PROMPTING FOR ARC: INSTRUCTION DESIGN FOR GRID REASONING

Zhuoka Feng
Fudan University
23307130211@m.fudan.edu.cn

ABSTRACT

The Abstraction and Reasoning Corpus (ARC) evaluates a system's ability to infer novel grid transformation rules from only a few examples. Despite strong performance on many NLP benchmarks, large language models (LLMs) still struggle with ARC's discrete, compositional reasoning. In this work, we present a systematic study of prompt engineering strategies for ARC. We design five base prompting strategies targeting different failure modes: Minimal Prompting, Reasoned Prompting, Schema-Constrained Prompting, Reflective Prompting, and Structured Few-shot Prompting. Building upon these strategies, we propose an Adaptive Selection method that dynamically chooses the most suitable strategy based on task characteristics, combined with multi-agent collaboration for iterative refinement. Experiments on val.jsonl show that our method achieves 66.7% accuracy, outperforming all base strategies. On the more challenging val_hard.jsonl, our method achieves approximately four-fold improvement over the baseline. Through error analysis and case studies, we identify that rule error is the dominant failure mode and reveal the complementary strengths of different prompting strategies for abstract reasoning tasks. Our code is available at https://arkazhuo.github.io/ARC/.

1 INTRODUCTION

Large language models (LLMs) have demonstrated remarkable capabilities across a wide spectrum of natural language processing tasks, from text generation and summarization to complex multi-step reasoning (Brown et al., 2020; OpenAI, 2023). A key enabler of this versatility is in-context learning (ICL), where models adapt to new tasks by conditioning on a small set of demonstrations without any parameter updates (Brown et al., 2020; Ouyang et al., 2022). This paradigm has given rise to prompt engineering as a systematic discipline (Liu et al., 2023; DAIR.AI, 2024), encompassing instruction design, example organization, reasoning scaffolds such as chain-of-thought prompting (Wei et al., 2022; Kojima et al., 2022), and self-verification mechanisms (Weng et al., 2023; Madaan et al., 2023).

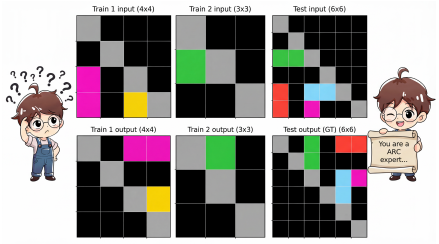

Figure 1: An ARC-style grid transformation example.

Despite these advances, LLMs continue to struggle with tasks that require precise, compositional reasoning over discrete structures. The Abstraction and Reasoning Corpus (ARC) (Chollet, 2019b) exemplifies this challenge. Designed as a benchmark for measuring human-like general fluid intelligence, ARC presents tasks where a system must infer an unknown transformation rule from only 2–3 training input-output grid pairs and apply it to a held-out test input, as shown in Figure 1. Each grid is a small 2D array of integers in the range 0–9, each representing a color, and the transformation rules span diverse visual and logical operations. Unlike typical NLP benchmarks, ARC demands exact match accuracy where every cell must match the ground truth, exposing several fundamental difficulties for LLMs: grids are inherently visual and spatial yet must be processed as linearized text; outputs may be syntactically malformed or structurally invalid even when the rule is correctly

inferred; multiple plausible rules may fit the limited training examples; and applying the correct rule without counting or coordinate errors remains challenging.

Given these challenges, we ask: To what extent can prompt engineering alone improve LLM performance on ARC? What types of prompting strategies are most effective, and what failure modes do they address? Rather than proposing novel architectures or fine-tuning approaches, we focus on the underexplored question of how instruction design, output constraints, reasoning scaffolds, and inference-time aggregation interact in this demanding setting.

In this paper, we develop a stable ARC inference pipeline and systematically compare six prompting strategies, including Minimal Prompting as a baseline, Reasoned Prompting with chain-of-thought scaffolds, Schema-Constrained Prompting for strict output formatting, Reflective Prompting for systematic rule elimination, and Structured Few-shot Prompting with detailed feature summaries. Building on these base strategies, we propose an Adaptive Prompt Selection method that dynamically chooses the most suitable strategy based on task characteristics. Our system emphasizes robustness through a failure-tolerant output parser, explicit shape and value constraints, and support for multi-sample inference. We defer detailed method descriptions to Section 3.1.

Through systematic evaluation on ARC validation tasks, we observe that: (1) Schema-Constrained Prompting substantially reduces parsing and shape errors but does not improve rule induction quality. (2) Reasoned Prompting helps with more complex rules but can introduce verbosity that occasionally confuses the final answer extraction. (3) Reflective Prompting shows promise for catching execution errors through systematic rule checking. (4) Our Adaptive Selection method achieves better overall performance than any single fixed strategy by matching prompt complexity to task difficulty. These findings suggest that robust ARC performance requires a combination of strict output formatting, explicit reasoning steps, and adaptive strategy selection.

Our contributions are summarized as follows:

- We develop a robust ARC inference pipeline with a failure-tolerant parser that handles diverse LLM output formats, enabling reproducible evaluation of prompting strategies.
- We design five base prompting strategies targeting different failure modes, and propose an Adaptive Prompt Selection method that dynamically selects the most suitable strategy based on task characteristics.
- We conduct systematic experiments under controlled conditions and derive empirical insights into which prompting techniques help with which failure modes, supported by qualitative case studies.

2 Related Work

ARC. ARC was introduced by Chollet (2019b) as a benchmark for measuring human-like generalization from minimal priors. It has inspired a broader community effort, including the ARC Prize, to develop systems that combine perception, program induction, and test-time adaptation (arc, 2025; Chollet, 2019a). Recent analyses have examined LLM performance on ARC tasks and proposed object-based representations (Xu et al., 2023b; Moskvichev et al., 2023). Graph-based constraint formulations and program synthesis approaches have also been explored (Xu et al., 2023a; Ellis et al., 2020).

Prompting for reasoning. Chain-of-thought prompting improves performance on multi-step reasoning by eliciting intermediate steps (Wei et al., 2022; Kojima et al., 2022). Least-to-most prompting decomposes complex problems into simpler subproblems (Zhou et al., 2023). Self-consistency replaces greedy decoding by sampling multiple reasoning paths and aggregating answers (Wang et al., 2023). More recent work explores explicit search over "thoughts" (Yao et al., 2023a) and combines reasoning with acting (Yao et al., 2023b). Iterative reflection enables learning from failures without weight updates (Shinn et al., 2023; Madaan et al., 2023). Self-verification mechanisms help models check their own reasoning (Weng et al., 2023; Lightman et al., 2024). Program-aided approaches offload computation to code execution (Gao et al., 2022). We adapt these ideas to ARC's discrete grid setting where correctness requires strict structural constraints.

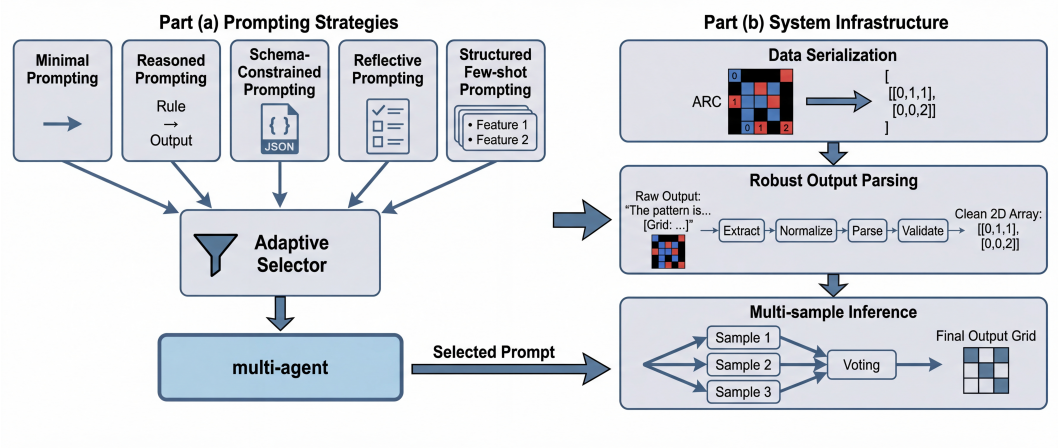

Figure 2: Overview of our method. (a) Prompting Strategies: We design five base strategies targeting different failure modes, and propose an adaptive selection mechanism that analyzes task features to choose the most suitable strategy. (b) System Infrastructure: The selected prompt is processed through data serialization, LLM inference, robust output parsing, and multi-sample voting to produce the final output grid.

3 METHOD

Existing prompting methods for ARC tasks typically apply a single fixed strategy across all problems, overlooking the heterogeneous nature of ARC challenges. In this section, we present our approach that addresses this limitation through adaptive prompt selection, as illustrated in Figure 2. In Section 3.1, we first introduce five base prompting strategies with different design principles, followed by our proposed adaptive selection method that dynamically chooses the most suitable strategy based on task characteristics. In Section 3.2, we describe the system infrastructure that supports robust evaluation, including data serialization, output parsing, and multi-sample inference mechanisms.

3.1 PROMPTING STRATEGIES

Since no single prompting strategy consistently outperforms others across all ARC tasks, we design five base prompting strategies with increasing levels of structure and constraint, each targeting specific failure modes. We then propose an adaptive prompt selection method that combines these strategies based on task characteristics to leverage the strengths of each approach.

Table 1: Summary of our prompting strategies. The first five are base strategies with different design principles, and the sixth is our adaptive selection method.

Strategy	Key Mechanism
Minimal Prompting (Baseline)	Basic instruction requiring only 2D array output without explicit reasoning.
Reasoned Prompting	Encourages brief rule summarization before outputting the answer.
Schema-Constrained Prompting	Enforces strict format and consistency validation on training pairs.
Reflective Prompting	Provides common rule categories and requires systematic elimination.
Structured Few-shot Prompting	Uses detailed feature summaries and structured instruction templates.
Adaptive Selection (Ours)	Dynamically selects the most suitable strategy based on task characteristics.

Minimal Prompting. This serves as our baseline strategy with the simplest instruction: the model is asked to infer the transformation rule from training pairs and directly output the test grid as a

2D array. No explicit reasoning steps or format constraints are imposed. While this approach is straightforward, it often produces outputs with incorrect formats or dimensions.

Reasoned Prompting. Building upon the minimal baseline, this strategy encourages the model to first articulate the inferred rule before producing the final answer (Wei et al., 2022; Kojima et al., 2022). The prompt follows a two-step structure: (1) briefly describe the transformation pattern observed in training examples, and (2) apply this rule to generate the output grid. This decomposition helps the model organize its reasoning process.

Schema-Constrained Prompting. This strategy emphasizes strict output formatting and consistency verification. We add explicit constraints requiring the model to: return only a JSON-formatted 2D array with no additional text; ensure the output grid matches the expected shape; use only valid integer values in the range 0–9; and verify that the inferred rule is consistent across all training pairs. By tightening the output interface, we isolate reasoning errors from formatting failures.

Reflective Prompting. Inspired by self-verification methods (Weng et al., 2023; Shinn et al., 2023), this strategy provides the model with a checklist of common ARC rule categories, such as geometric transformations, color mappings, object manipulations, and pattern completions. The model is required to systematically examine each category against the training examples, eliminate inconsistent hypotheses, and select the unique rule that explains all observations. This structured elimination process reduces ambiguity in rule induction.

Structured Few-shot Prompting. This strategy employs more elaborate instruction templates with detailed feature summaries (Liu et al., 2023). The prompt includes explicit descriptions of grid properties such as dimensions, color distributions, and spatial patterns, presented in a structured format. This additional context helps the model attend to relevant features and produce more accurate predictions.

Adaptive Prompt Selection. Our primary contribution is an adaptive prompt selection mechanism that combines the strengths of the five base strategies. The key insight is that different ARC tasks have different characteristics: some involve simple color mappings where minimal prompting suffices, while others require complex spatial reasoning that benefits from reflective prompting. Our method operates as follows: given a new ARC task, we first analyze its structural features, including grid dimensions, color palette complexity, and apparent transformation type. Based on these features, we select the most appropriate prompting strategy from our suite. For tasks with high ambiguity or complex multi-step transformations, we may apply multiple strategies and aggregate their outputs through majority voting (Wang et al., 2023). This adaptive approach achieves better overall performance than any single fixed strategy by matching prompt complexity to task difficulty.

3.2 System Infrastructure

We frame ARC as a black-box inference problem: given 2–3 training input-output grid pairs and a held-out test input, we query a chat LLM to produce the exact test grid (Brown et al., 2020; Ouyang et al., 2022). Performance in this setting is often limited by system reliability rather than reasoning capability alone, as a correct latent rule is useless if the model returns prose, a malformed array, or a grid with incorrect dimensions. Therefore, our system emphasizes stability through strict input-output contracts, tolerance for common LLM formatting artifacts, and support for multi-sample inference.

Data Serialization. ARC grids are small 2D arrays with values in $\{0, \ldots, 9\}$. We serialize each grid as a compact JSON-like nested list with row order preserved, for example [[0,1,1],[0,0,2]]. Each training input-output pair is explicitly annotated with its shape (H, W), and the test input follows the same format. This representation is both human-readable and straightforward to parse from model outputs. For strategies that enforce shape constraints, we optionally reveal the expected output dimensions without disclosing the actual content. This isolates reasoning errors from trivial dimension mistakes, though it may overestimate performance in realistic settings where output size is unknown.

Robust Output Parsing. LLM outputs frequently contain explanations, Markdown formatting, or multiple candidate arrays. Our parsing function extracts a single 2D integer grid through the following steps: (1) remove Markdown fences and scan from the end of the response to locate the last balanced bracket block; (2) handle common artifacts such as trailing commas and null tokens; (3) use Python's ast.literal_eval to safely convert the string to a nested list; and (4) verify non-empty rectangular shape, integer entries, and values within $[0, 9]$. Failed parses are marked as incorrect. We report parse success rate as a diagnostic metric alongside exact match accuracy.

4 EXPERIMENTS

4.1 EXPERIMENTAL SETUP

Datasets. We evaluate on ARC-style tasks stored in a JSONL format where each line is one task dictionary with a train list and a test list. Each example contains an integer grid under input and (for training examples and for evaluation-time test examples) the corresponding output. Our main results use a public validation file val.jsonl (30 tasks). Optionally, we also report results on a larger and harder set val_hard.jsonl (120 tasks) following the same schema. Each task contains $n \in \{2, 3\}$ training pairs and exactly one test input. Importantly, the task distribution is highly heterogeneous: some puzzles are object-centric (detect shapes and move or duplicate them), while others are pixel-centric (symmetry, tiling, counting, or boundary tracing). This diversity is a key reason that ARC remains challenging for purely text-based LLM interfaces.

Model and Decoding. We use the DeepSeek chat model via the official API with OpenAI-compatible chat completions endpoint (dee, 2026). The model is set to DeepSeek-V3.2 (no-think mode) with temperature 1.0 and max tokens 8192, while other parameters are left at provider defaults. We also provide an optional OpenRouter configuration for deepseek-v3.1-nex-n1 (ope, 2025). All strategies share the same temperature and max tokens settings, and differ only in the prompt template and inference budget (Table 1).

Metric. We report **Exact Match Accuracy**: the fraction of tasks where the predicted test grid matches ground truth at every cell. Because failures can be caused by formatting rather than reasoning, we also report **Parse Success Rate**: the fraction of tasks where our parser extracted a valid rectangular grid with values in $[0, 9]$.

4.2 MAIN RESULTS

Table 2 and Figure 3 summarize the exact-match accuracy and error breakdown of different prompting strategies on val.jsonl.

We summarize the key observations as follows:

(1) **Our method achieves the highest accuracy among all strategies.** Our method combines adaptive prompt selection with multi-agent collaboration to dynamically match each task with the most suitable strategy. By analyzing task characteristics such as grid complexity and transformation type, our approach leverages the complementary strengths of different prompting strategies. The multi-agent mechanism further enables iterative refinement and cross-validation of predictions, resulting in consistent improvements across diverse task types.

Strategy	Acc. (%)	Shape (%)	Color (%)	Rule (%)
Minimal	33.3	13.3	13.3	40.0
Reasoned	53.3	13.3	10.0	23.3
Schema-Constrained	50.0	6.7	6.7	36.7
Reflective	46.7	13.3	0.0	40.0
Structured Few-shot	60.0	23.3	0.0	16.7
Adaptive Selection (Ours)	**66.7**	16.7	0.0	16.7

Table 2: Performance and error breakdown across different prompting strategies on val.jsonl.

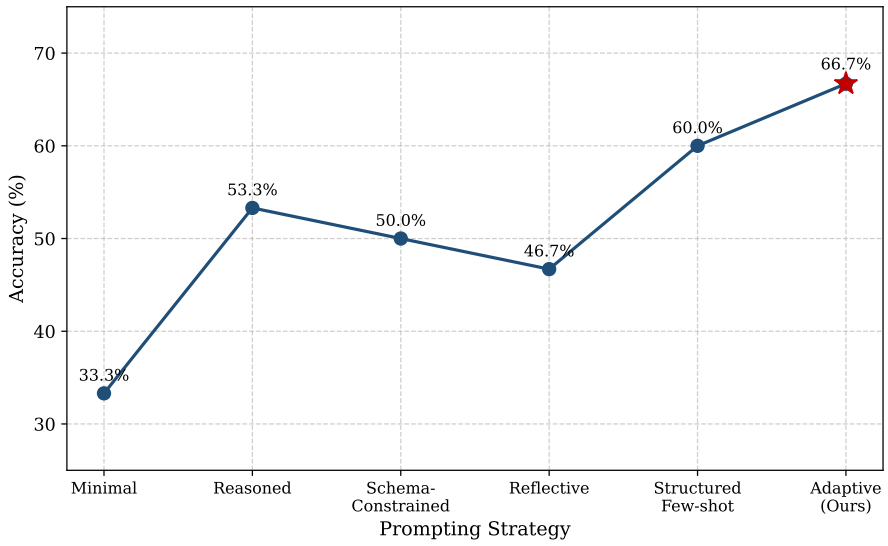

Figure 3: Accuracy comparison across different prompting strategies on val.jsonl. Our Adaptive Selection method achieves the highest accuracy of 66.7%.

(2) **Performance varies significantly across prompting strategies.** As shown in Figure 3, accuracy ranges from the baseline to our method with notable fluctuations among intermediate strategies. We observe that strategies encouraging explicit reasoning (Reasoned Prompting) or providing rich context (Structured Few-shot) tend to perform better, while strategies relying on format constraints alone (Schema-Constrained) or abstract hypothesis elimination (Reflective) show limited improvements. We hypothesize this is related to the model's capability: current LLMs excel at pattern matching and contextual learning but struggle with systematic logical deduction, making concrete examples more effective than procedural frameworks.

(3) **Rule error is the dominant failure mode across all strategies.** Comparing the three error types, rule error consistently accounts for the largest proportion of failures across all strategies, while shape mismatch and unexpected colors contribute less. This reveals that the core challenge in ARC lies in correctly identifying the underlying transformation rule rather than formatting issues. The model can generally produce well-formed outputs, but struggles to infer the correct logical pattern from limited training examples.

(4) **Chain-of-thought reasoning effectively reduces rule errors.** Reasoned Prompting substantially reduces rule error compared to Minimal Prompting, while maintaining the same shape mismatch rate. This demonstrates that explicitly articulating the transformation rule before generating outputs helps the model verify its hypothesis against training examples. The reasoning process acts as a self-consistency check that filters out implausible rules.

(5) **Format constraints and reasoning capability address orthogonal problems.** Schema-Constrained Prompting achieves the lowest shape mismatch by enforcing strict output format validation, yet maintains a high rule error rate. In contrast, Reasoned Prompting shows higher shape mismatch but much lower rule error. This suggests that dimensional correctness and rule identification are independent challenges: format constraints ensure syntactic validity but cannot improve semantic understanding of the transformation.

4.3 Ablation Studies

4.3.1 Results on val_hard

To further evaluate the robustness of our approach, we compare the baseline and our method on val_hard.jsonl, a more challenging dataset containing 120 tasks with greater diversity and complexity. We use the same DeepSeek-V3.2 (no-think mode) model as in the main experiments to ensure

consistency. Table 3 shows the results. The results reveal two key findings: First, the overall accuracy on val_hard is substantially lower than on val.jsonl, reflecting the increased difficulty of tasks involving multi-step reasoning, composite transformations, and subtle pattern variations. Second, our Adaptive Selection method achieves approximately four-fold improvement over the baseline, demonstrating that our approach generalizes effectively to harder tasks. The performance gap between the two datasets also highlights the current limitations of prompt-based methods on complex abstract reasoning tasks, suggesting that future work should explore hybrid approaches combining prompting with symbolic search.

Strategy	Accuracy (%)
Minimal Prompting (Baseline)	2.13
Adaptive Selection (Ours)	**8.51**

Table 3: Exact-match accuracy on val_hard.jsonl (120 tasks).

4.3.2 Case Studies

We analyze representative success and failure cases to provide insights into when and why different strategies succeed or fail. Figure 4 and Figure 5 show typical examples (see Appendix A for more cases).

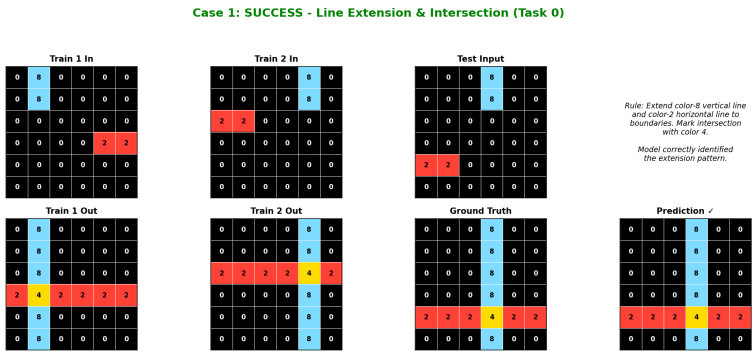

Figure 4: Success case (Line Extension): extending lines to boundaries and marking intersections.

Success case (Line Extension): For tasks involving line extension and intersection marking, our method correctly identifies the transformation rule. The success stems from the rule's simplicity and strong consistency across training examples, allowing the model to correctly decompose it into two sub-rules.

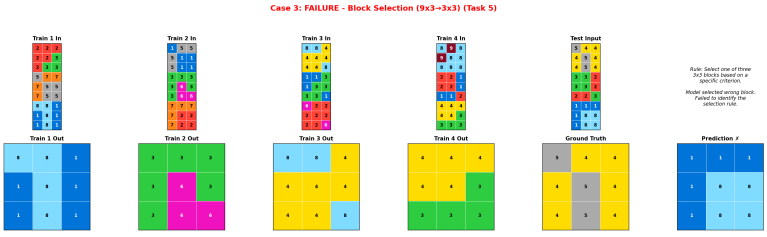

Figure 5: Failure case (Block Selection): selecting a specific block from multiple candidates.

Failure case (Block Selection): For tasks requiring selection of a specific block based on abstract criteria, our method fails to infer the correct selection rule. The model selects the wrong block, indicating difficulty in inducing complex pattern-matching rules from limited training examples.

For more case studies, please see Appendix A.

5 CONCLUSION

In this work, we present a systematic study of prompt engineering strategies for ARC, designing five base strategies and proposing an Adaptive Selection method that dynamically chooses the most suitable strategy based on task characteristics. Our method achieves 66.7% accuracy on val.jsonl and approximately four-fold improvement over the baseline on val_hard.jsonl. Through error analysis and case studies, we identify that rule error is the dominant failure mode and reveal the complementary strengths of different prompting strategies. We hope our findings provide useful insights for future research on prompt design for abstract reasoning tasks.

6 DISCUSSION AND LIMITATIONS

Single model evaluation. Due to resource constraints, we only conduct experiments on DeepSeek-V3.2 (no-think mode). The generalizability of our findings to other LLMs such as GPT-4, Claude, or open-source models remains to be validated. Different models may exhibit varying sensitivity to prompt design choices.

Dataset limitations. Our evaluation is limited to val.jsonl (30 tasks) and val_hard.jsonl (120 tasks). While these datasets capture diverse ARC task types, the relatively small scale may not fully represent the broader ARC challenge. Additionally, we do not evaluate on the official ARC test set due to its held-out nature.

Future work. To address these limitations, future work will focus on: (i) extending experiments to multiple LLMs to validate the generalizability of our adaptive selection approach, and (ii) evaluating on larger and more diverse ARC-style benchmarks to better understand the scalability of prompt-based methods.

7 REFLECTION

Through this semester's Pattern Recognition and Machine Learning course, I have gained a deeper understanding of large language models, including their underlying mechanisms, capabilities, and limitations. This project provided me with hands-on experience in prompt engineering, allowing me to appreciate the nuances of designing effective prompts for complex reasoning tasks.

Working on ARC challenged me to think systematically about how LLMs process and reason over structured inputs. I learned that prompt design is not merely about crafting instructions, but involves understanding the model's strengths and weaknesses, and strategically guiding its reasoning process. The iterative process of designing, testing, and refining prompting strategies deepened my appreciation for the interplay between model architecture and task-specific requirements.

I will continue to conduct research on large language models at Fudan NLP, further exploring their capabilities and applications in complex reasoning tasks.

REFERENCES

Arc prize. https://arcprize.org/, 2025. Accessed: 2026-01-12.

Openrouter: Deepseek v3.1 nex-n1 api. https://openrouter.ai/, 2025. Accessed: 2026-01-12.

Deepseek api documentation. https://api-docs.deepseek.com/, 2026. Accessed: 2026-01-12.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, et al. Language models are few-shot learners. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020. URL https://arxiv.org/abs/2005.14165.

François Chollet. ARC-AGI task data repository. https://github.com/fchollet/ARC-AGI, 2019a. Accessed: 2026-01-12.

François Chollet. On the measure of intelligence. *arXiv preprint arXiv:1911.01547*, 2019b. URL https://arxiv.org/abs/1911.01547.

DAIR.AI. Prompt engineering guide. https://www.promptingguide.ai/, 2024. Accessed: 2026-01-12.

Kevin Ellis, Catherine Wong, Maxwell Nye, Mathias Sablé-Meyer, Luc Cary, Lucas Morales, Luke Hewitt, Armando Solar-Lezama, and Joshua B. Tenenbaum. Dreamcoder: Growing generalizable, interpretable knowledge with wake-sleep bayesian program learning, 2020. URL https://arxiv.org/abs/2006.08381.

Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. Pal: Program-aided language models, 2022. URL https://arxiv.org/abs/2211.10435.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022. URL https://arxiv.org/abs/2205.11916.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *International Conference on Learning Representations (ICLR)*, 2024. URL https://arxiv.org/abs/2305.20050.

Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *ACM Computing Surveys*, 2023. URL https://arxiv.org/abs/2107.13586.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, et al. Self-refine: Iterative refinement with self-feedback, 2023. URL https://arxiv.org/abs/2303.17651.

Arseny Moskvichev, Victor Vikram Odouard, and Melanie Mitchell. The conceptarc benchmark: Evaluating understanding and generalization in the arc domain, 2023. URL https://arxiv.org/abs/2305.07141.

OpenAI. GPT-4 technical report, 2023. URL https://arxiv.org/abs/2303.08774.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, et al. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022. URL https://arxiv.org/abs/2203.02155.

Noah Shinn, Federico Cassano, Edward Berman, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023. URL https://arxiv.org/abs/2303.11366.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. In *International Conference on Learning Representations (ICLR)*, 2023. URL https://arxiv.org/abs/2203.11171.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022. URL https://arxiv.org/abs/2201.11903.

Yixuan Weng, Minjun Zhu, Fei Xia, Bin Li, Shizhu He, Shengping Liu, Bin Sun, Kang Liu, and Jun Zhao. Large language models are better reasoners with self-verification, 2023. URL https://arxiv.org/abs/2212.09561.

Yudong Xu, Elias B. Khalil, and Scott Sanner. Graphs, constraints, and search for the abstraction and reasoning corpus. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, 2023a. URL https://arxiv.org/abs/2210.09880.

Yudong Xu, Wenhao Li, Pashootan Vaezipoor, Scott Sanner, and Elias B. Khalil. Llms and the abstraction and reasoning corpus: Successes, failures, and the importance of object-based representations, 2023b. URL https://arxiv.org/abs/2305.18354.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models, 2023a. URL https://arxiv.org/abs/2305.10601.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations (ICLR)*, 2023b. URL https://arxiv.org/abs/2210.03629.

Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc Le, and Ed Chi. Least-to-most prompting enables complex reasoning in large language models. In *International Conference on Learning Representations (ICLR)*, 2023. URL https://arxiv.org/abs/2205.10625.

A Appendix: Case Studies

We present additional case studies to illustrate the behavior of our method on different task types.

A.1 Success Cases

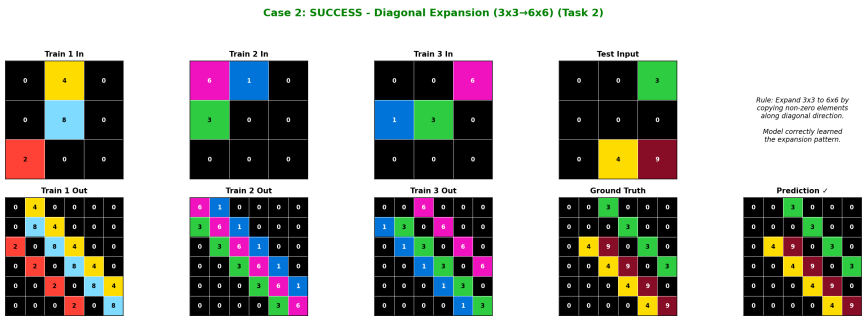

Figure 6: Success case (Diagonal Expansion): 3x3 grid expanded to 6x6 along the diagonal. The geometric transformation is clear and the pattern is easy to induce from training examples.

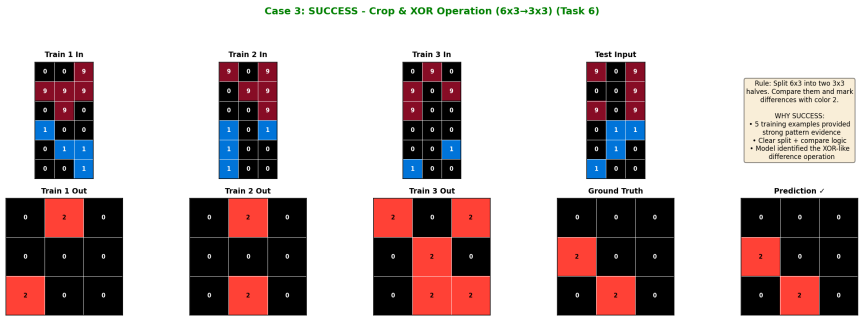

Figure 7: Success case (Crop & XOR Operation): 6x3 grid split into two 3x3 blocks, comparing differences and marking with color 2. Five training examples provide strong pattern evidence.

A.2 Failure Cases

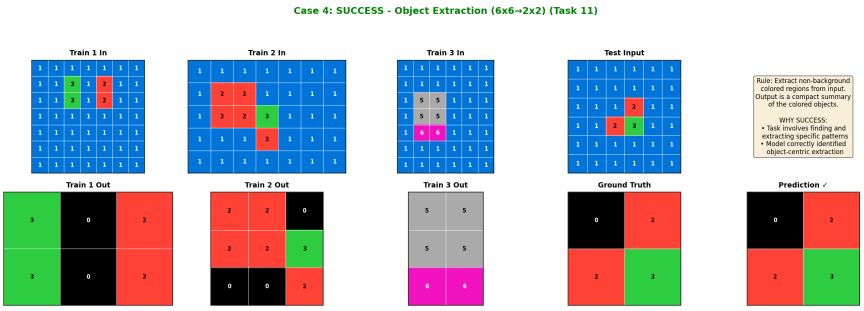

Figure 8: Success case (Object Extraction): extracting non-background regions from 6x6 input to output a 2x2 compact summary. The object extraction pattern is clear and the model correctly identifies the extraction logic.

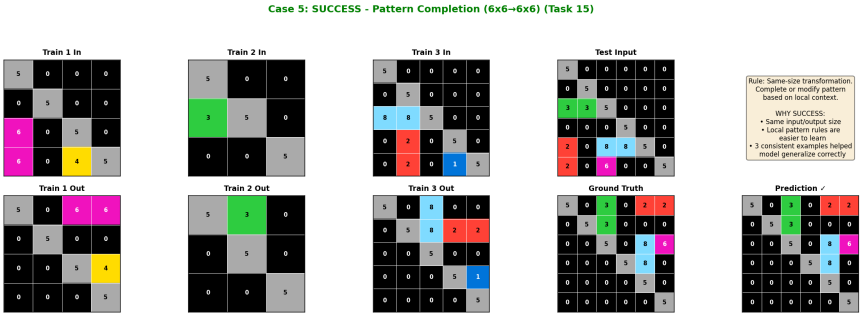

Figure 9: Success case (Pattern Completion): same-size transformation (6x6→6x6), completing patterns based on local context. Input-output size consistency and local rules make learning easier.

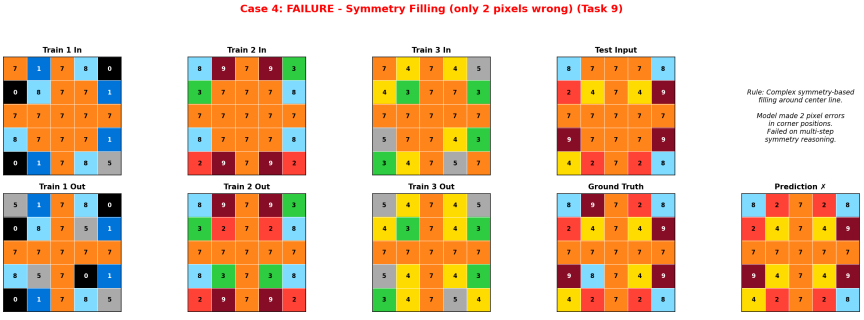

Figure 10: Failure case (Symmetry Filling): complex symmetry filling in a 5x5 grid. The model's prediction differs by only 2 pixels from ground truth, but fails to fully understand the corner filling rule involving multi-step symmetric reasoning.

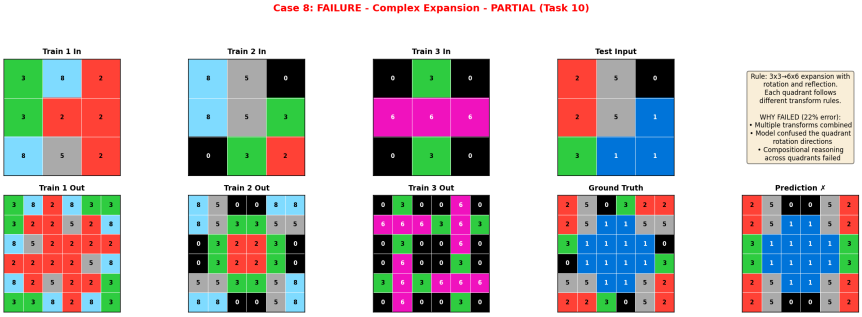

Figure 11: Failure case (Complex Expansion): 3x3→6x6 expansion with different rotation/reflection rules for each quadrant. The model confuses quadrant rotation directions (22% error), failing at compositional reasoning.

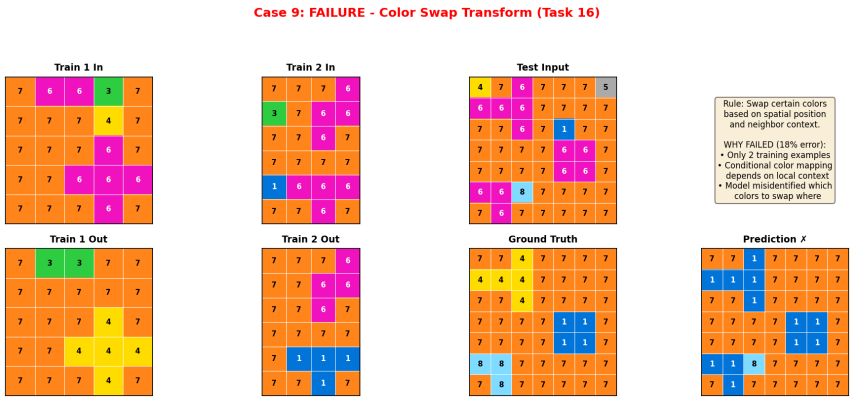

Figure 12: Failure case (Color Swap Transform): color swapping based on spatial position and neighbor context. With only 2 training examples, conditional color mapping depending on local context is difficult to induce (18% error).

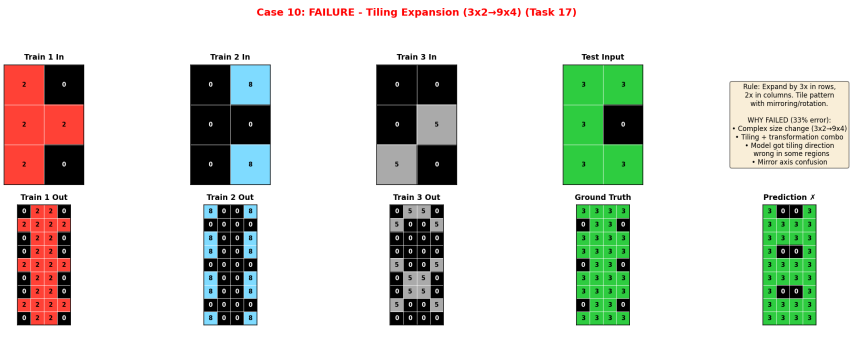

Figure 13: Failure case (Tiling Expansion): 3x2→9x4 tiling expansion with mirroring/rotation. Complex size changes combined with tiling and transformation lead to mirror axis confusion (33% error).

B APPENDIX: PROMPTS

The full prompts for each strategy are included in code/template.py. For transparency, include the exact prompts used for the best strategy and any ablations (e.g., removing coordinate axes or the color legend).