Cognitively Decoupled Meta-Induction for Out-of-Distribution Generalization

Anonymous ACL submission

Abstract

In-context learning (ICL) tackles downstream tasks conditioned on demonstrations. This paradigm relies heavily on neighbor-based matching between queries and demonstrations, limiting out-of-distribution (OOD) performance. For better generalization, rule-based inductive reasoning derives task-level regularities, which we term task-level inductive representations (TIR), before query inference. However, existing methods lack generalizable inductive ability across diverse tasks. Inspired by meta-learning, we propose Cognitive-Decoupling meta-Induction (CoDI) for TIR-based inductive reasoning in general task. CoDI comprises a *Cognitive-Decoupling Framework* for query-blind TIR induction, and a *Dual-Loop Meta-Inductive Learning* method for cross-task inducer optimization. Empirically, CoDI achieves substantial performance gains on five ID and OOD benchmarks, and persists OOD gains without explicit induction at inference. Code is available at `https://anonymous.4open.science/r/CoDI`.

1 Introduction

In-context learning (ICL) is a prevalent paradigm for tackling downstream tasks, allowing large language models (LLMs) to solve new queries conditioned on limited contextual labeled demonstrations (Brown et al., 2020). Despite achieving promising performance, LLMs inherently tend to perform nearest neighbor matching, with strong dependence on sample similarity between queries and demonstrations (Li et al., 2025). When encountering distribution shifts and unseen tasks, such similarity-dominated shallow reasoning often suffers severe performance degradation (Min et al., 2022b; Wang et al., 2024).

To address this limitation, rule-based inductive reasoning derives generalizable principles for query inference, which we name task-level inductive representations (TIR). The difference between ICL and TIR-based reasoning is shown in Figure 1. The former relies on surface matching with single demonstrations, while the latter approaches learn universal task patterns from all supports for better transfer. Some works introduce such induction through chain-of-thought and structured reasoning (Wei et al., 2022; Zhou et al., 2024). Yet, because rule extraction and query prediction share the same context, the resulting reasoning tends to be query-conditioned and hard to reuse across instances (Li et al., 2025; Zhou et al., 2024). A complementary direction pursues task-level rules via automated prompt optimization (Zhou et al., 2023; Fernando et al., 2024; Agrawal et al., 2026), but such method optimizes rules per task and must re-optimize on transfer.

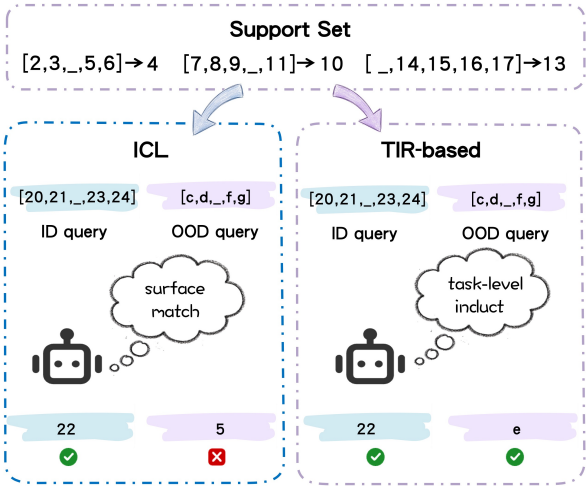

Figure 1: ICL (left) surface-matches and fails OOD. TIR-based reasoning (right) induces a task-level rule that generalizes to both ID and OOD.

In comparison, humans follow the induction-before-execution paradigm even on unfamiliar tasks (Lake et al., 2015), owing to inductive learning accumulated from past experience. We refer to this cross-task TIR inference capability as meta-induction. Recent work reveals that LLMs gener-

ally lack this capability (Li et al., 2025), as their inductive behavior is dominated by input-output memorization rather than reusable rules. However, instilling this capacity in LLMs presents two challenges: i) ensuring meta-induction generalizes across tasks without overfitting to instance-specific patterns or task-local rules; ii) learning TIR in the absence of ground-truth labels and established evaluation criteria.

To address these challenges, we propose **Co**gnitively **D**ecoupled meta-**I**nduction (**CoDI**), a TIR-based inductive reasoning method in general task. As shown in Figure 2, CoDI comprises a *Cognitive-Decoupling Framework* and *Dual-Loop Meta-Inductive Learning* method. In the Cognitive-Decoupling Framework, the inducer, implemented by a query-blind Strategy Generator (SG), induces TIR solely from supports, and the executor, implemented by an Answer Solver (AS), executes TIR for each query. Based on above framework, Dual-Loop Meta-Inductive Learning optimizes SG across tasks via meta-RL, guided by execution feedback without ground-truth TIR labels, where an inner loop applies candidate TIR and an outer loop refines induction via two complementary rewards for extrinsic correctness and intrinsic certainty.

Experiments on BBH, BIG-Bench, MATH-500, StrategyQA, and ReClor show that CoDI (Qwen3-4B) outperforms few-shot ICL on in-distribution (ID) and out-of-distribution (OOD) evaluations. On BIG-Bench, CoDI achieves $+120\%$ and $+18\%$ gains on ID and OOD splits, respectively. Furthermore, a smaller trained SG surpasses a larger untrained one on OOD tasks, confirming TIR quality as an independently optimizable capability orthogonal to model scale. Notably, OOD gains persist even without the induction step at inference, indicating that meta-inductive training internalizes inductive priors into model parameters.

In summary, our contributions are as follows:

- We formulate TIR-based inductive reasoning as a learning problem that requires inferring reusable task rules from support examples alone before query execution.
- We present CoDI to instill meta-induction without strategy supervision, decoupling rule induction from query execution and optimizing inducers across tasks via reinforcement learning on execution feedback.
- Experiments on five benchmarks show CoDI outperforming few-shot ICL in- and out-of-distribution, with smaller trained inducers beating larger untrained ones and OOD gains persisting without inference-time induction.

2 Problem Statement

In this section, we formalize the task of TIR-based inductive reasoning.

Notation. Let $\mathcal{T}$ denote a task space. Each task $\tau \sim P(\mathcal{T})$ has a K-shot demonstration set $\mathcal{D}^{(\tau)}_{\text{supp}} = \{(x_i, y_i)\}_{i=1}^{K}$ and query set $\mathcal{D}^{(\tau)}_{\text{qry}}$ of pairs (x, y^*). The TIR of τ is a natural-language strategy s induced from $\mathcal{D}^{(\tau)}_{\text{supp}}$, and π_{ind} denotes the inducer policy, where ind abbreviates *inducer*.

Task. Given $\mathcal{D}^{(\tau)}_{\text{supp}}$ and query x, TIR-based inductive reasoning samples $s \sim \pi_{\text{ind}}(\cdot \mid \mathcal{D}^{(\tau)}_{\text{supp}})$ as the TIR and uses an executor to predict $\hat{y}(s, x)$ for y^*. The optimization objective is

$$\mathcal{J} = \max_{\pi_{\text{ind}}} \mathbb{E}_{\tau,\, \mathcal{D}^{(\tau)}_{\text{supp}},\, (x, y^*)} \Big[\mathbb{E}_{s \sim \pi_{\text{ind}}(\cdot \mid \mathcal{D}^{(\tau)}_{\text{supp}})} \big[\mathbf{1}\big[\hat{y}(s, x) = y^*\big] \big] \Big]. \tag{1}$$

where $\mathbf{1}[\cdot]$ is the indicator function, the outer expectation is over tasks, supports, and queries, and the inner expectation is over $s \sim \pi_{\text{ind}}$.

3 CoDI

In this section, we present the proposed CoDI for TIR-based inductive reasoning, built on a cognitive-decoupling architecture (§3.2) and a dual-loop meta-inductive learning scheme (§3.3).

3.1 Overview

The proposed CoDI employs a cognitive-decoupling induction–execution framework: the query-blind inducer abstracts TIR and the executor solves new query based on it. To enhance the inductive reasoning ability for general tasks, a dual-loop meta-inductive learning approach is proposed: the inner loop is executed based on TIR, and improves the answers through ICL; the outer loop is based on the feedback signal from the inner loop execution, and optimizes the abstracts of TIR through RL. This design is overviewed in Figure 2.

3.2 Cognitive-Decoupling Framework

CoDI cognitively decouples induction and execution through two modules. The Strategy Generator (SG) induces TIR from few-shot demonstrations

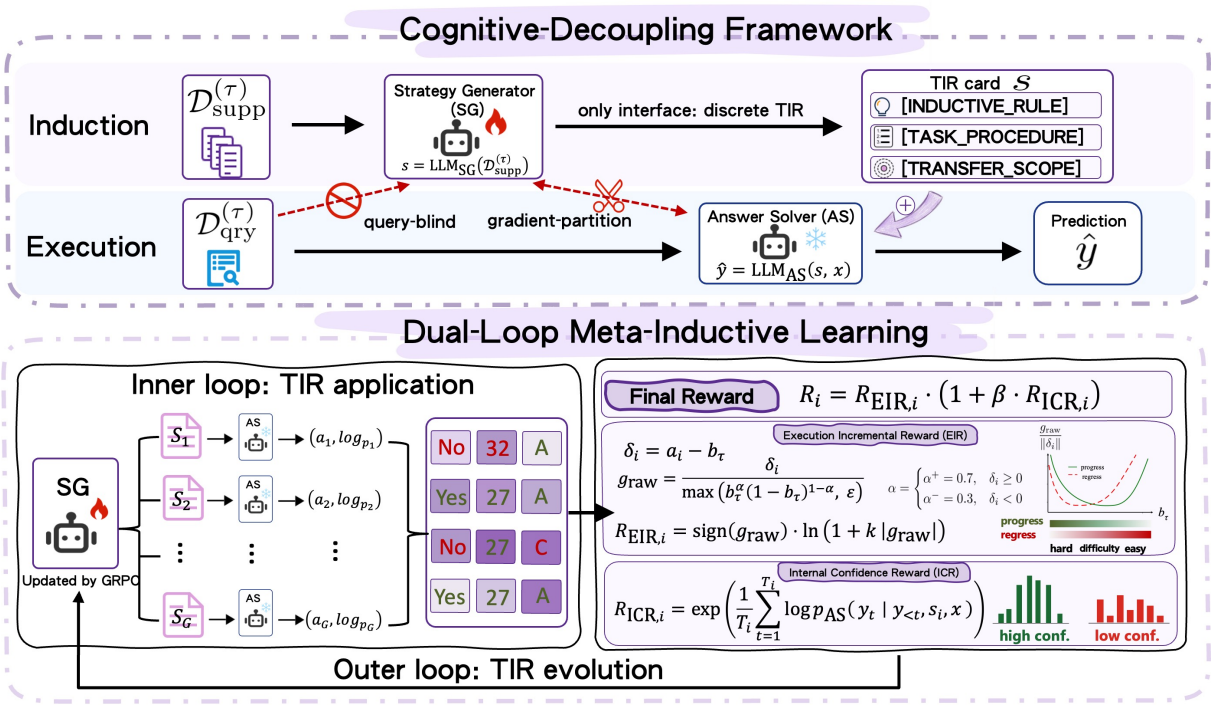

Figure 2: Overview of CoDI. The **Cognitive-Decoupling Framework** induces query-blind TIR with SG and executes each query with AS. **Dual-Loop Meta-Inductive Learning** applies candidate TIRs in the **inner loop** and optimizes SG across tasks in the **outer loop** via GRPO with EIR and ICR.

alone, and the Answer Solver (AS) answers each query by executing the TIR.

Inducer (SG). The SG induces a natural-language strategy s from $\mathcal{D}_{\text{supp}}^{(\tau)}$ alone, where s serves as the TIR of task τ. The query-blind interface isolates reusable task regularities from query-specific cues. Specifically, s comprises `[INDUCTIVE_RULE]`, `[TASK_PROCEDURE]` (3–6 executable steps), and `[TRANSFER_SCOPE]` (Appendix B). Each section plays a distinct role: `[INDUCTIVE_RULE]` states verifiable task-family regularities; `[TASK_PROCEDURE]` operationalizes them as executable steps; and `[TRANSFER_SCOPE]` specifies applicability boundaries. This semi-structured format yields comparable rollouts across candidates and enables format validation during training. The induction step is formalized as

$$s = \text{LLM}_{\text{SG}}(\mathcal{D}_{\text{supp}}^{(\tau)}). \tag{2}$$

Executor (AS). The AS applies s to query x via the concatenated prompt (s, x). It remains frozen throughout training; gradient partitioning blocks back-propagation to SG, so SG is updated only through the discrete strategy interface. The execution step is formalized as

$$\hat{y} = \text{LLM}_{\text{AS}}(s,\ x). \tag{3}$$

Gradient partitioning confines updates to π_{SG}. Without this barrier, gradients from query-conditioned execution would leak into the inducer and blur the TIR interface, conflating reusable task regularities with instance-specific matching. Because AS remains frozen, execution accuracy reflects each TIR's transfer value rather than co-adaptation between modules.

3.3 Dual-Loop Meta-Inductive Learning

Building on the cognitive-decoupling framework, dual-loop meta-inductive learning evolves TIR when no strategy labels are available. Execution under a frozen AS provides a weak, task-agnostic supervision signal for π_{SG}. π_{SG} is first warmed up via SFT on teacher demonstrations to satisfy the schema. During RL, format-invalid rollouts receive zero reward and are excluded from AS execution.

Inner loop: TIR application. At each training step, SG induces G TIR candidates $\{s_i\}_{i=1}^{G}$ from the support set $\mathcal{D}_{\text{supp}}^{(\tau)}$. The frozen AS applies each s_i to query x through in-context prompting on (s_i, x), without updating parameters. This step tests whether each induced TIR generalizes beyond the few-shot demonstrations. Execution yields a prediction and per-token log-probabilities; the re-

sultant accuracy a_i and decoding distribution serve as reward signals for the outer loop.

Outer loop: TIR evolution. The outer loop optimizes π_{SG} via cross-task GRPO with two complementary rewards:

- **Execution Incremental Reward (EIR)** measures difficulty-adaptive accuracy gain. Let $a_i \in [0,1]$ be the answer accuracy of AS under s_i, $\delta_i = a_i - b_\tau$ its deviation from a per-task cached baseline b_τ (fixed before the G-rollout group; updated via exponential moving average to prevent within-group leakage), and g_{raw} the difficulty-normalized gain (log-smoothing constant k; Appendix E):

$$\delta_i = a_i - b_\tau, \quad g_{\text{raw}} = \frac{\delta_i}{\max\big(b_\tau^{\alpha}(1-b_\tau)^{1-\alpha},\ \varepsilon\big)} \tag{4}$$

  where $\alpha = \alpha^+$ if $\delta_i \geq 0$, else α^- ($\alpha^+ > \alpha^-$; Appendix A.3):

$$R_{\text{EIR},i} = \text{sign}(g_{\text{raw}}) \cdot \ln\big(1 + k\,|g_{\text{raw}}|\big). \tag{5}$$

  For positive δ_i, EIR sets $\alpha{=}\alpha^+$, down-weighting gains on easy tasks; for negative δ_i, it sets $\alpha{=}\alpha^-$, amplifying penalties on high-baseline regressions (Appendix A.3).

- **Internal Confidence Reward (ICR)** measures the extent to which s_i concentrates AS's decoding distribution, distinguishing systematic accuracy from chance-level correctness. $R_{\text{ICR},i} \in (0,1]$ is the geometric mean of per-token probabilities under AS, normalized by output length T_i:

$$R_{\text{ICR},i} = \exp\Big(\frac{1}{T_i}\sum_{t=1}^{T_i} \log p_{\text{AS}}\big(y_t \mid y_{<t}, s_i, x\big)\Big). \tag{6}$$

EIR and ICR are combined via multiplicative gating:

$$R_i = R_{\text{EIR},i} \cdot \big(1 + \beta \cdot R_{\text{ICR},i}\big), \tag{7}$$

where β is set in Appendix E. The factor $(1{+}\beta R_{\text{ICR},i}) > 0$ up-weights positive EIR proportionally to prediction confidence without reversing the sign of negative EIR. EIR normalizes rewards by per-task difficulty so that π_{SG} is not biased toward easy tasks. ICR complements EIR by up-weighting strategies whose frozen AS decoding is concentrated, separating systematic correctness from lucky hits. Together, EIR and ICR supply complementary, label-free signals for meta-inductive training; the full SFT→RL pipeline and hyperparameters are in Appendix A.

4 Experiments

4.1 Experimental Setup

Baselines. **FS** (few-shot ICL) prepends K demonstrations for decoding (Dong et al., 2024). **S-FS** (strategy-based few-shot) induces a strategy before answering (Gao et al., 2024), sharing the query-blind interface without RL. **MetaICL** (Min et al., 2022a) performs cross-task meta-training under equal compute without explicit induction. **GEPA** (Agrawal et al., 2026) performs inference-time prompt optimization. **Gemini-3-flash** and **GPT-5** are closed-source upper bounds. See Appendix A for details.

Datasets and Metrics. Training data are drawn from **StrategyBench** (Anonymous, 2026), a curated subset of **BIG-Bench** (Srivastava et al., 2023) filtered along five inductibility criteria (Appendix I.1) with teacher-scored SFT demonstrations; see Appendix I for construction. Evaluation spans **BIG-Bench ID/OOD** (task-level splits defined in Appendix J), **BBH** (Suzgun et al., 2023), **MATH-500** (Lightman et al., 2023), **StrategyQA** (Geva et al., 2021), and **ReClor** (Yu et al., 2020), with exact-match accuracy throughout. $\Delta\%$ denotes relative change from same-backbone FS: $\Delta = (\text{score} - \text{FS})/\text{FS} \times 100\%$; Avg. in Table 1 averages the six benchmark columns.

4.2 Main Results

Table 1 reports the main results, from which three findings emerge.

Cross-task training alone cannot yield TIR. Under equal training budgets, MetaICL scores below the untrained FS baseline on OOD (28.67 vs. 53.33, Qwen3-4B) and trails CoDI by over 34 macro-average points, showing cross-task parameter updates alone fail to produce reusable task-level strategies. GEPA improves over FS on in-distribution tasks, but its gains derive from query-conditioned prompt evolution rather than query-decoupled strategy induction, placing it structurally apart from the reusable TIR that CoDI targets.

Query-blind interface alone is insufficient. To disentangle interface from meta-inductive training, we compare S-FS and FS under the same backbone. S-FS retains CoDI's query-blind interface

Type	Model	Method	BBH (Δ%)	BIG-Bench ID (Δ%)	BIG-Bench OOD (Δ%)	MATH-500 (Δ%)	StratQA (Δ%)	ReClor (Δ%)	Avg.
w/o Training	Gemma-3-4B	FS	37.41	26.00	45.00	54.00	58.95	48.00	44.89
		S-FS	25.00 −33%	18.33 −30%	28.00 −38%	60.00 +11.1%	59.97 +1.7%	46.40 −3.3%	39.62
	Qwen3-4B	FS	35.56	29.33	53.33	63.13	69.97	83.60	55.82
		S-FS	27.78 −22%	30.00 +2%	38.33 −28%	65.20 +3.3%	64.19 −8.3%	81.20 −2.9%	51.12
		GEPA[§]	79.59	61.65	56.88	61.20	71.32	79.80	68.41
	Llama-3.1-8B	FS	31.11	20.00	44.33	43.40	70.60	39.40	41.47
		S-FS	21.30 −32%	13.33 −33%	23.00 −48%	37.80 −12.9%	53.13 −24.7%	40.60 +3.0%	31.53
	Gemini-3-flash[†]	FS	95.37	84.67	84.00	65.93	83.50	84.53	83.00
		S-FS	91.67 −4%	76.00 −10%	78.67 −6%	60.60 −8.1%	87.29 +4.5%	85.00 +0.6%	79.87
	GPT-5[†]	FS	94.44	80.33	73.67	71.40	85.54	97.93	83.89
		S-FS	88.89 −6%	63.00 −22%	61.67 −16%	68.00 −4.8%	97.93 +14.5%	98.00 +0.1%	79.58
w/ Training	Gemma-3-4B	FS	40.00 +7%	31.33 +21%	50.33 +12%	61.60 +14.1%	69.72 +18.3%	49.20 +2.5%	50.36
		CoDI	**42.67** +14%	**43.00** +65%	**55.67** +24%	**62.20** +15.2%	**67.98** +15.3%	**65.60** +36.7%	**56.19**
	Qwen3-4B	FS	40.93 +15%	41.33 +41%	55.56 +4.2%	65.80 +4.2%	73.53 +5.1%	85.80 +2.6%	60.49
		CoDI	**82.22** +131%	**64.67** +120%	**63.00** +18%	**76.15** +20.6%	**77.44** +10.7%	**86.60** +3.6%	**75.01**
		MetaICL	33.52	32.00	28.67	16.80	58.95	72.00	40.32
	Llama-3.1-8B	FS	32.22 +4%	22.11 +11%	45.56 +3%	50.00 +15.2%	81.51 +15.5%	45.60 +15.7%	46.17
		CoDI	**41.67** +34%	**23.00** +15%	**48.33** +9%	**47.80** +10.1%	**73.94** +4.7%	**48.00** +21.8%	**47.12**

Table 1: Exact-match accuracy comparing CoDI with FS, S-FS, GEPA, MetaICL, and closed-source APIs on BBH, BIG-Bench ID/OOD, MATH-500, StrategyQA, and ReClor (w/o and w/ training). $\Delta\% = (\text{score} - \text{FS})/\text{FS} \times 100$; red/blue = gain/loss. [†]Unfine-tuned; Gemini-3-flash = `gemini-3-flash-preview`. [§]GEPA: `gemini-3-flash-preview` optimizer, Qwen3-4B executor, four reflective iterations.

but excludes cross-task RL. On OOD, S-FS trails FS across all three backbones (up to −48% relative drop on Llama-3.1-8B) and remains far below trained CoDI, showing query-blind decoupling alone is insufficient for generalizable TIR.

Explicit TIR-aligned training drives systematic gains. CoDI consistently outperforms all baselines under full TIR-aligned training. On Qwen3-4B, it raises OOD accuracy by +18% over the untrained FS baseline, with especially pronounced gains on BBH (+131%) and BIG-Bench ID (+120%). Positive OOD improvements hold across all three open-source backbones and extend to MATH-500, StrategyQA, and ReClor. These results confirm that optimizing explicitly for TIR generalizes task-level strategy induction across diverse backbones and domains.

4.3 Ablation Study

We ablate EIR/ICR, training pipeline, decoupling, and TIR format on Qwen3-4B.

Reward design and training pipeline. We analyze the effect of reward design (EIR/ICR), training stages (SFT and RL), and architectural decoupling on Qwen3-4B. As shown in Table 2, the full model achieves 63.00 OOD accuracy. Removing ICR (β=0) preserves ID but reduces OOD by 9.3 pp, indicating ICR supports cross-task generalization. Without EIR, ID falls to 49.00, as rewards without difficulty-adaptive normalization conflate task difficulty with strategy quality. Raw-accuracy and LLM-judge rewards, SFT-only training, and end-to-end non-decoupled training all trail substantially (OOD ≤57.33 and ≤40.33), indicating that neither alternative rewards nor query-coupled optimization suffices for generalizable TIR induction.

Configuration	ID	BBH	OOD
CoDI (EIR+ICR, full)	**64.67**	**82.22**	**63.00**
Reward component ablation			
w/o ICR (β=0, EIR only)	62.67	69.81	53.67
w/o EIR (ICR only)	49.00	69.07	53.67
w/ raw Acc reward	31.67	48.15	56.67
w/ LLM Judge reward	49.00	69.44	57.33
Training pipeline ablation			
w/o SFT (RL only)	60.67	78.70	55.00
w/o RL (SFT only)	31.67	33.89	40.00
Decoupled architecture			
w/o decoupled arch. (e2e training)	33.67	38.52	40.33

Table 2: Qwen3-4B ablation of reward components, training stages, and decoupled architecture.

Strategy format. Table 3 compares TIR formats. The semi-structured schema outperforms free-form and CoT-style alternatives across ID, BBH, OOD, and ICR. Among its three sections, removing `[TASK_PROCEDURE]` collapses OOD to 2.67 and thus has the largest impact. Moreover, ablating `[INDUCTIVE_RULE]` or `[TRANSFER_SCOPE]` lowers OOD by over 20 pp each, showing that both the induced rule and its transfer scope are required for generalization.

Strategy Format	ID	BBH	OOD	ICR
CoDI (semi-structured)	**64.67**	**82.22**	**63.00**	**84.73**
Free-form (no schema)	46.00	53.52	40.00	75.59
CoT-style (implicit structure)	59.56	72.84	49.33	75.57
Semi-structured section ablation				
w/o [INDUCTIVE_RULE]	42.33	49.44	39.67	74.27
w/o [TASK_PROCEDURE]	1.33	1.48	2.67	71.67
w/o [TRANSFER_SCOPE]	31.67	36.85	33.67	80.35

Table 3: Qwen3-4B ablation comparing alternative TIR formats and individual schema sections.

5 Analysis

5.1 What Drives the Gains?

Gains exceed scale. Analysis reveals that OOD improvements primarily reflect TIR optimization rather than pretrained capacity alone. With AS fixed to Qwen3-4B, untuned SG OOD accuracy rises from 0.12 at 1.7B to 0.57 at 32B, confirming that scale remains helpful. Trained CoDI-4B nonetheless reaches 0.62 and exceeds the 32B baseline by 9% despite 8× fewer parameters (Figure 3). Because the gap persists after untuned scaling saturates, TIR quality functions as a distinct optimization target orthogonal to model size.

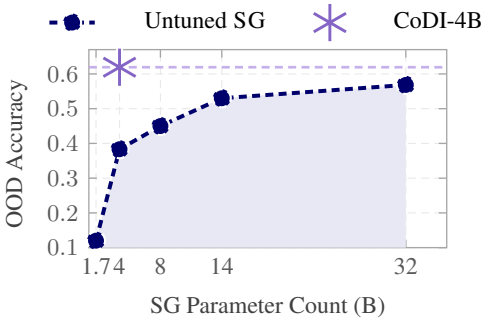

Figure 3: Trained CoDI-4B exceeds untuned 32B SG on BIG-Bench OOD with 8× fewer parameters, supporting TIR optimization over scale alone for OOD gains.

Induction becomes cross-example. We next test whether CoDI induces rules from multiple supports rather than copying individual examples. If rules still depended on single demonstrations, leave-one-out (LOO) removal should strongly alter [INDUCTIVE_RULE] semantics. On matched evaluation tasks, we regenerate each induced rule after removing one of $K=3$ supports and measure embedding shift (Appendix H). Mean LOO sensitivity drops from 0.325 to 0.087 (-73.4%) across ten task types (Figure 4). Accordingly, the gains align with cross-example abstraction rather than instance memorization.

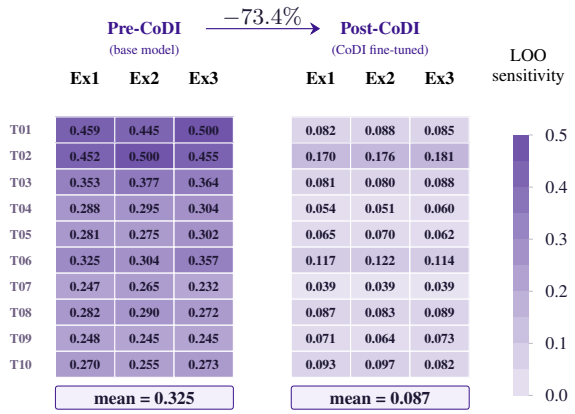

Figure 4: Mean LOO rule sensitivity drops 73.4% after training, supporting cross-example abstraction.

5.2 What Does Meta-Induction Reshape?

Reasoning turns rule-directed. Analysis reveals that OOD gains co-occur with shorter, rule-directed traces over longer chain-of-thought. We collect <think> spans from few-shot evaluation with thinking enabled on matched OOD instances and assign E/D/P/R sentence labels (Appendix D). OOD accuracy rises 104% while trace length falls 21%. Example-recitation units fall 51% and D→P transitions rise 42%, so supports anchor rule formation rather than serve as copied text (Figure 5). We vary K and compare CoDI with few-shot ICL on BIG-Bench OOD (Figure 6). CoDI leads by 12 pp at $K=1$, and the margin narrows as ICL gains additional demonstrations. Accordingly, OOD gains reflect a parametric meta-inductive prior rather than richer demonstration contexts alone.

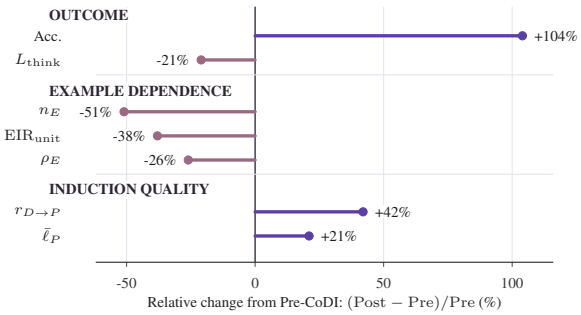

Figure 5: Post-CoDI raises OOD accuracy (+104%) while shortening traces (-21%) and example recitation (-51%), refuting a length-driven account and supporting rule-directed reasoning.

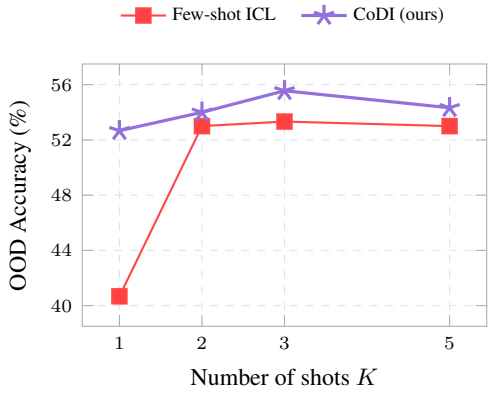

Figure 6: BIG-Bench OOD accuracy (%) vs. shot count K on Qwen3-4B (CoDI vs. few-shot ICL.)

Strategies favor rule-based modes. We further find that performance gains concentrate on rule-oriented induction modes rather than extending uniformly across all strategy types. Gemini-3-flash scores Pre- and Post-CoDI strategies on six induction dimensions (Appendix C). Post-CoDI improves on Constraint (+1.50), Procedural (+1.07), and Schema (+0.72) but declines on Analogical (-0.63) and Pattern (-0.62) (Figure 7). These declines mark a boundary where surface-matching induction persists after training. Taken together with the trace analysis, the shifts support rule-driven over example-recitation induction while leaving analogical and pattern-based induction comparatively under-developed.

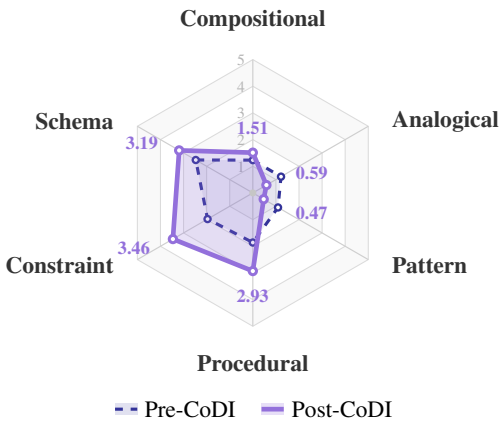

Figure 7: Radar chart of Pre- and Post-CoDI strategy scores on six induction modes (Gemini-3-flash).

Case study. Table 4 instantiates the trace and induction-mode shifts above on one BIG-Bench Navigate episode with K=3 supports and a test query unseen in the supports. FS treats the query as most similar to Ex3 and reuses its Yes label, which aligns with the example-recitation pattern in Figure 5 and with surface matching in §1. S-FS induces a strategy without seeing the query, yet its rule restates the Ex3 trajectory and yields the same wrong Yes answer, so query-blind decoupling alone does not yield a task-family rule. CoDI abstracts coordinate updates verified on all three supports and executes them to reach $(2, 2)$, which yields the correct No answer and aligns with the Procedural and Constraint gains in Figure 7. Accordingly, this case supports the hypothesis that generalization to unseen instances depends on rules verified across all supports rather than on query decoupling or surface matching alone.

5.3 What Do Induced Strategies Achieve?

Correctness and transferability lead. Analysis reveals that RL-optimized TIR excels on correctness and transferability while underperforming on executability and conciseness. We compare CoDI-Qwen3-4B against GPT-5 via pairwise judgments on four dimensions (Appendix G; Figure 8). CoDI leads on Correctness (52% vs. 46%) and Transferability (49% vs. 31%) but trails on Executability (40% vs. 57%) and Conciseness (31% vs. 39%). This asymmetry is consistent with rewards that prioritize portable rules over surface polish. The Executability and Conciseness gaps may further reflect evaluator asymmetry, as GPT-5 judges and generates with strong instruction-following while CoDI targets a fixed 4B executor (§7).

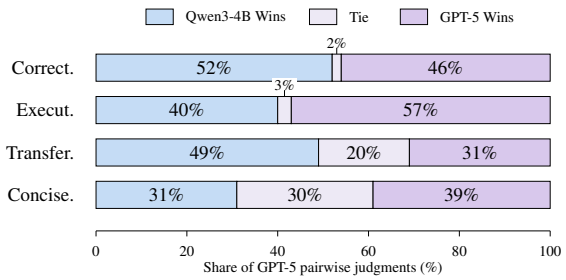

Figure 8: Pairwise GPT-5 judgments favor CoDI on Correctness and Transferability but not on Executability or Conciseness, defining the polish-oriented boundary of RL-optimized TIR.

Gains transfer to OOD execution. Analysis reveals that induced strategies raise OOD accuracy under a fixed solver, with the largest margins on rule-transfer-critical task types and limited gains where baselines already saturate. We hold Qwen3-4B AS fixed and vary only strategy source across five OOD types (Table 5). CoDI achieves the

Few-shot (K=3)**.** BIG-Bench Navigate; decide return to origin. Ex1 `N2,S2`→Yes; Ex2 `E3,W3`→Yes; Ex3 `N1,S1,E2,W2`→Yes (4 moves, returns).	**Test query.** `N4,S2,E3,W1`. Move combination unseen in supports; return to start? Gold No (final $(2,2) \neq (0,0)$).

Method	**Solving process**	**Pred.**	**Behavior**
FS	Compare query with all supports in context. Select Ex3 as the nearest match (four moves). Reuse the Yes label from Ex3 without coordinate tracking. Apply an action-count shortcut rather than origin logic, never checking whether the final position returns to $(0,0)$.	Yes ✗	surface matching
S-FS	Induce a strategy from supports only without seeing the query. Rule text states track N/S/E/W and test the origin. At solve time, pattern-matches the test query to Ex3 and copies that move sequence instead of applying the rule. Output Yes without independent coordinate calculation or endpoint verification.	Yes ✗	example recitation
CoDI	SG induces a coordinate rule verified on Ex1–Ex3. AS initializes $(0,0)$ and parses four instructions. Updates coordinates to $(0,4)$, $(0,2)$, $(3,2)$, and $(2,2)$. Tests $(2,2) \neq (0,0)$ and outputs No.	No ✓	task-level regularity

Table 4: Navigate case study on a matched BIG-Bench Navigate episode (K=3; untrained FS/S-FS, trained CoDI, Qwen3-4B). Rows compare solver processes on the same episode. ✗/✓mark incorrect and correct predictions.

largest margin on Ctx.Param.Conf. (+15.0 pp over the best closed-source baseline), where context-parameter conflicts demand explicit rule transfer. On VitaminC, it exceeds GPT-5 FS by 13.3 pp but still trails Gemini-Flash FS by 6.7 pp. On Elem.Math, Track.Shuf., and Bool.Expr., strong closed-source FS baselines remain competitive or exceed CoDI, marking ceiling-dominated settings where strategy induction adds limited value. Details appear in Appendix F.

Task Type	**Gem.-Flash** FS / S-FS	**GPT-5** FS / S-FS	**Qw3-4B** (CoDI)
Ctx.Param.Conf.	75.00 / 65.00	70.00 / 65.00	**90.00**
VitaminC	85.00 / 80.00	65.00 / 40.00	**78.33**
Elem.Math	99.00 / 89.00	95.00 / 75.00	**97.67**
Track.Shuf.	100.00 / 91.67	85.00 / 90.00	**99.44**
Bool.Expr.	90.00 / 95.00	100.00 / 85.00	**98.33**

Table 5: CoDI-induced strategies yield the largest OOD gains on rule-transfer-critical types (Ctx.Param.Conf., VitaminC) under a fixed Qwen3-4B AS; margins shrink on ceiling-dominated types.

6 Related Work

Meta-learning and in-context adaptation. MAML (Finn et al., 2017) adapts parameters via gradient updates without explicit cross-instance strategies. MetaICL (Min et al., 2022a) likewise meta-trains in-context learning at the parameter level. Mirage (Li et al., 2025) and OOD studies (Wang et al., 2024; de Wynter, 2026) report inductive failure and neighbor dependence in ICL. Motivated by these limitations, our work introduces a Cognitive-Decoupling Framework for discrete, query-blind TIR, decoupling induction from frozen execution.

Induction from examples and prompt optimization. StrategyLLM (Gao et al., 2024) and Self-Discover (Zhou et al., 2024) induce strategies from examples. Induct-Learn (Chen et al., 2024) and LEAP (Zhang et al., 2024) derive instructions from few-shot supports. All remain per-task at inference time; Induct-Learn and LEAP are not evaluated. CoT (Wei et al., 2022), STaR (Zelikman et al., 2022), DeepSeek-R1 (Guo et al., 2025), GEPA (Agrawal et al., 2026), and Prompt-MII (Xiao et al., 2025) condition execution on queries, tasks, or datasets. Our work builds upon this paradigm with Dual-Loop Meta-Inductive Learning, optimizing query-blind TIR via cross-task RL and EIR×ICR feedback.

7 Conclusion

Few-shot OOD generalization demands explicit, optimizable task-level induction beyond execution scale. To this end, we presented CoDI, combining a Cognitive-Decoupling Framework for query-blind TIR with Dual-Loop Meta-Inductive Learning for label-free cross-task inducer optimization. Experiments show that neither parameter-level adaptation nor query-blind decoupling alone yields reusable TIR, yet TIR-aligned training drives systematic OOD gains across benchmarks. Accordingly, meta-induction constitutes a trainable capability complementary to model scale. Future work may evaluate finer task-level shifts, strengthen analogical and pattern induction, and study strategy–solver co-adaptation across languages and modalities.

Limitations

Computational cost. Each training episode requires $G \times M = 32$ frozen AS forward passes, roughly $32\times$ standard SFT compute. Training completes in 12–24 hours on $8\times$A800 GPUs, but cost grows with task scale and rollout count. Our claims therefore apply at this computational regime rather than to arbitrary deployment scale.

Evaluation coverage. Evaluation covers English text-reasoning benchmarks (BIG-Bench, BBH, MATH-500, StrategyQA, ReClor); broader language, modality, or distribution shifts remain untested. GPT-5 pairwise judgments (Figure 8) show CoDI lags on Executability and Conciseness but leads on Correctness and Transferability, consistent with §5 and evaluator–executor asymmetry. Analysis further shows that analogical and pattern-based induction modes remain comparatively weak after training, and strategy substitution gains shrink on ceiling-dominated OOD types (Table 5). These bounds concern evaluation coverage, strategy polish, and induction-mode scope, not the core OOD gains from TIR-aligned training.

Strategy–solver coupling. We fix AS during training and update SG via execution feedback for stable credit assignment. TIR is natural-language and interface-decoupled, yet effective quality depends on the training-time executor. Substitution gains under a fixed executor (Table 5) support portable structure, but strategy–solver co-adaptation lies outside our current scope.

References

Lakshya A Agrawal, Shangyin Tan, Dilara Soylu, Noah Ziems, Rishi Khare, Krista Opsahl-Ong, Arnav Singhvi, Herumb Shandilya, Michael J Ryan, Meng Jiang, Christopher Potts, Koushik Sen, Alex Dimakis, Ion Stoica, Dan Klein, Matei Zaharia, and Omar Khattab. 2026. GEPA: Reflective Prompt Evolution Can Outperform Reinforcement Learning. In *International Conference on Learning Representations*.

Anonymous. 2026. Strategybench: A benchmark for evaluating strategy induction in in-context learning. Under review.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, and 12 others. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc.

Po-Chun Chen, Sheng-Lun Wei, Hen-Hsen Huang, and Hsin-Hsi Chen. 2024. Induct-learn: Short phrase prompting with instruction induction. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 5204–5231, Miami, Florida, USA. Association for Computational Linguistics.

Adrian de Wynter. 2026. Is in-context learning learning? In *The Fourteenth International Conference on Learning Representations*.

Qingxiu Dong, Lei Li, Damai Dai, Ce Zheng, Jingyuan Ma, Rui Li, Heming Xia, Jingjing Xu, Zhiyong Wu, Baobao Chang, Xu Sun, Lei Li, and Zhifang Sui. 2024. A survey on in-context learning. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 1107–1128.

Chrisantha Fernando, Dylan Banarse, Henryk Michalewski, Simon Osindero, and Tim Rocktäschel. 2024. Promptbreeder: Self-referential self-improvement via prompt evolution. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 13481–13544. PMLR.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. 2017. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, page 1126–1135. JMLR.org.

Chang Gao, Haiyun Jiang, Deng Cai, Shuming Shi, and Wai Lam. 2024. StrategyLLM: Large Language Models as Strategy Generators, Executors, Optimizers, and Evaluators for Problem Solving. In *Neural Information Processing Systems*.

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

Google DeepMind. 2025a. Gemini 3 Flash Model Card. Experiments use the Google AI preview model ID `gemini-3-flash-preview`.

Google DeepMind. 2025b. gemma-3-4b-it. Hugging Face model.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, and 542 others. 2024. The llama 3 herd of models. *Preprint*, arXiv:2407.21783.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, and 175 others. 2025. Deepseek-r1 incentivizes reasoning in llms through reinforcement learning. *Nature*, 645(8081):633–638.

John H. Holland, Keith J. Holyoak, Richard E. Nisbett, and Paul R. Thagard. 1986. *Induction: Processes of Inference, Learning, and Discovery*. MIT Press, Cambridge, MA.

Brenden M. Lake, Ruslan Salakhutdinov, and Joshua B. Tenenbaum. 2015. Human-level concept learning through probabilistic program induction. *Science*, 350(6266):1332–1338.

Jiachun Li, Pengfei Cao, Zhuoran Jin, Yubo Chen, Kang Liu, and Jun Zhao. 2025. MIRAGE: Evaluating and explaining inductive reasoning process in language models. In *International Conference on Learning Representations (ICLR)*.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2023. Let's verify step by step. *arXiv preprint arXiv:2305.20050*.

Meta. 2024. Meta-Llama-3.1-8B-Instruct. Hugging Face model.

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, Seattle, United States. Association for Computational Linguistics.

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*, pages 11048–11064, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

OpenAI. 2025. GPT-5 System Card.

Aarohi Srivastava, Abhinav Rastogi, Abhishek Rao, Abu Awal Md Shoeb, Abubakar Abid, Adam Fisch, Adam R. Brown, Adam Santoro, Aditya Gupta, Adrià Garriga-Alonso, Agnieszka Kluska, Aitor Lewkowycz, Akshat Agarwal, Alethea Power, Alex Ray, Alex Warstadt, Alexander W. Kocurek, Ali Safaya, Ali Tazarv, and 431 others. 2023. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. *Transactions on Machine Learning Research*. Featured Certification.

Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc V. Le, Ed H. Chi, Denny Zhou, and Jason Wei. 2023. Challenging BIG-Bench tasks and whether chain-of-thought can solve them. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 13003–13051.

Gemma Team, Aishwarya Kamath, Johan Ferret, Shreya Pathak, Nino Vieillard, Ramona Merhej, Sarah Perrin, Tatiana Matejovicova, Alexandre Ramé, Morgane Rivière, Louis Rouillard, Thomas Mesnard, Geoffrey Cideron, Jean bastien Grill, Sabela Ramos, Edouard Yvinec, Michelle Casbon, Etienne Pot, Ivo Penchev, and 197 others. 2025. Gemma 3 technical report. *Preprint*, arXiv:2503.19786.

Qixun Wang, Yifei Wang, Yisen Wang, and Xianghua Ying. 2024. Can in-context learning really generalize to out-of-distribution tasks? *Preprint*, arXiv:2410.09695.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, brian ichter, Fei Xia, Ed Chi, Quoc V Le, and Denny Zhou. 2022. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*, volume 35, pages 24824–24837. Curran Associates, Inc.

Emily Xiao, Yixiao Zeng, Ada Chen, Chin-Jou Li, Amanda Bertsch, and Graham Neubig. 2025. Prompt-mii: Meta-learning instruction induction for llms. *Preprint*, arXiv:2510.16932.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, and 41 others. 2025. Qwen3 technical report. *Preprint*, arXiv:2505.09388.

Weihao Yu, Zihang Jiang, Yanfei Dong, and Jiashi Feng. 2020. ReClor: A reading comprehension dataset requiring logical reasoning. In *International Conference on Learning Representations*.

Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah Goodman. 2022. Star: Bootstrapping reasoning with reasoning. In *Advances in Neural Information Processing Systems*, volume 35, pages 15476–15488. Curran Associates, Inc.

Tianjun Zhang, Aman Madaan, Luyu Gao, Steven Zheng, Swaroop Mishra, Yiming Yang, Niket Tandon, and Uri Alon. 2024. In-context principle learning from mistakes. *Preprint*, arXiv:2402.05403.

Pei Zhou, Jay Pujara, Xiang Ren, Xinyun Chen, Heng-Tze Cheng, Quoc V. Le, Ed H. Chi, Denny Zhou, Swaroop Mishra, and Huaixiu Steven Zheng. 2024. Self-discover: Large language models self-compose reasoning structures.

Yongchao Zhou, Andrei Ioan Muresanu, Ziwen Han, Keiran Paster, Silviu Pitis, Harris Chan, and Jimmy Ba. 2023. Large language models are human-level prompt engineers. *Preprint*, arXiv:2211.01910.

A Training Algorithm and Implementation Details

All datasets and pretrained models used in this work are publicly released artifacts; their use and redistribution follow the licenses and terms specified in the original publications cited in §4.1 and Appendix I. CoDI training code is linked in the abstract; StrategyBench construction scripts and task-level splits are available for double-blind review in Appendix I. Both will be released publicly under the MIT License upon acceptance.

A.1 Models, Evaluation Protocols, and Baselines

Models and APIs. We evaluate CoDI on **Qwen3-4B/8B** (Yang et al., 2025), **Llama-3.1-8B** (Grattafiori et al., 2024), and **Gemma-3-4B** (Team et al., 2025). Closed-source APIs (**Gemini-3-flash** (Google DeepMind, 2025a), **GPT-5** (OpenAI, 2025)) serve as inference-time upper bounds only and are not fine-tuned. Throughout, **Gemini-3-flash** denotes Google AI model ID `gemini-3-flash-preview`; this model is also used for data filtering, SFT demonstration generation, and annotation or judging in the main analysis. Table names are shorthand: **Gemma-3-4B** denotes `gemma-3-4b-it` (Google DeepMind, 2025b) and **Llama-3.1-8B** denotes `Llama-3.1-8B-Instruct` (Meta, 2024).

Baselines and implementation. MetaICL and GEPA are described in §4.1. GEPA uses `gemini-3-flash-preview` as the optimizer and Qwen3-4B as the executor, with four reflective iterations. Defaults are G=8, K=3, and β=0.5 on 8×A800 GPUs; full hyperparameters are in Table 6 and the training pipeline below.

Numeric format. Reported accuracies are percentages (e.g., 45.00 denotes 45.00%); $\Delta\%$ is the percentage-point change relative to the same-backbone FS baseline (§4.1).

A.2 Hyperparameters

Two-stage pipeline. CoDI training comprises Phase 1 (SFT warm-up) and Phase 2 (inductive alignment); hyperparameters are in Table 6.

Phase 1: SFT warm-up. π_{SG} is fine-tuned on teacher-constructed demonstrations to generate format-compliant strategies, supervising schema validity without task-accuracy objectives. After a few hundred steps (>90% format-valid rate), $\pi_{\text{SG}}^{(0)}$ initializes per-task EIR baselines $\{b_\tau\}$ from inference accuracies on training tasks.

Hyperparameter	Value
Model configuration	
Strategy Generator (SG)	Qwen3-4B (trainable)
Answer Solver (AS)	Qwen3-4B (frozen, non-thinking mode)
Hardware	8×A800 (80 GB)
Training time	∼12–24 h (one full CoDI run)
Training hyperparameters	
Rollout count G	8
Few-shot shots K	3
Learning rate	1×10^{-6} (AdamW)
Max generation length	16,384 tokens
Training data	∼20K samples
Reward parameters	
Asymm. weight α^+ (positive gain)	0.7
Asymm. weight α^- (negative gain)	0.3
ICR amplification β	0.5
Log-smoothing constant k	10
Lower-bound cutoff ε	0.05
Baseline EMA decay γ	0.9

Table 6: Hyperparameters for CoDI training.

Phase 2: Inductive alignment. We optimize π_{SG} with the joint reward in Eq. (7) via cross-task GRPO while AS remains frozen. Format-invalid rollouts are excluded from AS execution and receive a reward of zero (format gating below). Each episode samples one task type uniformly at random.

Format gating. A binary format check validates whether all required schema sections are present and structurally valid. Format-invalid strategies are assigned a total reward of $R_i = 0$ and are not passed to AS for execution; EIR and ICR are not computed for them. Setting the reward to zero (rather than a negative penalty) avoids artificially inflating the variance of the reward signal while still withholding any positive reinforcement for malformed outputs, allowing GRPO to steer the policy away from format violations without destabilising training.

A.3 EIR: Difficulty-Adaptive Normalizer

The normalizer $c(b_\tau)$ in Eq. (4) is defined as

$$c(b_\tau) = \max\big(b_\tau^{\alpha}(1-b_\tau)^{1-\alpha},\ \varepsilon\big) \qquad (8)$$

where α is selected by the sign of δ_i and $\varepsilon = 0.05$ lower-bounds the denominator when b_τ approaches 0 or 1.

Mathematical properties. The function $b^{\alpha}(1-b)^{1-\alpha}$ on $b \in [0,1]$ is an unnormalized $\text{Beta}(\alpha{+}1, 2{-}\alpha)$ density, maximized uniquely at $b = \alpha$. When $b_\tau = \alpha$, $c(b_\tau)$ is largest and $|g_{\text{raw}}|$ is smallest; as b_τ moves toward easy ($b_\tau \to 1$)

or hard ($b_\tau \to 0$) extremes, $c(b_\tau)$ shrinks and $|g_{\text{raw}}|$ grows, up-weighting deviations on extreme-difficulty tasks.

Role of asymmetric α. With $\alpha^+ = 0.7 > \alpha^- = 0.3$, EIR applies two difficulty calibrations:

- **Positive ($\delta_i \geq 0$, $\alpha = \alpha^+$):** $c(b_\tau)$ peaks at $b_\tau = 0.7$, giving mid-to-high difficulty tasks a natural baseline while amplifying gains on very hard tasks ($b_\tau \ll 0.7$).
- **Negative ($\delta_i < 0$, $\alpha = \alpha^-$):** $c(b_\tau)$ peaks at $b_\tau = 0.3$; regressions on high-baseline tasks ($b_\tau \gg 0.3$) receive stronger penalties, whereas low-baseline regressions retain larger $c(b_\tau)$ and preserve early exploration.

Ablation. Setting $\alpha^+ = \alpha^- = 0.5$ causes near-complete training collapse (BBH 4.26, OOD 3.67; Table 9), confirming that asymmetric EIR weighting is necessary for stable cross-task GRPO.

B Full Strategy Schema and Format Comparison

This section (i) compares three representative strategy formats on prompts and typical SG outputs (Table 3), (ii) specifies the CoDI semi-structured template, and (iii) provides two cross-task examples.

B.1 Format comparison, specification, and examples

Using BIG-Bench Navigate (grid tracking; decide whether the agent returns to the origin), we compare three formats along a structure axis: Free-form → CoT-style → Semi-structured (CoDI, recommended), matching Table 3. The table also ablates individual mandatory sections (`[INDUCTIVE_RULE]`, `[TASK_PROCEDURE]`, `[TRANSFER_SCOPE]`).

Free-form (unstructured). Prompt:

```
Based on the following examples, generate a strategy to solve this type of problem.

[few-shot examples]

Strategy:
```

Typical SG output:

```
Track coordinates. North +y, South -y, East +x, West -x. Check if (x,y) = (0,0) at end.
```

Limitation: Output is unconstrained in length and structure, so token-level log-probability baselines are incomparable across rollouts and ICR is ill-defined. SG often paraphrases specific examples rather than stating verifiable task-family rules, yielding the weakest OOD generalization.

CoT-style (implicit structure). Prompt:

```
Based on the following examples, analyze the pattern and generate a step-by-step strategy for solving this type of problem. Think carefully and reason step by step.

[few-shot examples]

Strategy:
```

Typical SG output:

```
Let me analyze the examples: - Example 1: coordinates update with each direction... - Example 2: same pattern confirmed...

General approach: 1. Initialize position (0,0) 2. Parse each instruction... 3. Check if final position is origin

This should work for similar grid navigation problems, though diagonal movements might need special handling.
```

Limitation: Reasoning steps appear, but rules and procedures are interleaved and boundary conditions are ad hoc. Without fixed section headers, AS sees variable prompts, ICR is only moderately reliable, and OOD transfer is weaker than under semi-structured control.

Semi-structured / CoDI (recommended). Prompt:

```
Based on the following examples, generate a strategy using EXACTLY the following sections: [INDUCTIVE_RULE] / [TASK_PROCEDURE] / [TRANSFER_SCOPE]

[few-shot examples]

Strategy:
```

Typical SG output (expanded examples below):

```
[INDUCTIVE_RULE] Update coordinates: North (+y), South (-y), East (+x), West (-x). Decide whether the endpoint equals the origin (0,0).

[TASK_PROCEDURE] 1. Initialize (x,y)=(0,0) 2. Parse each instruction for direction and steps 3. Update coordinates 4. Output Yes if (x,y)==(0,0), else No

[TRANSFER_SCOPE] Applies when directional moves are given and the query asks about returning to the start. Not applicable when diagonal moves or conditional branches appear; extend Step 2 accordingly.
```

Advantage: Fixed section headers yield comparable token-level baselines and measurable ICR; INDUCTIVE_RULE enforces explicit induction and TRANSFER_SCOPE manages boundaries, giving the strongest OOD performance.

Semi-structured specification. CoDI requires SG to emit three named sections; we define their semantics and format violations below.

Mandatory sections. Each strategy must contain three sections delimited by header lines ([SECTION_NAME]):

- **[INDUCTIVE_RULE]**: Task-family regularities inferred from K examples, verifiable against supports but not tied to any single instance. Boundary cases belong in [TRANSFER_SCOPE].
- **[TASK_PROCEDURE]**: Three to six numbered, executable steps covering parsing, reasoning, and output formatting.
- **[TRANSFER_SCOPE]**: Scope specification with trigger conditions, at least one non-applicable case, and any exceptions to the inductive rule.

Format violations—missing sections, misspelled headers, or empty section bodies—assign $R_i = 0$ and exclude the strategy from AS execution (format gating above).

Worked example (Navigate). Full Navigate strategy illustrating the three-section template:

```
[INDUCTIVE_RULE]

From the examples: each instruction
has the form "walk {steps} steps
{direction}." Update North (+y), South
(-y), East (+x), West (-x); test whether
the final point is (0,0).

[TASK_PROCEDURE]

1. Initialize (x, y) = (0, 0). 2.
Parse each instruction for direction
and step count. 3. Update coordinates
accordingly. 4. After all instructions,
check whether (x, y) = (0, 0). 5. Output
"Yes" if true, otherwise "No."

[TRANSFER_SCOPE]

Trigger: a sequence of directional
moves with a return-to-start
query. Application: apply the
coordinate-tracking steps directly.
Non-applicable: (1) diagonal moves; (2)
conditional branches (e.g., "if ...,
then go north"), which require extra
parsing in Step 2.
```

Worked example (Word sorting). Word Sorting example showing cross-domain use:

```
[INDUCTIVE_RULE]

Sort the given word list alphabetically
(case-insensitive). Compare characters
left to right; digits precede letters;
special characters follow ASCII order.

[TASK_PROCEDURE]

1. Lowercase all words for comparison.
2. Sort by alphabetical order with
character-wise comparison. 3. If
prefixes match, place the shorter word
first. 4. Output the sorted list as
comma-separated tokens.

[TRANSFER_SCOPE]

Trigger: unordered word list requiring
alphabetical output. Application:
follow Steps 1-4. Non-applicable: (1)
frequency- or semantics-based sorting
requires a new criterion in Step 2; (2)
multi-word phrases must be treated as
single tokens in Step 1.
```

C Six Induction Modes Scoring Prompt

Figure 9 gives the Gemini-3-flash prompt for scoring strategies on six induction modes (0–5 each). The taxonomy follows classic induction theory (Holland et al., 1986) and maps onto CoDI sections: Compositional and Procedural target [TASK_PROCEDURE]; Analogical and Pattern Extrapolation target [INDUCTIVE_RULE]; Constraint-based and Schema target [TRANSFER_SCOPE].

D Thinking-Unit Taxonomy and Trace Metrics

This section supports §5.2 and Figure 5. Traces are collected under the same few-shot OOD protocol as the main evaluation, with thinking mode enabled so that the model emits <think> spans before answering. We compare Pre-CoDI and Post-CoDI checkpoints on matched OOD instances, split each <think> span into sentence-level units, and assign E/D/P/R labels with Gemini-3-flash (Figure 10).

D.1 Unit taxonomy (E/D/P/R)

Phase-1 prefix. Let the labeled unit sequence be $(y_1, \ldots, y_n)$. **Phase-1** is the maximal prefix of consecutive E labels at the start of the trace: indices $\{1, \ldots, t\}$ where $y_1 = \cdots = y_t = E$ and either $t = n$ or $y_{t+1} \neq E$. We report $n_E^{(\mathrm{ph1})} = \sum_{i \leq t} \mathbf{1}[y_i{=}E]$, which equals t on this definition.

Tag	Name	Ind.?	Description (annotation target)
E	Example-recitation	No	The unit recalls, quotes, or closely paraphrases a *specific* support example without abstracting beyond it.
D	Direct-application	No	The unit applies facts or procedures to the current instance without first stating a cross-example pattern.
P	Pattern-abstraction	Yes	The unit states a general rule or invariant intended to hold across instances (inductive content).
R	Rule-application	Yes	The unit applies a rule or pattern that was already stated earlier in the trace.

Table 7: Cognitive tags for sentence-level units inside `<think>`. "Ind.?" indicates whether the unit carries inductive (rule-level) content.

D.2 Trace metrics and statistical tests

Let $(y_1, \ldots, y_n)$ be labels after dropping empty segments; n is the number of units. Denote counts $n_c = \sum_i \mathbf{1}[y_i{=}c]$ for $c \in \{E, D, P, R\}$, and $n_{P\cup R} = n_P + n_R$. **Trajectory EIR (unit ratio).** Whenever $n_{P\cup R} > 0$, define $\mathrm{EIR}_{\mathrm{unit}} = n_E/n_{P\cup R}$ (distinct from the GRPO reward R_{EIR} in Eq. (5)). **Example share.** $\rho_E = n_E/n$. **Think length.** L_{think} is the character length of the full `<think>` span (including whitespace). **D→P rate.** $r_{D\to P} = \frac{1}{n-1}\sum_{i=1}^{n-1} \mathbf{1}[y_i{=}D,\ y_{i+1}{=}P]$ (reported as a percentage). **Mean P length.** $\bar{\ell}_P = \mathrm{mean}\{\mathrm{len}(\mathrm{unit}_i) \mid y_i{=}P\}$ (characters); undefined if $n_P{=}0$ (samples excluded from $\bar{\ell}_P$ averages only). **Accuracy.** $\mathrm{Acc}_{\mathrm{OOD}}$ is exact-match accuracy on the same OOD few-shot evaluation set as Fig. 5.

Inference and testing. All sentence labels are produced by Gemini-3-flash with temperature 0; inter-run agreement will be summarised with Cohen's $\kappa{=}---$ upon annotation freeze. Unless otherwise noted, p-values are two-sided paired tests at the trajectory level over matched tasks (Pre vs. Post checkpoints); we report them for the primary continuously scored process metrics below.

E Sensitivity Analysis

Table 9 reports hyperparameter sensitivity for K (shot count), β, α, and k (log-smoothing constant in EIR) on Qwen3-4B; * marks the default configuration used in all main-text experiments.

Shot count K. Performance peaks at $K{=}3$ (BBH 82.22, OOD 63.00) and drops slightly at $K{=}5$ (BBH 78.52, OOD 57.67). The largest single-step gain is $K{=}1{\to}2$ (+24.32 BBH), reflecting enriched support-set contrast for the `INDUCTIVE_RULE` section; gains slow beyond $K{=}3$. Even at $K{=}1$ (BBH 44.20, OOD 34.71), the model retains non-trivial induction ability, confirming that meta-inductive priors are partially encoded as parametric knowledge.

Metric	Pre	Post	Δ	p
L_{think} (chars)	5276	4171	-1105	$< 10^{-3}$
$\mathrm{EIR}_{\mathrm{unit}}$	1.079	0.674	-0.405	$3.0{\times}10^{-4}$
$r_{D\to P}$ (%)	4.36	6.21	+1.85	$4.0{\times}10^{-5}$
$\bar{\ell}_P$ (chars)	89.4	108.6	+19.2	$8{\times}10^{-6}$
$\mathrm{Acc}_{\mathrm{OOD}}$ (exact)	0.452	0.924	+0.472	—
$n_E^{(\mathrm{ph1})}$ (count)	18.54	9.06	-9.48	—
ρ_E (ratio)	16.50	12.14	-4.36	—

Table 8: Pre/Post-CoDI trace statistics (Qwen3-4B, matched OOD tasks, $K{=}3$). p: paired tests for the four continuously annotated process metrics; "—" indicates statistics reported at the aggregate level without the same paired p-value pipeline in this draft.

Parameter	Value	ID	BBH	OOD
K (shots)	1	48.89	44.20	34.71
	2	55.67	68.52	48.00
	$3^\star$	64.67	82.22	63.00
	5	63.67	78.52	57.67
β (ICR weight)	0.3	50.00	44.20	44.67
	$0.5^\star$	64.67	82.22	63.00
	0.7	39.00	48.89	47.00
α (EIR asym.)	asym. 0.1/0.9	55.67	71.85	47.33
	asym. $0.3/0.7^\star$	64.67	82.22	63.00
	sym. 0.5/0.5	4.00	4.26	3.67
k (log-smooth.)	1	30.00	35.00	34.00
	5	37.33	41.85	36.67
	$10^\star$	64.67	82.22	63.00
	20	41.67	49.81	36.00

Table 9: Hyperparameter sensitivity on Qwen3-4B for K, β, α, and k (*: default; exact-match accuracy, %).

ICR amplification β. $\beta{=}0.5$ is optimal across all metrics. Performance degrades substantially at both $\beta{=}0.3$ (BBH 44.20, OOD 44.67) and $\beta{=}0.7$ (BBH 48.89, OOD 47.00), indicating a non-monotone sensitivity: too low an amplification under-weights the inductive signal, while too high distorts the reward landscape and destabilises GRPO.

EIR asymmetry α. Asymmetric weighting $(\alpha^+, \alpha^-) = (0.7, 0.3)$ achieves the best result (OOD 63.00). The symmetric setting $(0.5, 0.5)$ causes near-complete training collapse (BBH 4.26, OOD 3.67), confirming that the asymmetric de-

sign is critical for GRPO stability across diverse task difficulties. A stronger penalty bias $(0.1, 0.9)$ intermediate-degrades performance (OOD 47.33), suggesting that over-penalising negative outcomes suppresses exploration on hard tasks.

Log-smoothing constant k. k controls the degree of logarithmic compression applied to the difficulty-normalized gain g_{raw} in EIR (distinct from shot count K). The default k=10 achieves the best performance across all splits (BBH 82.22, OOD 63.00). At k=1, insufficient smoothing leads to noisy reward signals that severely destabilise training (BBH 35.00, OOD 34.00); performance rises substantially as k increases to 5 (BBH 41.85, OOD 36.67) and peaks at k=10. Beyond the optimum, increasing k to 20 over-compresses difficulty gradients and degrades performance (BBH 49.81, OOD 36.00), revealing a clear non-monotone sensitivity: the logarithmic compression must be neither too weak nor too strong, with k=10 providing the best balance.

F Solver Transferability

Table 10 fixes the CoDI-4B SG and substitutes different AS models to examine how well the induced strategies transfer across executors. 4B, 8B, and 32B AS all pair well with CoDI-4B strategies (OOD: 61.93/62.89/61.30), while the 14B AS is notably lower (51.14)—an anomaly suggesting that transfer sensitivity is governed by instruction-tuning quality rather than raw model scale. Systematic solver compatibility analysis is left for future work.

SG	AS	**OOD↑**	**Δ vs.** CoDI-4B AS
CoDI-4B	4B	61.93	−1.07
	8B	62.89	−0.11
	14B	51.14	−11.86
	32B	61.30	−1.70

Table 10: Solver transfer with fixed CoDI-4B SG; deltas are absolute OOD changes relative to CoDI-4B AS (63.00).

G Strategy Quality Evaluation Prompt

Figure 11 gives the GPT-5 prompt used to compare strategies generated by the CoDI-trained Qwen3-4B and GPT-5. The model identities are provided as Model A and Model B in the prompt, and GPT-5 returns dimension-level preferences and reasons in JSON format.

H LOO Rule Sensitivity and Inductive-Rule LLM Judging

Figure 4 reports embedding-based leave-one-out (LOO) sensitivity of the `[INDUCTIVE_RULE]` span on matched evaluation tasks with K=3. GPT-5 LLM-as-judge scores on four rubric dimensions are collected with the prompt in Fig. 12.

Row	**Task type**
T01	Movie recommendation
T02	Dyck languages
T03	Penguins in a table
T04	Geometric shapes
T05	Formal fallacies
T06	Temporal sequences
T07	Nonsense-word grammar
T08	Logical deduction
T09	Salient translation error detection
T10	Date understanding

Table 11: Task index for the rows in Fig. 4.

Stratified sampling and rule generation. We stratify by `problem_type`, draw evaluation shards, and use vLLM to sample M=5 independent `[INDUCTIVE_RULE]` texts per task under full few-shot conditioning.

LOO sensitivity and heatmap cells. Let $\mathbf{e}_i$ be the bge-m3 embedding of rule i, and $\mathbf{e}_i^{\{-j\}}$ the embedding after re-generating the rule with support example j removed. Per-rule sensitivity is $s_{ij} = 1 - \cos(\mathbf{e}_i, \mathbf{e}_i^{\{-j\}})$. Figure 4 plots task-level values $\bar{s}_{\tau j} = \frac{1}{M}\sum_{i=1}^{M} s_{ij}$ for ten representative tasks τ and slots $j \in \{1, 2, 3\}$. Cell colors are clipped at $v_{\max}$=0.5 for readability. The headline means $\bar{s}_{\text{LOO}}$ are the averages of these 30 cells (Pre: 0.325; Post: 0.087).

Optional softmax attribution. Softmax weights $a_{ij} = \text{softmax}(s_{i1}, s_{i2}, s_{i3})_j$ are useful for per-rule attribution summaries; the heatmap cells in Fig. 4 use the raw cosine-based sensitivities averaged over rules as above.

LLM-as-judge aggregates. GPT-5 assigns integer scores in $\{1, \ldots, 5\}$ with a one-sentence justification per dimension. We report mean `coverage_score` as *Coverage*, and mean $(6 - \texttt{copying_score})$ as *Originality* (higher `copying_score` is worse).

I StrategyBench Dataset

StrategyBench (Anonymous, 2026) is a benchmark for evaluating *strategy induction* and *strategy*

application in in-context learning. It is constructed from BIG-Bench (Srivastava et al., 2023) through task filtering, format standardization, task categorization, and split construction, targeting tasks where a model must infer reusable solution strategies from few-shot demonstrations and apply them to unseen test queries.

Data format. Each example follows a unified input–output structure: the **input** field contains the task instruction and query instance; the **output** field contains the target answer used for automatic evaluation. For strategy-based experiments, each test episode additionally includes a small set of few-shot demonstrations used either to induce a strategy or to provide in-context examples for direct inference. Full construction details are given in Anonymous 2026.

I.1 Five-Criterion Task Filtering

To ensure that training data are suitable for strategy induction and downstream reinforcement learning, original BIG-Bench tasks are filtered and processed along five dimensions:

1. **Objective evaluability.** Tasks should have relatively objective and automatically checkable answers; tasks with highly subjective judgments, open-ended opinions, ethical preferences, or unstable answer boundaries are excluded from the main training and evaluation sets.
2. **Format stability.** Tasks should have stable input–output formats after standardization, so that few-shot demonstrations, generated strategies, and model predictions can be organized into a consistent episode format.
3. **Rule inductibility.** Tasks should contain patterns, rules, mappings, or procedures inferable from a small number of demonstrations; tasks that mainly depend on memorized external facts or arbitrary labels are less suitable for strategy induction.
4. **Low external-knowledge dependency.** Tasks should be solvable mainly from the provided examples and query context rather than requiring substantial external knowledge, reducing the risk that performance reflects memorization or world knowledge rather than strategy induction.
5. **Transfer potential.** Tasks should allow the induced strategy to be applied to new instances within the same task family or under distribution shifts, supporting evaluation of whether a strategy is reusable beyond the exact demonstrations.

Unsuitable tasks are removed or placed into a special set; retained tasks are standardized and used to construct training, validation, and test episodes. As BIG-Bench-derived synthetic reasoning tasks, our data do not contain personally identifying information; tasks with subjective or potentially sensitive content were excluded during filtering.

Teacher scoring. Retained tasks are further processed with a teacher model (**Gemini-3-flash**; §A.1) for data filtering, SFT demonstration generation, and quality scoring, providing the supervised initialization used in Phase 1 (§4.1).

Code and splits. For double-blind review, `https://anonymous.4open.science/r/StrategyBench-D53C/` hosts the five-criterion filtering pipeline, standardized episode format, and ID/OOD split definitions used in §4.1. Reviewers can reproduce dataset construction and splits from that repository.

J Task-Level ID/OOD Partition

J.1 In-Distribution (ID) Split

The ID split comprises tasks or subtasks present in the training data, with test *instances* held out from training instances. It evaluates whether a model can induce and apply strategies when task rules, input–output formats, and demonstration patterns resemble those seen during training.

J.2 Out-of-Distribution (OOD) Split

The OOD split is created by completely holding out certain tasks, subtasks, or task categories from training and using them only for evaluation. OOD examples differ more strongly from training data in task source, input–output format, reasoning structure, or required capabilities.

Granularity levels. StrategyBench defines finer OOD levels—*OOD-subtask* (unseen subtasks under related sources), *OOD-task* (unseen tasks with related ability requirements), *OOD-category* (unseen coarse categories with larger distribution shifts), and *BBH* (selected Big-Bench Hard tasks)—for diagnostic analysis. In this paper we report aggregate **ID** and **OOD** splits at task level unless otherwise noted.

Prompt Used for Six-Mode Induction Scoring

You are an expert in inductive reasoning. Score the following model-generated strategy on six theoretically grounded modes of induction. Each mode is rated on an integer scale 0–5; scores must be justified by direct reference to the strategy text.

Task description: {task}

Strategy to score: {strategy text}

Six induction modes:

1. **Compositional** — Does the strategy combine multiple atomic sub-rules into a composite procedure (e.g. chaining conditions, nesting operations)?
2. **Analogical** — Does the strategy draw structural correspondences across examples, mapping shared relational patterns rather than surface features?
3. **Pattern Extrapolation** — Does the strategy extract a pattern that visibly extends beyond the demonstrated instances (e.g. to longer sequences, new value ranges, unseen entities)?
4. **Procedural** — Does the strategy abstract a step-wise algorithm from input–output pairs (clear ordering, state updates, termination conditions)?
5. **Constraint-based** — Does the strategy identify and articulate implicit constraints, preconditions, or boundary conditions (e.g. when the rule does *not* apply)?
6. **Schema-induction** — Does the strategy explicitly classify the task type itself, recognising the meta-level structure (e.g. "this is a coordinate-tracking problem")?

0–5 scoring rubric (applied per dimension):

- **0** — Absent: no evidence of this mode in the strategy.
- **1** — Hinted: a fragment alludes to it but is not developed.
- **2** — Partial: present but incomplete or ambiguous.
- **3** — Adequate: clearly present, executable, but minimal.
- **4** — Strong: well-developed, multi-step or multi-condition.
- **5** — Exemplary: cleanly abstracted, generalisable, with explicit scope statements.

Calibration constraints: (i) Scores must reference specific text spans (quote the supporting sentence in `evidence`). (ii) If a mode is structurally inapplicable to the task type, mark as `N/A` rather than 0. (iii) Modes are scored independently; high score on one does not imply high score on others.

Output strictly in JSON:

```
{
  "compositional": {"score": 0-5 or "N/A", "evidence": "...", "reason": "..."},
  "analogical": {"score": 0-5 or "N/A", "evidence": "...", "reason": "..."},
  "pattern_extrapolation": {"score": 0-5 or "N/A", "evidence": "...", "reason":
"..."},
  "procedural": {"score": 0-5 or "N/A", "evidence": "...", "reason": "..."},
  "constraint_based": {"score": 0-5 or "N/A", "evidence": "...", "reason": "..."},
  "schema_induction": {"score": 0-5 or "N/A", "evidence": "...", "reason": "..."},
  "composite": {"mean_score": float, "applicable_modes": int}
}
```

Figure 9: Prompt used for six-mode induction scoring. Each strategy receives an independent 0–5 score per mode; the composite is the mean over applicable modes (N/A modes excluded from averaging).

Prompt Used for Thinking-Unit Classification (E/D/P/R)

System. You are an expert cognitive analyst studying how language models reason during few-shot problem solving. Classify a *single* reasoning sentence into exactly one of four categories by its cognitive function. Be strict and consistent; output valid JSON only, with no extra text.

User.

Task context. Problem: {problem_text}

Support examples (summary): {examples_summary}

Four categories.

1. **E (Example-recitation):** The sentence directly recalls, quotes, or closely paraphrases content from a specific support example. No abstraction beyond that example.
2. **D (Direct-application):** The sentence applies knowledge, a fact, or a procedure to the current problem instance without first deriving or stating a general pattern (calculations, lookups, direct inferences from given facts).
3. **P (Pattern-abstraction):** The sentence states a general principle, rule, or pattern intended to hold across multiple examples or instances.
4. **R (Rule-application):** The sentence explicitly applies a general rule or pattern that was stated or inferred earlier in the reasoning trace.

Sentence to classify: {unit_text}

Output strictly as JSON (one line, no extra keys): `{"label": "<E|D|P|R>", "reason": "<one sentence>"}`

Figure 10: Prompt for Gemini-3-flash sentence-level tagging of `<think>` units.

Prompt Used for Strategy Quality Analysis

Please act as a professional strategy evaluator and compare the following two model-generated strategies across four dimensions.

Task description: {task}

Model A Strategy: {Model-A-generated strategy}

Model B Strategy: {Model-B-generated strategy}

Compare the two strategies on the following four dimensions:

1. **Correctness** — Does the strategy accurately capture the task rules and produce correct solutions?
2. **Executability** — Is the strategy clear and actionable enough for a model to follow step by step?
3. **Transferability** — Is the strategy abstract and generalizable enough to apply to unseen tasks of the same type?
4. **Conciseness** — Is the strategy expressed concisely without redundant or repetitive content?

For each dimension, indicate which is better and provide a brief reason. Output in JSON format:

```
{
  "correctness_better": "Model A" / "Model B" / "tie",
  "correctness_reason": "reason",
  "executability_better": "Model A" / "Model B" / "tie",
  "executability_reason": "reason",
  "transferability_better": "Model A" / "Model B" / "tie",
  "transferability_reason": "reason",
  "conciseness_better": "Model A" / "Model B" / "tie",
  "conciseness_reason": "reason",
  "overall_better": "Model A" / "Model B" / "tie",
  "overall_reason": "reason"
}
```

Figure 11: Prompt used for strategy quality analysis.

Prompt Used for Inductive-Rule Quality (JSON-only)

System. You are an expert evaluator for few-shot inductive-rule quality. You will receive (i) a task description, (ii) K=3 support input–output pairs, and (iii) a generated inductive rule `[INDUCTIVE_RULE]`. Score the rule objectively on four dimensions (1–5) with a one-sentence justification each. Output **only** valid JSON (no markdown, no extra text).

Calibration (do not score). *Example A* (high abstraction / low copying): sentiment task; rule "Identify whether the review expresses a net positive or negative evaluation of a product attribute; classify accordingly." Expected: abstraction 5, coverage 5, copying 1, transferability 5. *Example B* (low abstraction / high copying): same demos; rule "If the review mentions battery life dying quickly, classify it as Negative." Expected: abstraction 1, coverage 1, copying 5, transferability 1.

Task. {task_description}

Support. Ex1: in {ex1_input}, out {ex1_output}; Ex2: in {ex2_input}, out {ex2_output}; Ex3: in {ex3_input}, out {ex3_output}.

Rule. {inductive_rule}

Rubric (1–5 each). (1) `abstraction_score`: task-level principle vs. paraphrasing demos (5 fully abstract; 1 restates one demo). (2) `coverage_score`: supported by multiple demos (5 all three; 1 only one). (3) `copying_score`: copied from a single demo (5 strong copying; 1 none)—**higher is worse.** (4) `transferability_score`: useful on unseen instances of the same task type.

JSON schema.
```
{"abstraction_score": int (1-5), "abstraction_reason": "one sentence",
 "coverage_score": int (1-5), "coverage_reason": "one sentence",
 "copying_score": int (1-5), "copying_reason": "one sentence",
 "transferability_score": int (1-5), "transferability_reason": "one sentence"}
```

Figure 12: GPT-5 prompt for scoring each `[INDUCTIVE_RULE]` on abstraction, coverage, copying (reverse-scored), and transferability.