosen demonstrations can induce negative transfer under multiple-choice prompting. In contrast, PICL remains consistently effective, indicating that confusion-aware retrieval and dynamic demonstration insertion transfer beyond open-ended generation and provide robust gains under choice-based reasoning.

We further analyze the sensitivity of PICL on multiple-choice benchmarks with respect to the same two hyperparameters: the maximum number of interventions r and the number of demonstrations inserted per intervention k.

Fig. 7 reports the results when varying r from 1 to 4 on MMLU and MMLU_Pro. We observe that PICL achieves its strongest performance under a small number of interventions (typically $r \in \{1, 2\}$), while increasing r beyond 2 leads to a clear degradation. This trend mirrors our findings on open-ended reasoning tasks: a small number of well-timed interventions can effectively correct the model's trajectory, whereas frequent interruptions may disrupt the continuity of choice-based reasoning and introduce additional contextual noise, ultimately hurting accuracy.

We also vary k from 1 to 4 in Fig. 8. PICL is most robust when k is small (typically $k \in \{1, 2\}$), and larger k does not yield further gains and can slightly reduce performance. Overall, these results suggest that even under multiple-choice prompting, PICL benefits primarily from relevance rather than quantity: inserting one (or at most two) highly matched demonstrations is sufficient to resolve confusion, while inserting more examples may introduce competing patterns that dilute the most useful signal.

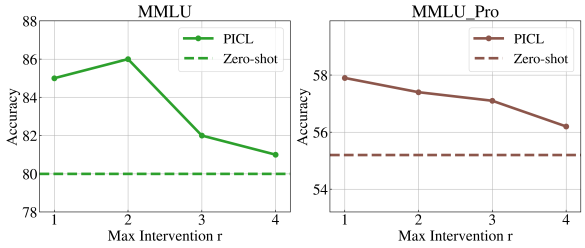


Figure 7: Performance comparison between PICL and Zero-shot with r ranging from 1 to 4 on multiple-choice benchmarks.

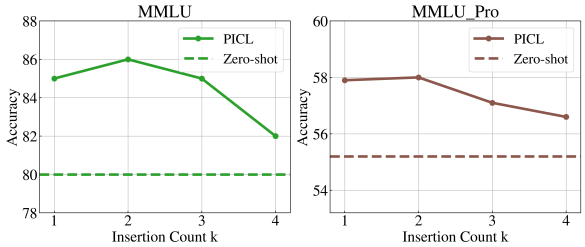


Figure 8: Performance comparison between PICL and Zero-shot with k ranging from 1 to 4 on multiple-choice benchmarks.

C Generalization to Broader NLP Tasks

Beyond mathematical reasoning benchmarks, we further evaluate PICL on two broader NLP tasks to examine whether its advantages transfer outside the math domain. Specifically, we consider OBQA (Mihaylov et al., 2018), a question answering benchmark, and HellaSwag (Zellers et al., 2019), a commonsense reasoning benchmark, both evaluated on Deepseek-R1-Distilled-Qwen-7B. As shown in Table 4, PICL achieves the best performance on both datasets, obtaining 75.8 on OBQA and 50.8 on HellaSwag, with an average score of 63.28. Compared with the zero-shot baseline, PICL improves performance by +2.4 on OBQA, +4.3 on HellaSwag, and +3.4 on average. Notably, while a few static selection methods remain competitive on individual datasets, most of them fail to consistently outperform zero-shot, suggesting that fixed demonstrations may still introduce irrelevant patterns beyond mathematical reasoning. In contrast, PICL remains consistently effective across both question answering and commonsense reasoning, indicating that confusion-aware retrieval and dynamic demonstration insertion generalize well to broader NLP tasks.

Method	OBQA	HellaSwag	Avg.
Zero-shot	73.4	46.5	59.9
Random	69.9	46.1	58.0
Best-validate	62.5	47.7	55.1
Similarity	69.1	43.8	56.4
BM25	63.7	45.3	54.5
BGEM3	69.5	47.3	58.4
PPL	70.3	39.5	54.9
Influence	65.6	49.2	57.4
LMS3	72.3	48.4	60.3
PICL (Ours)	**75.8**	**50.8**	**63.3**

Table 4: Accuracy (%) comparison on OBQA and HellaSwag, two broader NLP benchmarks, evaluated on Deepseek-R1-Distilled-Qwen-7B. Avg. is computed over the two datasets. The best and runner-up methods are highlighted in bold and underlined, respectively.

D Generalization to Broader Backbones

To examine whether PICL generalizes beyond the DeepSeek-R1-distilled model family, we further evaluate it on two broader backbones, namely Qwen3-8B and Llama3-8B, on three mathematical reasoning benchmarks: MATH500, AMC23, and AIME24. As shown in Table 5, PICL achieves the best performance on both backbones, obtaining an average accuracy of 87.3 on Qwen3-8B and 27.3 on Llama3-8B. Compared with the zero-shot baseline, PICL improves the average score by +4.3 on Qwen3-8B and +5.0 on Llama3-8B. Notably, several static retrieval-based baselines still fail to consistently outperform zero-shot, especially on Qwen3-8B, suggesting that the negative transfer induced by fixed demonstrations is not limited to the DeepSeek-R1-distilled backbones. In contrast, PICL remains consistently effective across both broader backbones, indicating that confusion-aware retrieval and dynamic demonstration insertion are robust to changes in the underlying model family.

E Robustness Analysis of Interruption Token Set

To examine the robustness of the interruption mechanism in PICL, we compare two choices of interruption token sets. The original set contains only a single token, {`wait`}, while the expanded set includes a broader group of semantically related hesitation or redirection markers, {`wait`, `maybe`, `instead`, `suggest`}.

Table 6 reports the results on GSM8K, MATH500, AMC23, and AIME24. We observe that the two variants yield similar overall performance, with the expanded token set slightly improving the average accuracy from 83.0 to 83.6. In particular, the expanded set brings a gain on AMC23 (90.0 → 92.5), while maintaining comparable performance on the other three datasets. These results suggest that PICL is not overly sensitive to the exact choice of interruption tokens, as long as they capture semantically meaningful signals of hesitation or self-correction. This indicates that the interruption-token mechanism is stable in practice, and that it can be flexibly adapted to different model families with only minor lexical adjustments.

F Efficiency Analysis

To quantify the efficiency of the proposed method, we conducted a statistical analysis of the inference length using DeepSeek-R1-Distilled-Qwen-7B. We aggregated the number of tokens generated for the reasoning chain of each query across all evaluated datasets. As shown in Fig. 9, the average token consumption for PICL is 3,951, compared to an average of 3,409 for the baseline approaches. This comparison demonstrates that the absolute token usage of PICL remains within a reasonable range, ensuring that the enhanced reasoning capability does not come at the expense of excessive computational latency.

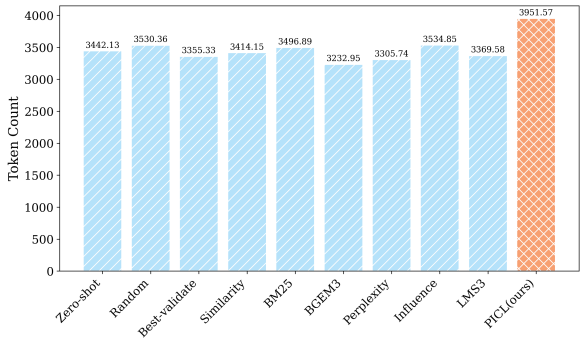


Figure 9: Average token used per question for reasoning in different methods.

G Prompt Design

To implement the aforementioned stages, PICL employs specific prompt designs tailored to different reasoning scenarios and the confusion detection task, which is shown in Fig. 10. For zero-shot and PICL settings, the prompt structure is designed to guide the model in step-by-step reasoning, with a requirement to enclose the final answer within

Method	*Qwen3-8B*				*Llama3-8B*			
	MATH500	AMC23	AIME24	Avg.	MATH500	AMC23	AIME24	Avg.
Zero-shot	91.6	87.5	70.0	83.0	46.0	17.5	3.3	22.3
Random	91.8	82.5	63.3	79.2	41.0	15.0	3.3	19.8
Best-validate	91.0	87.5	66.7	81.7	42.2	17.5	3.3	21.0
Similarity	92.0	85.0	66.7	81.2	43.4	12.5	3.3	19.7
BM25	89.5	82.5	73.3	81.8	43.2	12.5	3.3	19.7
BGEM3	90.6	82.5	66.7	79.9	43.8	15.0	6.7	21.8
PPL	92.0	90.0	63.3	81.8	44.4	20.0	3.3	22.6
Influence	91.5	85.0	66.7	81.1	44.2	12.5	3.3	20.0
LMS3	91.8	90.0	70.0	83.9	43.0	20.0	6.7	23.2
PICL (Ours)	**92.8**	**92.5**	**76.7**	**87.3**	**46.8**	**25.0**	**10.0**	**27.3**

Table 5: Accuracy (%) comparison on MATH500, AMC23, and AIME24 across two broader backbones, Qwen3-8B and Llama3-8B. Avg. is computed over the three datasets. The best and runner-up methods are highlighted in bold and underlined, respectively.

Method	GSM8K	MATH500	AMC23	AIME24	Avg.
PICL w/ original interrupt token set	91.7	**90.2**	90.0	60.0	83.0
PICL w/ expanded interrupt token set	**91.8**	90.0	**92.5**	60.0	**83.6**

Table 6: Accuracy (%) comparison under two interruption token sets in PICL. The original set uses only `wait`, while the expanded set additionally includes `maybe`, `instead`, and `suggest`. Avg. is computed over GSM8K, MATH500, AMC23, and AIME24. The best and runner-up results are highlighted in bold and underlined, respectively.

Start Prompt:

Zero-shot & PICL: <|im_start|>system\nPlease reason step by step, and put your final answer within \\boxed{}.<|im_end|>\n<|im_start|>user\n Question: [*question*] <|im_end|>\n<|im_start|>assistant\n

Few-shot: <|im_start|>system\nPlease reason step by step, and put your final answer within \\boxed{} given provided examples.<|im_end|>\n <|im_start|>user\n Examples: [*examples*] Question: [*question*] <|im_end|>\n <|im_start|>assistant\n

Detection Prompt:

PICL: Please analyze the intermediate problem-solving process for a math question and determine whether it has shown any signs of confusion at this stage. Answer with "Yes" if this process appears confused, and "No" if the process does not appear confused. [*currect generated sequence*]. If your answer is "Yes", you need to summarize the current confusion and write it in the format confusion{ }.

Figure 10: An overview of prompt used in this work.

boxed{ }. This ensures a standardized output format for consistent evaluation. In few-shot scenarios, the prompt is augmented with provided examples, prompting the model to leverage these examples for reasoning while still adhering to the final answer formatting requirement.

Notably, the Detection Prompt plays a critical role in the Confusion Identification and Summarization stage. It instructs the model to analyze the intermediate problem-solving process, determine if there are signs of confusion, and answer with "Yes" or "No". If confusion is identified ("Yes"), the prompt further requires a concise summary of the confusion in a specified format confusion{ }, which directly contributes to generating the confusion summary $\mathcal{C}$ as formalized in the stage description. These prompt designs collectively facilitate the effective execution of PICL's core mechanisms, ensuring accurate confusion detection and contextually relevant demonstration integration.