PatternICL: Pattern-Aware Instruction Tuning and Reinforcement Learning for In-Context Learning

Anonymous ACL submission

Abstract

Large language models (LLMs) demonstrate remarkable capabilities through in-context learning (ICL), establishing it as a primary paradigm for adapting LLMs to downstream tasks. However, recent studies reveal a critical limitation: despite the foundational role of exemplars, LLMs frequently bypass the provided context and fail to rely on these demonstrations as expected during inference. In this work, we further observe that this alignment gap persists even when models are explicitly prompted to adhere to in-context examples, indicating an underutilization of contextual patterns that limits the efficiency and robustness of contextual evidence use. To address this challenge, we propose PatternICL, a novel two-stage training framework designed to explicitly teach LLMs to capture and leverage data patterns within in-context examples. In the first stage, we automatically construct pattern-aware instructions from existing benchmarks and employ supervised fine-tuning (SFT) to establish a pattern-aware reasoning and output format. In the second stage, we leverage Group Relative Policy Optimization (GRPO) driven by a multi-dimensional composite reward to reinforce both example selection and prediction fidelity in a computationally efficient manner. Extensive evaluations across 20 natural language understanding datasets demonstrate that PatternICL yields consistent and significant improvements over competitive baselines across varying model scales and architectures.

1 Introduction

Large language models (LLMs) demonstrate remarkable capabilities through in-context learning (ICL), achieving impressive performance on downstream tasks by conditioning on only a few input-label pairs (examples/cases) without requiring parameter updates (Min et al., 2022b). This powerful emergent property highlights their immense application potential, establishing ICL as a dominant paradigm for deploying LLMs across diverse downstream scenarios (Duan et al., 2024).

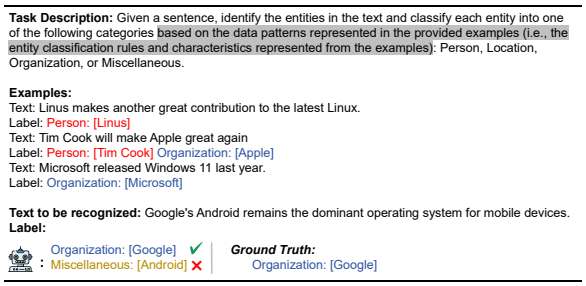

Figure 1: A named entity recognition example that demonstrates how LLMs ignore in-context data patterns. The gray-highlighted text in the task description instructs the model to classify entities based on patterns in the examples.

The remarkable success of ICL has sparked extensive research into its underlying mechanics, uncovering behaviors that often counter human intuition. Notes (Min et al., 2022b; Long et al., 2024) have revealed that replacing the ground-truth labels in demonstrations with random permutations from the label space only marginally impairs classification accuracy. This puzzling finding implies that LLMs frequently bypass the specific input-label patterns showcased in the context, relying instead on a coarse understanding of the task's overarching structure and format (Min et al., 2022b; Huang et al., 2025).

Building upon these insights, we further demonstrate a more troubling vulnerability: even when explicitly instructed to adhere to the provided data patterns, LLMs frequently disregard such guidance. Consequently, they underutilize the in-context examples, weakening both contextual evidence use and cross-domain generalization. To illustrate this, Figure 1 depicts a concrete failure mode in a named entity recognition (NER) task. Here, both the

first and third examples establish a clear negative-pattern: operating systems (`Linux` and `Windows 11`) are deliberately excluded from entity extraction. Given the test sentence containing `Android`—also an operating system—a pattern-adherent model should omit it. However, despite instructions explicitly targeting "the data patterns represented in the provided examples", strong frontier models still categorize `Android` as `Miscellaneous`. This underscores a pervasive tendency of LLMs to over-rely on internal parametric knowledge at the expense of contextual evidence.

We attribute this sub-optimal behavior to the fact that ICL capacity typically emerges as a passive byproduct of model scaling (Hahn and Goyal, 2023), meaning that existing architectures are never explicitly trained to look up, align, and capture patterns across context boundaries. Faced with this bottleneck, standard engineering fixes either resort to intricate prompt engineering (Pang et al., 2023; Huang et al., 2025) or leverage more elaborate test-time example selection and reasoning procedures (Peng et al., 2024; Qin et al., 2024; Chen et al., 2025). Nevertheless, these approaches encounter severe roadblocks: (1) prompt engineering suffers from fragile alignment and high variance across different model families (Yu et al., 2023; Pang et al., 2023); (2) test-time case analysis exponentially inflates inference latency and computational overhead, particularly when reasoning models fall into loops of overthinking (Qin et al., 2024; Chen et al., 2025).

To mitigate this dilemma, we propose **PatternICL**, a novel two-stage training paradigm that explicitly encourages LLMs to capture in-context data patterns and perform pattern-guided predictions. Conceptually, our approach formalizes ICL as an organic case-based reasoning workflow: the model learns to selectively retrieve the most pattern-similar cases from the context and systematically adapt them for final inference. In the first stage, we automatically synthesize pattern-aware instruction data from existing pools to specify input-example relevance, employing supervised fine-tuning (SFT) to establish a pattern-aware reasoning and output layout. In the second stage, we optimize the policy via Group Relative Policy Optimization (GRPO), leveraging a multi-dimensional composite reward that scores both retrieval precision and prediction accuracy. This reinforces pattern fidelity without shifting the architectural inference cost.

Our core contributions are summarized as follows:

- **A Novel Paradigm**: We introduce PatternICL, a two-stage alignment framework fusing pattern-aware instruction tuning (SFT) with reinforcement learning (GRPO) to explicitly guide LLMs to exploit contextual pattern similarities and improve evidence use in ICL.
- **Automated Data Engine**: We engineer an annotation-free pipeline to construct diverse pattern-aware instruction datasets from legacy benchmarks. The resulting framework is highly extensible and seamlessly unifies both sentence-level tasks (e.g., sentiment analysis) and token-level tasks (e.g., named entity recognition).
- **Strong Empirical Performance**: Rigorous empirical evaluations across 20 natural language understanding datasets validate the robust scalability of our method. Notably, PatternICL delivers a substantial boost of **6.71 points** in average score for Qwen2.5-32B-Instruct over its vanilla baseline and consistently outperforms strong prompting-based baselines.

2 Related Work

2.1 In-Context Learning

In-context learning (ICL) has gained significant attention since GPT-3 (Brown et al., 2020), with many studies focusing on example selection because the quality, diversity, and ordering of demonstrations strongly affect downstream performance (Liu et al., 2022; Wu et al., 2023; Gupta et al., 2023; Peng et al., 2024; Qin et al., 2024; Wang et al., 2024). As a naturally emergent capability of LLMs (Hahn and Goyal, 2023), ICL has also drawn interest from studies seeking to understand its mechanisms. (Xie et al., 2022) argue that LLMs infer latent concepts shared among examples, while (Min et al., 2022b) and (Long et al., 2024) show that replacing labels with random ones causes little damage, suggesting weak use of example-level patterns. A complementary line of work explicitly trains models for stronger ICL behavior, such as MetaICL (Min et al., 2022a), pre-training to learn in context (Gu et al., 2023), and meta demonstration distillation (Li et al., 2024). However, these methods primarily strengthen general ICL ability rather than explicitly aligning models to reuse

pattern-level evidence inside the prompt. (Huang et al., 2025) address this by summarizing training-set patterns as extra prompts, but the method is task-specific and prompt-heavy.

From a CBR perspective, ICL can be viewed as a form of case-based reasoning where the in-context examples serve as the case base, example selection corresponds to case retrieval, and the model's prediction process mirrors case adaptation.

2.2 Instruction Tuning and Reinforcement Learning for LLMs

Instruction tuning (IT) followed by reinforcement learning (RL) has become a common LLM training paradigm (Duan et al., 2024; Chung et al., 2024; Guo et al., 2025). IT teaches instruction-following from "input, output" pairs (Wei et al., 2022a; Longpre et al., 2023), while RL further optimizes behavior with rewards. Compared with PPO-style methods that rely on clipped policy updates and often require an auxiliary reward model (Schulman et al., 2017; Ouyang et al., 2022), GRPO (Shao et al., 2024) removes the critic through group-relative comparisons. We adopt IT + GRPO so that IT teaches the pattern-aware format and GRPO refines example selection quality without human preference annotations.

3 Method

3.1 Overview

PatternICL is a two-stage training approach. In **Stage 1 (SFT)**, we automatically construct pattern-aware instruction data from existing datasets and fine-tune the LLM to identify and utilize data patterns in ICL examples (Figure 2). The construction proceeds as follows: (1) we fine-tune DeBERTa on each dataset to obtain an encoder that captures task-specific patterns; (2) we encode all instances and measure their pattern similarity via cosine similarity; (3) for each instruction, we select examples that are pattern-matching, pattern-mismatching, or ambiguous relative to the input; (4) the output is designed so that the model first selects the most pattern-similar example(s) via `<sel>` tags, then reasons and predicts accordingly. To handle real-world scenarios where no similar example exists, we also construct instructions composed entirely of pattern-mismatching examples, where the model learns to output `<sel> none </sel>` and rely on its own understanding.

In **Stage 2 (GRPO)**, we apply Group Relative

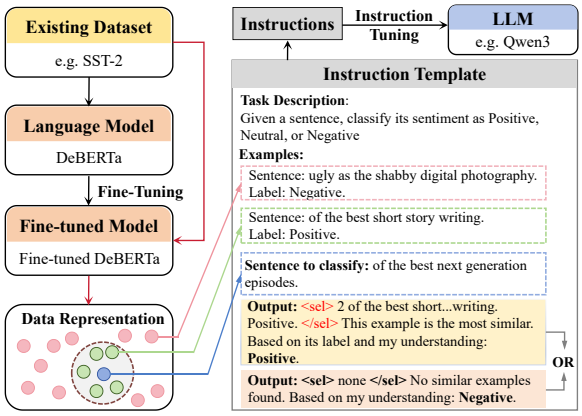

Figure 2: The process of constructing pattern-aware instruction data for SFT.

Policy Optimization (Shao et al., 2024) to further refine the SFT model. A composite reward evaluates each generated output on format compliance, selection quality (measured by pattern similarity), and prediction accuracy, reinforcing better pattern-based decision-making without increasing inference cost. This two-stage decomposition is designed to separate *behavior specification* from *behavior refinement*. SFT teaches the model what a pattern-aware ICL trajectory should look like, while GRPO distinguishes between outputs that are merely correct and outputs that are correct for better contextual reasons. In this sense, PatternICL optimizes not only the end prediction, but also the path by which the model arrives at that prediction.

3.2 Dataset Collection

We utilize 20 datasets across 8 NLU tasks, including all 9 GLUE (Wang et al., 2019) datasets (SST-2, CoLA, QNLI, MNLI, WNLI, RTE, QQP, MRPC, and STS-B), 4 hate speech detection datasets (de Gibert et al., 2018) (hate_speech18, tweet-hate, tweet-atheism, and tweet-feminist), 4 question answering datasets (Tafjord et al., 2019; Khot et al., 2020) (quarel, openbookqa, qasc, and commonsense_qa), and 3 named entity recognition datasets (Tjong Kim Sang and De Meulder, 2003; Walker et al., 2006) (CoNLL03, ACE04, and ACE05). Following (Wei et al., 2022a), we split these into two non-overlapping sets (Part A and Part B, 10 each) and adopt cross-evaluation: fine-tune on Part A, evaluate on Part B, and vice versa. We keep the split as task-balanced as possible, so that datasets from the same task category are distributed evenly across the two parts whenever feasible. Specifically, Part A contains SST-2, QNLI,

RTE, QQP, STS-B, hate_speech18, tweet-hate, quarel, openbookqa, and CoNLL03, while Part B contains CoLA, MNLI, WNLI, MRPC, tweet-atheism, tweet-feminist, qasc, commonsense_qa, ACE04, and ACE05. This cross-partition design is important for our setting because PatternICL does not aim to memorize dataset-specific answer distributions; it aims to teach a transferable habit of reading examples for patterns. By training and evaluating on disjoint dataset partitions, we test whether the learned behavior generalizes across tasks rather than only within the same benchmark collection.

3.3 Pattern-Aware Representations

Following prior work (Liu et al., 2022; Zhou and Srikumar, 2022), fine-tuned model representations are more aligned with task-specific data patterns. For each dataset d_i, we fine-tune DeBERTa-v3-large (He et al., 2021) on it to obtain a task-specific model $\mathcal{L}^i$, and then encode all instances $\{x^i_j\}$ through $\mathcal{L}^i$ to obtain their pattern-aware representations. We use cosine similarity to measure pattern similarity between instances. Here, "data patterns" refer to task-conditioned latent features that emerge in the encoder representation space after supervised fine-tuning and that support label transfer under ICL. Concretely, these features need not be expressible as explicit symbolic rules; rather, they correspond to manifold structure in the task-specific hidden space that captures regularities relevant to prediction (e.g., positive adjective-noun expressions in sentiment analysis or exclusion rules in NER). Following (Reimers and Gurevych, 2019), we empirically determine thresholds through manual examination: pairs above 0.8 exhibit highly similar patterns, below 0.4 show no pattern matching, and between these values are ambiguous. We emphasize that these representations are not intended to be universal semantic embeddings. Their purpose is narrower: to provide a task-conditioned geometry in which example similarity better reflects the kinds of regularities that matter for label transfer in ICL. This is why task-specific fine-tuning is critical in our pipeline and why using an unfine-tuned encoder weakens downstream performance in the ablation study.

3.4 Pattern-Aware Instruction Construction

We design instructions without manual annotation by combining standard ICL components—task description, in-context examples, and the sentence to classify—with a pattern-aware output. Following FLAN-style template diversification (Wei et al., 2022a; Longpre et al., 2023), we design 10 diverse templates per dataset, randomly selecting one for each instruction.

3.4.1 Example Selection.

For each instruction, we randomly select $k \in \{3, 9, 18\}$ examples comprising pattern-matching, pattern-mismatching, and potentially pattern-matching data in a 1:1:1 ratio (*instructions containing pattern-matching data*). With 50% probability, all pattern-matching data is replaced with pattern-mismatching data (*instructions without pattern-matching data*), enabling the model to learn different strategies based on pattern availability. Concretely, pattern-matching examples have cosine similarity above 0.8 to the test input, pattern-mismatching examples fall below 0.4, and potentially pattern-matching examples lie in between, allowing the model to learn from clear positives, hard negatives, and ambiguous cases. We cap the number of selected examples at at most two because the NLU tasks considered here typically depend on relatively local feature alignment, such as sentiment-bearing expressions or exclusion rules in NER. In these settings, one or two well-matched cases are usually sufficient to expose the operative pattern while avoiding unnecessary selection noise. For more composition-heavy tasks, this constraint could be naturally relaxed to allow a larger selection budget.

3.4.2 Output Design and Dataset Scaling.

For instructions containing pattern-matching data, the output incorporates chain-of-thought reasoning (Wei et al., 2022b): the model first identifies up to two similar examples by enclosing a structured selection—including the positional index, a truncated content excerpt, and the label—within `<sel> .... </sel>` as (e.g., `<sel> 2 of the best ... writing. Positive. </sel>`), and then makes its prediction based on the selected examples (Figure 3). The positional index enables straightforward reward parsing, while the content excerpt helps verify that the correct example has been identified. We limit the number of selectable examples to at most two, as this is empirically sufficient to cover the pattern information required for most inputs while achieving a balance between effectiveness and efficiency. For instructions without pattern-matching data, the output begins with

(a) Sentence-Level Task (SST-2)

Task: Given a sentence, classify it as Positive or Negative.

Examples:
- Sentence: ugly as the shabby digital photography. Label: Negative.
- Sentence: of the best short story writing. Label: Positive.
- Sentence: an exceptional coming-of-age film. Label: Positive.

.....

Test Input: Sentence: of the best next generation episodes.

Output (with pattern match):
<sel>2 of the best short...writing. Positive.</sel>
This example is the most similar. Based on its label and my understanding:
Positive.

Output (without pattern match):
<sel>none</sel> No similar examples found. Based on my understanding:
Negative.

(b) Token-Level Task (Named Entity Recognition)

Task: Identify Person, Organization, Location and Miscellaneous entities.

Examples:
- Input: EU rejects German call to boycott British lamb.
  Label: [["EU", "org"], ["German", "misc"], ["British", "misc"]]
- Input: Only France and Britain backed Fischler's proposal.
  Label: [["France", "loc"], ["Britain", "loc"], ["Fischler", "per"]]

Identify Entities for:
Input Text: Two Iranian opposition leaders meet in Baghdad.

Output:
<sel>1 EU rejects...British lamb. [...]</sel>
<sel>2 Only France...proposal. [...]</sel>
"German" ~ "Iranian" (misc), "France" ~ "Baghdad" (loc).
Output: [["Iranian", "miscellaneous"], ["Baghdad", "location"]]

Figure 3: Instruction templates for (a) sentence-level tasks (SST-2) and (b) token-level tasks (NER). Each `<sel>...</sel>` tag contains the positional index, a truncated content excerpt, and the label of the selected example. The model selects up to two pattern-matching examples, or outputs `<sel> none </sel>` when no match exists.

`<sel> none </sel>` and relies on the model's own understanding to make the prediction. This explicit format is crucial because it transforms latent example usage into an observable training target. Instead of inferring from the final label alone whether the model used context appropriately, we can directly supervise whether it identified relevant evidence, abstained when no reliable match exists, and tied its prediction to selected cases. This makes the training signal much closer to the behavior we actually want from ICL.

We limit the number of training examples to 5k per dataset (upsampling smaller datasets when necessary), yielding 50k instructions per partition. Random template selection and data composition ensure diversity even for upsampled data.

3.4.3 Instruction Tuning.

We fine-tune the Qwen series (Qwen3-14B/32B, Qwen2.5-32B/72B-Instruct) by maximizing $\max_\theta \sum_{(\mathbf{x},\mathbf{y})\in\mathcal{I}} \log P_\theta(\mathbf{y} \mid \mathbf{x})$. We train for 3 epochs using AdamW (β_1=0.9, β_2=0.999), batch size 16, learning rate 5×10^{-5} with linear warmup and decay, on a multi-GPU accelerator cluster with LoRA (Hu et al., 2022) (rank 16).

3.5 Stage 2: GRPO-Based Reinforcement Optimization

SFT treats all correct outputs equally and cannot provide fine-grained signals about the *quality* of example selection. We apply GRPO (Shao et al., 2024) to allow the model to explore different outputs and learn from a continuous reward signal, optimizing its decision-making without changing the output format or length.

3.5.1 GRPO Training.

Given an instruction input $\mathbf{x}$, the current policy π_θ generates a group of $G = 8$ candidate outputs $o_1, o_2, \ldots, o_G$ by sampling the model eight times on the same input under stochastic decoding. Since each sampled response may differ, a single input can produce multiple candidate outputs. Each output o_i receives a reward $R(o_i, \mathbf{x})$, and the advantage is computed relative to the group:

$$\hat{A}_i = \frac{R(o_i, \mathbf{x}) - \text{mean}(\{R(o_j)\}_{j=1}^G)}{\text{std}(\{R(o_j)\}_{j=1}^G)} \quad (1)$$

The policy is then updated by maximizing the GRPO objective:

$$\mathcal{J}(\theta) = \mathbb{E}_{\mathbf{x}}\Bigg[\frac{1}{G}\sum_{i=1}^{G}\min\Big(r_i(\theta)\hat{A}_i, \text{clip}(r_i(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_i\Big) - \beta D_{\text{KL}}(\pi_\theta \parallel \pi_{\text{ref}})\Bigg] \quad (2)$$

where $r_i(\theta) = \pi_\theta(o_i|\mathbf{x})/\pi_{\text{old}}(o_i|\mathbf{x})$ is the importance ratio, ϵ is the clipping parameter, π_{ref} is the SFT model serving as the reference policy, and β controls the KL penalty.

3.5.2 Reward Design.

We design a composite reward $R = R_f + R_s + R_a$ that evaluates each generated output along three dimensions:

Format Reward (R_f). We verify that the output conforms to the expected format: it must contain one or two `<sel>...</sel>` tags, each starting with a valid positional index followed by a content

excerpt and label (or a single <sel> none </sel>), followed by a reasoning sentence and a prediction.

$$R_f = \begin{cases} 1 & \text{if format is valid} \\ 0 & \text{otherwise} \end{cases} \quad (3)$$

Selection Reward (R_s). Let $\mathrm{sim}(x, e)$ denote cosine similarity in the pattern-aware representation space, $\mathcal{S}$ ($|\mathcal{S}| \leq 2$) the selected examples, and τ=0.4 the similarity threshold. When similar examples exist ($\exists e$: $\mathrm{sim}(x, e) > \tau$): $R_s = \frac{1}{|\mathcal{S}|} \sum_{e \in \mathcal{S}} \mathrm{sim}(x, e)$ if $\mathcal{S} \neq \emptyset$, or -0.5 if the model outputs <sel> none </sel>. When no similar examples exist: $R_s = 1.0$ for <sel> none </sel>, or -0.5 for selecting any example. Averaging over $\mathcal{S}$ discourages indiscriminate selection.

Accuracy Reward (R_a). We verify whether the final prediction matches the ground-truth label:

$$R_a = \begin{cases} 1 & \text{if prediction is correct} \\ 0 & \text{otherwise} \end{cases} \quad (4)$$

The GRPO training data uses the same instruction inputs as SFT. We set $G = 8$, $\epsilon = 0.2$, $\beta = 0.04$, and train for 1 epoch with a learning rate of 1×10^{-5}. The resulting reward is intentionally simple and verifiable. Rather than introducing an additional learned reward model, we rely on explicit structural constraints and measurable selection quality. This keeps the optimization target interpretable and reduces the risk that the policy improves by exploiting opaque reward-model preferences unrelated to contextual pattern fidelity.

4 Experiments

4.1 Experimental Setting

Baselines. We compare against the original LLMs (4 models from Qwen series) and the strong prompting-based method **Guidelines** (Huang et al., 2025), which summarizes data patterns from the training set as prompts. To extend it to other NLU tasks, we manually design task-specific prompts for each dataset and follow the same predict-then-verify framework as the original method. For Qwen3 (reasoning models), we disable reasoning by default (analysis in Section 4.4).

Setting. The same 20 datasets are used for both training and evaluation with cross-validation to prevent data leakage. At inference time, we evaluate under two example selection strategies: Random, which randomly samples examples from the training set, and Top-K (Liu et al., 2022), which retrieves the K most semantically similar examples to the test input based on RoBERTa-large (Liu et al., 2019) sentence embeddings. We evaluate under both 3-shot and 10-shot settings. Temperature is set to 0.

4.2 Main Results

Table 1 shows the results. PatternICL consistently improves all models across all settings, with Qwen2.5-32B-Instruct achieving the largest gain of **6.71 points** (Top-k, 10-shot) (Wei et al., 2022b). Our method also consistently outperforms the strong Guidelines baseline (Huang et al., 2025).

Smaller models benefit more from PatternICL: Qwen2.5-32B-Instruct gains 6.71 points while the larger Qwen2.5-72B-Instruct gains 5.62 points (Top-k, 10-shot), suggesting that PatternICL is particularly effective at compensating for weaker baseline pattern-utilization capabilities. This pattern is consistent with our motivating hypothesis. Many instruction-tuned LLMs already encode enough task knowledge to solve the target task, but they differ in how reliably they align that knowledge with evidence provided in the prompt. PatternICL appears to help most when the base model is less capable of spontaneously extracting and reusing contextual regularities, thereby closing part of the gap to stronger backbones.

PatternICL benefits more from semantic selection and more examples than the original models: Qwen3-32B gains 2.56 points from Random-3 to Top-k-10, while PatternICL (Qwen3-32B) gains 3.48 points. This is because more semantically similar examples increase the likelihood of encountering similar patterns, which PatternICL actively leverages (Xie et al., 2022). Importantly, this trend suggests that PatternICL is not merely learning a new output template. If the gains came only from format imitation, improving retrieval quality would offer limited additional benefit. Instead, the larger improvement under Top-k retrieval indicates that the trained models have become more sensitive to the usefulness of the evidence available in context, which is precisely the behavior we seek to cultivate. The substantial gains under the Random setting further indicate that PatternICL does not depend solely on high-quality retrieval. When demonstrations are randomly sampled and may contain distractors or mismatching patterns, the model is better able to reject unhelpful context and follow the <sel> none </sel> trajectory, avoiding the pattern-miscopying failures that commonly affect vanilla ICL. Taken

Table 1: Experimental results on 20 datasets. "GLUE Avg." denotes the average performance on 9 GLUE datasets, and "Avg." denotes the average over all 20 datasets. "3-shot" and "10-shot" refer to using 3 and 10 examples respectively in ICL. † indicates statistically significant improvement over the corresponding baseline ($p < 0.05$, paired t-test).

Selection	Method (Model)	3-shot		10-shot	
		GLUE Avg.	Avg.	GLUE Avg.	Avg.
Random	Qwen3-14B	74.16	73.91	74.49	74.02
	Qwen3-32B	79.15	78.73	80.24	79.68
	Qwen2.5-32B-Instruct	71.45	70.52	72.68	71.74
	Qwen2.5-72B-Instruct	76.84	76.42	77.95	77.36
	Guidelines (Qwen3-32B)	80.28	79.78	81.01	80.87
	Guidelines (Qwen2.5-72B-Instruct)	78.36	77.96	80.55	80.12
	PatternICL (Qwen3-14B)	79.63†	79.27†	80.15†	80.04†
	PatternICL (Qwen3-32B)	**83.68**†	**82.65**†	**84.56**†	**84.44**†
	PatternICL (Qwen2.5-32B-Instruct)	77.58†	76.74†	78.71†	77.81†
	PatternICL (Qwen2.5-72B-Instruct)	81.23†	81.07†	83.24†	82.56†
Top-k	Qwen3-14B	74.58	74.32	75.15	75.04
	Qwen3-32B	80.73	80.31	81.86	81.29
	Qwen2.5-32B-Instruct	72.74	71.86	73.95	73.08
	Qwen2.5-72B-Instruct	78.35	77.96	79.48	78.92
	Guidelines (Qwen3-32B)	80.63	80.01	81.53	81.13
	Guidelines (Qwen2.5-72B-Instruct)	78.86	78.24	81.01	80.72
	PatternICL (Qwen3-14B)	80.47†	80.06†	81.45†	81.02†
	PatternICL (Qwen3-32B)	**85.21**†	**84.66**†	**86.25**†	**86.13**†
	PatternICL (Qwen2.5-32B-Instruct)	78.87†	78.35†	80.12†	79.79†
	PatternICL (Qwen2.5-72B-Instruct)	83.21†	83.03†	85.12†	84.54†

Table 2: Ablation study (top-k, 10-shot). w/o chain-of-thought: directly predict the label. w/ unfinetuned DeBERTa: use DeBERTa without task-specific fine-tuning. w/o GRPO: SFT model only, without GRPO stage.

Method	Glue Avg.	Avg.
PatternICL (Qwen2.5-32B-Instruct)	**80.12**	**79.79**
w/o chain-of-thought	78.18	78.01
w/ unfinetuned DeBERTa	77.63	77.26
w/o GRPO (SFT only)	78.43	77.64
PatternICL (Qwen2.5-72B-Instruct)	**85.12**	**84.54**
w/o chain-of-thought	83.17	82.51
w/ unfinetuned DeBERTa	83.64	82.82
w/o GRPO (SFT only)	82.96	82.34
PatternICL (Qwen3-14B)	**81.45**	**81.02**
w/o chain-of-thought	79.87	79.13
w/ unfinetuned DeBERTa	78.87	78.24
w/o GRPO (SFT only)	79.48	79.12
PatternICL (Qwen3-32B)	**86.25**	**86.13**
w/o chain-of-thought	85.17	84.65
w/ unfinetuned DeBERTa	85.34	84.68
w/o GRPO (SFT only)	84.64	83.98

together, these results suggest that PatternICL improves the effectiveness of contextual evidence utilization by making better use of the prompt rather than by changing the backbone architecture or relying on more expensive inference.

4.3 Ablation Study

Table 2 shows ablation results under the Top-k 10-shot setting. All three components contribute significantly, with their relative importance varying by model.

Removing chain-of-thought, replacing fine-tuned DeBERTa with its unfine-tuned counterpart, or removing GRPO all hurts performance. The degradation is especially clear for smaller models, indicating that explicit reasoning, task-specific pattern representations, and reinforcement optimization each contribute to stronger pattern utilization. The ablations also show that these components play distinct roles rather than offering interchangeable

Table 3: The effect of reasoning on ICL. All model performances are reported under the top-k setting. Nums refers to the number of examples. Token refers to the number of output tokens generated by the model.

Nums	Method	GLUE Avg.	Avg.	Token
3	Qwen3-32B	80.73	80.31	**12.3**
	Qwen3-32B (Reasoning)	82.79	81.92	286.3
	Qwen3-32B (PatternICL)	**85.21**	**84.66**	56.6
10	Qwen3-32B	81.86	81.29	**12.7**
	Qwen3-32B (Reasoning)	83.28	82.65	442.6
	Qwen3-32B (PatternICL)	**86.25**	**86.13**	57.7
20	Qwen3-32B	81.04	80.56	**12.9**
	Qwen3-32B (Reasoning)	83.61	83.01	543.8
	Qwen3-32B (PatternICL)	**86.96**	**86.28**	55.4

sources of supervision. Chain-of-thought structures how the model translates selected examples into a final prediction, task-specific DeBERTa representations determine whether the training signal reflects pattern similarity rather than loose semantics, and GRPO sharpens preferences among multiple valid-but-unequal responses after SFT. The fact that performance degrades consistently after removing any one of them supports the view that PatternICL is a coordinated pipeline rather than a collection of loosely related tricks.

4.4 The Effect of Reasoning on In-Context Learning

Table 3 compares Qwen3-32B with reasoning enabled (temperature 0.6, top-p 0.95) against PatternICL. Enabling reasoning consistently improves over the non-reasoning baseline, indicating that explicit inference-time reasoning helps the model better exploit the provided examples.

However, this benefit comes at a substantial cost. At 10-shot, the reasoning model generates 442.6 tokens, compared with only 12.7 for the non-reasoning baseline, while PatternICL uses only $\sim$57 tokens and still performs better. These results show that explicitly training the model to identify and reuse relevant patterns is substantially more cost-effective than inference-time reasoning. This comparison underscores the difference between *teaching* a model to use context and *forcing* it to think longer at test time. Reasoning can partly recover missed pattern use, but only by spending substantially more inference computation on each input. PatternICL instead transfers this burden into training, allowing the model to produce a shorter yet more context-sensitive decision process at inference time.

5 Conclusion

In this paper, we propose PatternICL, a novel and structurally grounded two-stage training approach designed to fundamentally enhance and stabilize in-context learning capabilities in large language models. By organically combining pattern-aware supervised instruction tuning with a computationally efficient Group Relative Policy Optimization (GRPO) framework, PatternICL explicitly teaches models to systematically identify, retrieve, and exploit intricate example-level data patterns within the provided context. This methodical optimization paradigm eliminates the conventional and fragile reliance on hand-crafted prompt engineering or prohibitively expensive test-time reasoning procedures. Rigorous evaluations across 20 diverse natural language understanding datasets yield consistent, statistically significant performance gains over strong competitive baselines, while concurrently validating that embedding pattern-utilization behaviors directly during training offers superior operational cost-effectiveness compared to long-form inference-time chain-of-thought alternatives.

More broadly, our empirical insights and design methodology offer a fresh perspective that reframes the community's collective understanding of the underlying mechanics of in-context learning. Our core findings strongly suggest that a substantial part of the current ICL performance bottleneck is fundamentally not rooted in the absolute absence of downstream task knowledge within the model's parameters, but rather in the lack of reliable tracking and architectural mechanisms for explicitly aligning that latent internal knowledge with localized contextual evidence provided in the prompt. PatternICL successfully bridges this critical gap by transforming case-based retrieval, example selection quality, and contextual pattern reuse into first-class, verifiable optimization targets during the alignment phase. Ultimately, we hope this established paradigm helps reframe in-context learning as a highly controllable, observable, and explicitly alignable behavior that can be fundamentally mastered through training, rather than treating it merely as an unpredictable, black-box emergent capability that can only be indirectly encouraged at test time.

6 Limitations

PatternICL currently focuses on natural language understanding tasks, so its effectiveness for generative tasks remains unverified. Our pattern-aware supervision pipeline also relies on heuristics such as similarity thresholds, task-specific DeBERTa encoders, and hand-designed reward components, which may not transfer equally well across all domains. In particular, because the selection reward depends on a task-conditioned encoder, encoder bias or representation error may propagate into R_s and occasionally overestimate noisy matches. Although our qualitative analyses suggest that GRPO's joint optimization over structure, accuracy, and selection reduces overcommitment to isolated similarity errors, this dependence remains a limitation of the current framework. In addition, although cross-partition evaluation reduces leakage, the 20 benchmark datasets still cover a limited slice of ICL settings. Future work should investigate learned similarity signals, richer multi-example reasoning, and broader evaluations on generation-heavy tasks.

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.

Qiguang Chen, Libo Qin, Jinhao Liu, Dengyun Peng, Jiannan Guan, Peng Wang, Mengkang Hu, Yuhang Zhou, Te Gao, and Wanxiang Che. 2025. Towards reasoning era: A survey of long chain-of-thought for reasoning large language models. *arXiv preprint arXiv:2503.09567*.

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, and 1 others. 2024. Scaling instruction-finetuned language models. *Journal of Machine Learning Research*, 25(70):1–53.

Ona de Gibert, Naiara Pérez, Aitor García-Pablos, and Montse Cuadros. 2018. Hate speech dataset from a white supremacy forum. In *Proceedings of the 2nd Workshop on Abusive Language Online (ALW2)*, pages 11–20.

Hanyu Duan, Yixuan Tang, Yi Yang, Ahmed Abbasi, and Kar Yan Tam. 2024. Exploring the relationship between in-context learning and instruction tuning. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 3197–3210.

Yuxian Gu, Li Dong, Furu Wei, and Minlie Huang. 2023. Pre-training to learn in context. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4849–4870.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, and 1 others. 2025. Deepseek-r1 incentivizes reasoning in llms through reinforcement learning. *Nature*, 645(8081):633–638.

Shivanshu Gupta, Matt Gardner, and Sameer Singh. 2023. Coverage-based example selection for in-context learning. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 13924–13950.

Michael Hahn and Navin Goyal. 2023. A theory of emergent in-context learning as implicit structure induction. *arXiv preprint arXiv:2303.07971*.

Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. 2021. DeBERTa: Decoding-enhanced BERT with disentangled attention. In *International Conference on Learning Representations*.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*.

Shizhou Huang, Bo Xu, Yang Yu, Changqun Li, and Xin Alex Lin. 2025. Guidener: Annotation guidelines are better than examples for in-context named entity recognition. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 24159–24166.

Tushar Khot, Peter Clark, Michal Guerquin, Peter Jansen, and Ashish Sabharwal. 2020. Qasc: A dataset for question answering via sentence composition. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 8082–8090.

Yichuan Li, Xiyao Ma, Sixing Lu, Kyumin Lee, Xiaohu Liu, and Chenlei Guo. 2024. MEND: Meta demonstration distillation for efficient and effective in-context learning. 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.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Quanyu Long, Yin Wu, Wenya Wang, and Sinno Jialin Pan. 2024. Does in-context learning really learn? rethinking how large language models respond and solve tasks via in-context learning. In *First Conference on Language Modeling*.

Shayne Longpre, Le Hou, Tu Vu, Albert Webson, Hyung Won Chung, Yi Tay, Denny Zhou, Quoc V Le, Barret Zoph, Jason Wei, and 1 others. 2023. The flan collection: Designing data and methods for effective instruction tuning. In *International Conference on Machine Learning*, pages 22631–22648. PMLR.

Sewon Min, Mike Lewis, Luke Zettlemoyer, and Hannaneh Hajishirzi. 2022a. Metaicl: Learning to learn in context. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2791–2809.

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

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, and 1 others. 2022. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744.

Chaoxu Pang, Yixuan Cao, Qiang Ding, and Ping Luo. 2023. Guideline learning for in-context information extraction. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 15372–15389.

Keqin Peng, Liang Ding, Yancheng Yuan, Xuebo Liu, Min Zhang, Yuanxin Ouyang, and Dacheng Tao. 2024. Revisiting demonstration selection strategies in in-context learning. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9090–9101.

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.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3982–3992.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Li, Mingchuan Zhang, Y.K. Wu, Daya Guo, Xiao Bi, and 1 others. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*.

Oyvind Tafjord, Peter Clark, Matt Gardner, Wen-tau Yih, and Ashish Sabharwal. 2019. Quarel: A dataset and models for answering questions about qualitative relationships. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 7063–7071.

Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003*, pages 142–147.

Christopher Walker, Stephanie Strassel, Julie Medero, and Kazuaki Maeda. 2006. ACE 2005 multilingual training corpus. Linguistic Data Consortium, Philadelphia.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. 2019. Glue: A multi-task benchmark and analysis platform for natural language understanding. In *7th International Conference on Learning Representations, ICLR 2019*.

Siyin Wang, Chao-Han Yang, Ji Wu, and Chao Zhang. 2024. Bayesian example selection improves in-context learning for speech, text and visual modalities. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 20812–20828.

Jason Wei, Maarten Bosma, Vincent Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M Dai, and Quoc V Le. 2022a. Finetuned language models are zero-shot learners. In *International Conference on Learning Representations*.

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

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.

Sang Michael Xie, Aditi Raghunathan, Percy Liang, and Tengyu Ma. 2022. An explanation of in-context learning as implicit bayesian inference. In *International Conference on Learning Representations (ICLR)*.

Wenhao Yu, Dan Iter, Shuohang Wang, Yichong Xu, Mingxuan Ju, Soumya Sanyal, Chenguang Zhu, Michael Zeng, and Meng Jiang. 2023. Generate rather than retrieve: Large language models are strong context generators. In *International Conference for Learning Representation (ICLR)*.

Yichu Zhou and Vivek Srikumar. 2022. A closer look at how fine-tuning changes bert. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1046–1061.

A Supplementary Material

A.1 Dataset Partition Details

For completeness, we list the two dataset partitions used in cross-evaluation. Part A contains SST-2, QNLI, RTE, QQP, STS-B, hate_speech18, tweet-hate, quarel, openbookqa, and CoNLL03. Part B contains CoLA, MNLI, WNLI, MRPC, tweet-atheism, tweet-feminist, qasc, commonsense_qa, ACE04, and ACE05. The split is designed to distribute task types as evenly as possible across the two partitions while keeping the train/evaluation sets disjoint. This setting allows us to assess whether PatternICL learns a transferable pattern-sensitive inference behavior rather than overfitting to specific dataset conventions.

Table 4: Cross-evaluation dataset partitions.

Partition	Datasets
Part A	SST-2, QNLI, RTE, QQP, STS-B, hate_speech18, tweet-hate, quarel, openbookqa, CoNLL03
Part B	CoLA, MNLI, WNLI, MRPC, tweet-atheism, tweet-feminist, qasc, commonsense_qa, ACE04, ACE05

A.2 Instruction Format Details

PatternICL relies on a structured output format so that example usage becomes directly observable during both SFT and GRPO. For instructions containing relevant examples, the model first emits one or two `<sel>...</sel>` spans, where each span includes the positional index of the selected example, a short content excerpt, and the example label. For instructions without a suitable match, the model emits `<sel> none </sel>` before producing the final prediction. This format provides a compact interface between latent case retrieval and explicit reasoning, and it makes selection quality machine-verifiable during reinforcement learning.

In sentence-level tasks, the selected excerpt is typically a short clause or sentence fragment that uniquely identifies the example. In token-level tasks such as NER, the excerpt is chosen to preserve the local entity pattern that motivates the selection. Across both settings, the final prediction is placed after the reasoning segment so that the model first commits to contextual evidence and only then produces the task answer.

Illustrative sentence-level output
`<sel> 2 ... best screenplay. Positive. </sel>`
The input and the selected example both express a clearly favorable judgment, so the label is `Positive`.
Illustrative abstention output
`<sel> none </sel>` The provided examples do not contain a reliable matching pattern, so the model falls back to its own task understanding before predicting the label.

Figure 4: Simplified examples of the structured outputs used by PatternICL.

A.3 Training Configuration Details

In the SFT stage, we train each target LLM for 3 epochs with AdamW, using $\beta_1 = 0.9$, $\beta_2 = 0.999$, a batch size of 16, and a peak learning rate of 5×10^{-5} with linear warmup and decay. We use LoRA with rank 16 on a multi-GPU accelerator cluster. For each dataset, we cap the constructed instruction count at 5k examples, upsampling smaller datasets when necessary, which yields 50k instructions per partition.

In the GRPO stage, we reuse the same instruction inputs as in SFT and sample $G = 8$ candidate outputs per input. We set the clipping coefficient to $\epsilon = 0.2$, the KL penalty to $\beta = 0.04$, and train for 1 epoch with a learning rate of 1×10^{-5}. We found this lightweight RL stage sufficient to sharpen example-selection behavior without destabilizing the output format learned during SFT.

Table 5: Main training hyperparameters.

Stage	Epochs	Learning Rate	Batch	Key Settings
SFT	3	5×10^{-5}	16	LoRA rank 16; multi-GPU accelerator cluster; 5k instructions per dataset
GRPO	1	1×10^{-5}	–	$G = 8$, $\epsilon = 0.2$, $\beta = 0.04$

A.4 Reward Interpretation

The composite reward combines format validity, selection quality, and prediction accuracy. This

decomposition is useful because these three objectives are related but not identical. A response may predict the correct label while selecting poor evidence, or it may select a good example but fail to convert that evidence into a correct prediction. By rewarding these dimensions separately, GRPO can prefer responses that are not only accurate, but also more faithful to the intended pattern-based inference process.

The selection reward is especially important in our setting because it determines whether the model learns to abstain when no good pattern match exists. Without this signal, the policy could be biased toward always selecting some example regardless of its actual relevance. Rewarding `<sel> none </sel>` when no reliable match exists encourages a more calibrated use of context and prevents spurious dependence on irrelevant demonstrations. At the same time, this reward inherits the strengths and weaknesses of the task-conditioned encoder used to define similarity. If the encoder assigns spuriously high similarity to a distractor, that bias may affect R_s. In practice, however, the policy is not optimized against selection reward alone: format validity and final-task accuracy provide additional constraints, and the qualitative patterns in Table 9 suggest that the resulting policy is more robust to distractor-heavy contexts than vanilla ICL.

Table 6: Interpretation of the three reward components.

Reward	Signal Type	Encourages	Discourages
R_f	Binary format check	Well-formed structured outputs	Missing tags, malformed selection spans, invalid output layout
R_s	Similarity-based selection score	Choosing relevant examples or abstaining when none match	Selecting arbitrary examples or forcing comparison to irrelevant cases
R_a	Binary task accuracy	Correct final predictions	Correctly formatted but wrong answers

A.5 Prompt Template Snippets

To illustrate how PatternICL operationalizes pattern-sensitive ICL, we provide abridged prompt snippets for representative sentence-level and token-level tasks. These snippets are not full training instances; rather, they highlight the structural elements that remain consistent across templates, including the task description, indexed in-context examples, and the explicit expectation that the model should ground its answer in example-level patterns.

Sentence-level template (abridged)
Task: Determine the sentiment of the input using the data patterns shown in the examples.
Example 1: "A flat and uninspired sequel." Label: Negative
Example 2: "One of the year's warmest and smartest comedies." Label: Positive
Example 3: "Technically polished but emotionally empty." Label: Negative
Input: "A funny and surprisingly heartfelt film."
Output format: first select the most relevant example(s) with `<sel>...</sel>`, then explain, then predict.

Token-level template (abridged)
Task: Extract entities according to the labeling pattern implied by the examples.
Example 1: "Linux was released in 1991." Output: none
Example 2: "Google acquired YouTube." Output: Google, YouTube
Example 3: "Windows 11 was announced." Output: none
Input: "Android expanded rapidly in Asia."
Output format: select relevant example(s) or output `<sel> none </sel>` before prediction.

Figure 5: Abridged prompt-template snippets for two task types.

A.6 Illustrative Reward Computation

Table 7 provides simplified examples of how the composite reward behaves under different response types. The similarity values shown for R_s are illustrative rather than tied to a specific evaluation instance, but they follow the same logic as the reward definition in the main paper. The examples highlight that PatternICL does not simply reward correctness; it also rewards selecting strong evidence and abstaining when no trustworthy match exists.

Table 7: Illustrative reward breakdowns for representative response types.

Response Type	R_f	R_s	R_a	Total
Valid format, strong selected match, correct prediction	1	0.91	1	2.91
Valid format, `<sel> none </sel>` when no good match exists, correct prediction	1	1.00	1	3.00
Valid format, weak selected match, wrong prediction	1	0.28	0	1.28
Valid format, forced irrelevant selection when no match exists	1	-0.50	0/1	0.50/1.50

A.7 Selection Accuracy Evaluation Protocol

A direct way to probe whether GRPO improves pattern-sensitive behavior is to evaluate selection accuracy independently of final task accuracy. Given an instruction with indexed in-context examples, one can define an oracle set by taking the examples whose task-conditioned similarity to the input exceeds the positive threshold used in data construction (0.8 in our setup). The model prediction inside `<sel>...</sel>` can then be mapped to a selected index set and compared against this oracle set.

Under this protocol, standard retrieval metrics such as precision, recall, and F1 can be computed over selected-example indices. We recommend reporting these metrics at multiple training checkpoints to distinguish three different behaviors: (1) higher final-task accuracy caused by better example selection, (2) higher final-task accuracy caused primarily by better downstream reasoning after selection, and (3) spurious reward improvement caused by format compliance without corresponding evidence quality. We leave a full checkpoint-level study of this probing signal to future work, but we include the protocol here because it provides a clean lens on whether the model is actually learning to choose better contextual evidence.

A.8 Representation Diagnostics Protocol

Because our notion of pattern similarity is grounded in task-conditioned geometry rather than generic sentence semantics, an additional diagnostic is to inspect whether supervised fine-tuning changes the structure of the encoder space in a task-relevant way. A practical protocol is to sample instances from a representative dataset, encode them with both the unfine-tuned encoder and the task-specific encoder, and then compare neighborhood purity or low-dimensional visualizations under the two spaces.

We stress that such plots should be treated as qualitative diagnostics rather than standalone proof. In our setting, a more reliable quantitative companion would be to measure whether nearest neighbors under the task-specific encoder exhibit stronger label-transfer consistency, stronger agreement with the oracle set defined above, or both. We leave this analysis for future work, but include the protocol here because it directly tests the representational assumption that underlies our selection reward.

A.9 Training-vs-Inference Cost Perspective

PatternICL introduces additional offline training cost because it requires both SFT and a lightweight GRPO stage. However, this cost is paid once during model adaptation, whereas inference-time reasoning incurs additional token generation for every request. The comparison in Table 3 suggests that PatternICL trades one-time training expense for lower recurring inference cost: the trained model keeps outputs relatively short while still outperforming a reasoning-enabled baseline.

This cost profile is important for practical deployment. If a model is expected to answer many ICL queries under similar task families, improving pattern use during training may be more economical than repeatedly invoking long reasoning traces at inference time. We therefore view PatternICL as shifting part of the computational burden from repeated online deliberation to reusable offline alignment.

A.10 Case-Oriented Analysis

Table 8 presents a case-oriented analysis derived from the motivating NER example in the main paper. The point of this table is not to reproduce an exact model transcript, but to make explicit the decision boundary that PatternICL is intended to learn. In this example, the critical signal is a *negative pattern*: operating-system mentions are consistently excluded from the entity outputs in the demonstrations, so a pattern-aware model should transfer that exclusion rule to `Android`.

Table 8: Case-oriented analysis of the motivating NER example.

Aspect	Vanilla ICL Tendency	PatternICL Target Behavior
Evidence used	Relies on generic knowledge that `Android` can be an entity-like term	Prioritizes the demonstration pattern that operating systems are excluded
Example handling	May compare to superficially related tokens without checking label-transfer consistency	Selects the example(s) that best support the exclusion rule, or abstains if evidence is weak
Final action	Predicts `Android` as an entity	Omits `Android` from the entity output
Why this matters	Produces plausible but context-insensitive predictions	Makes the prediction traceable to the in-context evidence

A.11 Qualitative Discussion

We observed two recurring qualitative patterns during development. First, strong base models often produce correct answers while citing examples that are only loosely related to the target input. This behavior is superficially successful but undermines the reliability of ICL as a controllable adaptation mechanism. PatternICL reduces this mismatch by explicitly rewarding more faithful contextual grounding.

Second, when no strong pattern match exists in the prompt, vanilla models frequently force a comparison anyway, leading to brittle or overconfident predictions. By contrast, the `<sel> none </sel>` option allows PatternICL to express that the current context is not informative enough and to fall back on parametric knowledge more transparently. We view this abstention behavior as an important practical property, especially in mixed-quality retrieval settings where the prompt may contain distractors or ambiguous cases.

Table 9: Typical qualitative differences between vanilla ICL and PatternICL.

Scenario	Vanilla ICL	PatternICL
Weak contextual match	Often forces comparison to a marginally related example	More likely to emit `<sel> none </sel>` and avoid spurious copying
Correct answer with weak grounding	May predict correctly while citing a loose example	Tends to select a closer case and align the reasoning with it
Distractor-heavy context	More sensitive to superficial semantic overlap	More selective about which examples are actually reused

A.12 Failure Taxonomy

Although PatternICL improves contextual grounding, it does not eliminate all failure modes. Table 10 summarizes several representative error types that remain plausible under our framework. We include this taxonomy to make the method's current boundary conditions explicit and to highlight directions where better data construction or richer decision policies may be needed.

Table 10: Representative failure types for PatternICL.

Failure Type	Description	Observed Behavior	Likely Cause / Future Direction
Semantic distractor	Surface lexical overlap dominates a weaker but more relevant pattern cue	The model selects a nearby example with matching keywords but mismatched label-transfer logic	Harder negative mining or contrastive selection supervision may improve robustness
Over-abstention	Borderline-relevant evidence is treated as unusable	The model emits `<sel> none </sel>` even when a weak but valid analogy exists	Fixed similarity thresholds may be too rigid; a learned gate could adapt abstention behavior
False positive selection	A distractor receives spuriously high similarity under the task-conditioned encoder	The selection is well-formed but grounded in the wrong contextual evidence	Better encoder calibration and additional diagnostics on neighborhood purity could reduce this risk
Multi-step mismatch	The required reasoning depends on composing multiple weak cues across examples	The model selects one plausible case but fails to aggregate distributed evidence	Extending the framework beyond a small selection budget may help on composition-heavy tasks