ARC-AGI Final Journal

Juejing Han
jhan446@gatech.edu

1 AGENT FUNCTION

My agent has a **rule-based framework**. After visually analyzing the training input-output pairs, I formalize the transformations as explicit rules for the agent.

My agent's rule catalog includes basic rules, parameterized rules, and combined rules. Basic rules are direct operations that do not rely on learned parameters, such as the Connect Ends Rule, which links two cells under specific color and position constraints. Parameterized rules explore **multiple variations of one transformation type**, such as the Mirror Rule, which tests different orientations, horizontal (left or right) or vertical (up or down), inferred from the data. Combined rules integrate multiple rules to solve more complex problems, such as combining the Crop Rule with the Recolor Rule to resize and recolor the grid.

My agent incorporates **shape recognition** to detect structural or geometric patterns, so it can identify specific forms like brackets or rectangles. It also utilizes the **Breadth-First Search (BFS)** algorithm to locate connected components.

There are 42 rules in my agent's rule catalog for 48 training problems. Each rule is composed of a detector and a solver. The detector verifies whether the transformation explains all training pairs. The solver applies the corresponding rule to generate the prediction, learning parameters when necessary.

Rules are classified into two categories by structural changes: shape-altering and shape-preserving transformations. My agent checks shape compatibility and then determines which rule category to evaluate. During inference, detectors are executed in a priority sequence from simpler to more complex operations. When a rule is verified **through direct pixel value comparison** between the predicted and target outputs, it is applied to the test input to generate the final prediction.

My agent selects from multiple problem-solving approaches depending on what it learns from the problem. Instead of generating a separate output for every possible approach, my agent selects the first valid rule that successfully

explains all training pairs. If no rule satisfies the training pairs, the agent returns an all-zero grid matching the test input dimensions as a safe fallback.

2 AGENT PERFORMANCE

My agent successfully solves 48 out of 48 training problems and 46 out of 48 test problems, achieving 94 solved tasks in total for the ARC-AGI Final performance evaluation (Table 1). Specifically, for training problems, my agent solves 16 in Milestone B, 16 in Milestone C, and 16 in Milestone D. For test problems, my agent solves 15 in Milestone B, 16 in Milestone C, and 15 in Milestone D. There are two failed test problems – Spiral Problem (28e73c20) from Milestone B and Crop & Interior Recolor Problem (e9b4f6fc) from Milestone D.

The reported Gradescope runtime is 0.03 seconds for Milestone B, 0.04 seconds for Milestone C, 0.19 seconds for Milestone D, and 0.36 seconds for the ARC-AGI Final performance evaluation.

Table 1 — Agent performance by category.

Category	Solved Training Problem	Solved Test Problem	Failed Problem	Runtime
Milestone B	16	15	1 test problem	0.03 s
Milestone C	16	16	None	0.04 s
Milestone D	16	15	1 test problem	0.19 s
Final	48	46	2 test problems	0.36 s

My rule-based agent defines a specific rule for each training pair, so every training problem can be solved by its corresponding rule. However, each rule is not narrowly specialized to that training pair – it is generalized to address a broader range of similar problems. For example, my agent employs a flexible color mapping mechanism that derives color relations from training pairs rather than relying on predefined color-value mapping. My agent sets the default background color to black, and it learns the actual background color from the input-output pairs whenever applicable. It also learns transformation parameters, such as mirroring orientations or rotation angles, to adapt to different patterns.

3 AGENT PERFORMANCE – SUCCESSES

My agent performs well on all 48 training problems because it has direct access to the input-output pairs. These patterns are fully observable and deterministic, so my agent can reliably identify the correct rule for each training problem.

The test problems are more challenging because their outputs are hidden, requiring generalization from the training transformations to capture the exact pattern for the test pairs. My agent successfully solves 46 out of 48 test problems. The reason is that my agent utilizes a flexible color mapping mechanism and learns transformation parameters to infer the correct pattern from the test input. For example, multiple rotation degrees (90, 180, 270) for the rotation-based problems (such as 6150a2bd and ed36ccf7), and a set of logical operators (AND, OR, XOR, NOR) for the logical-operator-based problems (such as 6430c8c4 and 195ba7dc).

Overall, my agent performs well on simple linear-scan problems without learned parameters (such as 22eb0ac0, 9af7a82c, and bbc9ae5d), parameter-dependent problems (such as 6150a2bd, e98196ab, and cf98881b), multi-stage problems that require sequential transformations (such as b94a9452, 3de23699, and f35d900a), and complex structural problems (such as 7b6016b9, b2862040, and d931c21c).

Four success examples and two struggle examples are presented in this report. Each case includes several training pairs and one test pair. For brevity, only up to three training pairs are shown.

3.1 Success Example 1 - 22eb0ac0

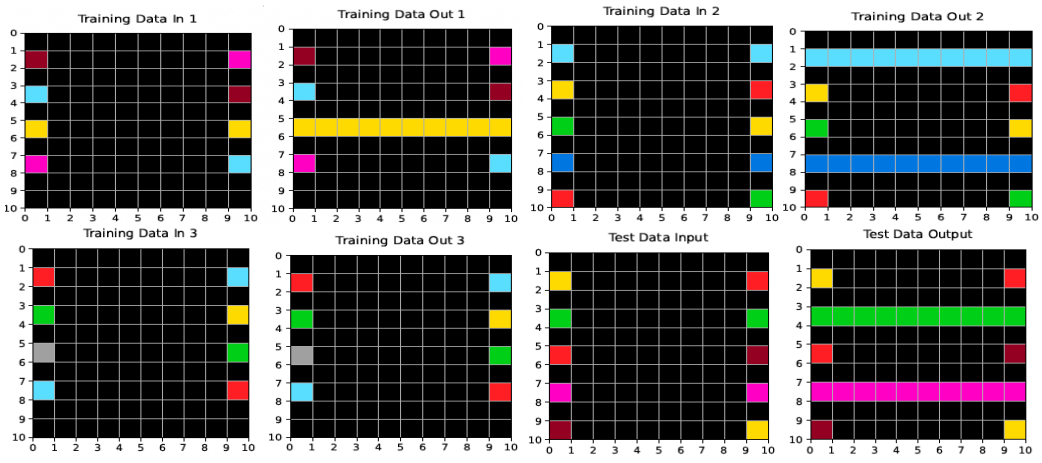

Figure 1— Success Example 1 showing three training pairs (top row and bottom left) and one test pair (bottom right).

The rule for problem 22eboaco (Figure 1) is **straightforward**. For each row, my agent inspects the leftmost and rightmost cells. If both endpoints share the same non-background color, it fills the entire row with that color. If this rule accounts for all training input-output pairs, my agent applies the same transformation to the test input to generate the final output.

3.2 Success Example 2 - 6150a2bd

The rule for problem 6150a2bd (Figure 2) is **parameter-dependent**. My agent checks whether the task is a rotation by applying a set of rotation parameters (0°, 90°, 180°, and 270°) to each training input and comparing the transformed grid to the corresponding output grid. My agent identifies the correct rotation parameter when all transformed inputs match their training outputs. Once a consistent rotation parameter is found across the training pairs, my agent applies the same rotation to the test input to generate the final output. The learned parameter setting enables my agent to solve similar rotation tasks with different rotation degrees, making it robust across variations of the same underlying transformation.

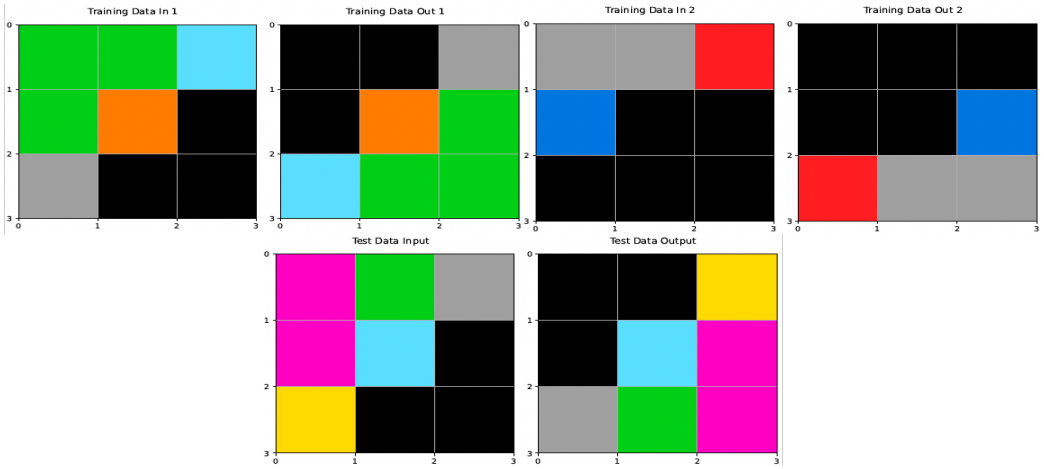

Figure 2 — Success Example 2 showing two training pairs (top row) and one test pair (bottom row).

3.3 Success Example 3 - b94a9452

My agent solves problem b94a9452 (Figure 3) by a **combination of rules**. It first applies the Tight Crop Rule to remove all-zero rows and columns. On the cropped grid, which contains two non-background colors, my agent applies the Color Swap Rule to swap those two colors. If this combined rule explains all training input-output pairs, my agent applies the same sequence of transformations to the test input to generate the final output.

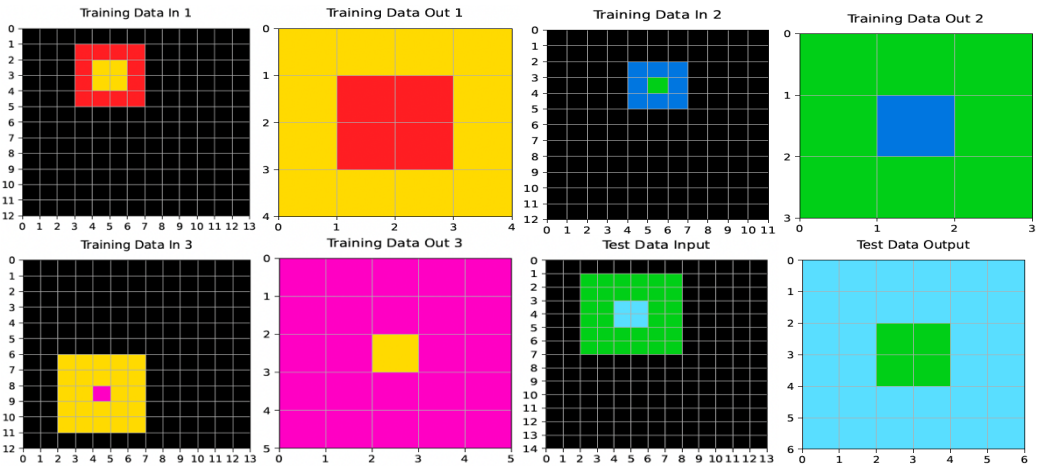

Figure 3—Success Example 3 showing three training pairs (top row and bottom left) and one test pair (bottom right).

3.4 Success Example 4 - 7b6016b9

My agent solves problem 7b6016b9 (Figure 4) by **utilizing BFS for complex structural identification**. It first identifies the shape color by checking which color forms a region that encloses interior empty spaces. It then separates the input grid into the shape, the interior holes within the shape, and the exterior background region outside the shape. My agent recolors the interior holes and exterior regions with the target colors learned from the training pairs. If this rule accounts for all training input-output pairs, my agent applies the same transformation to the test input to generate the final output.

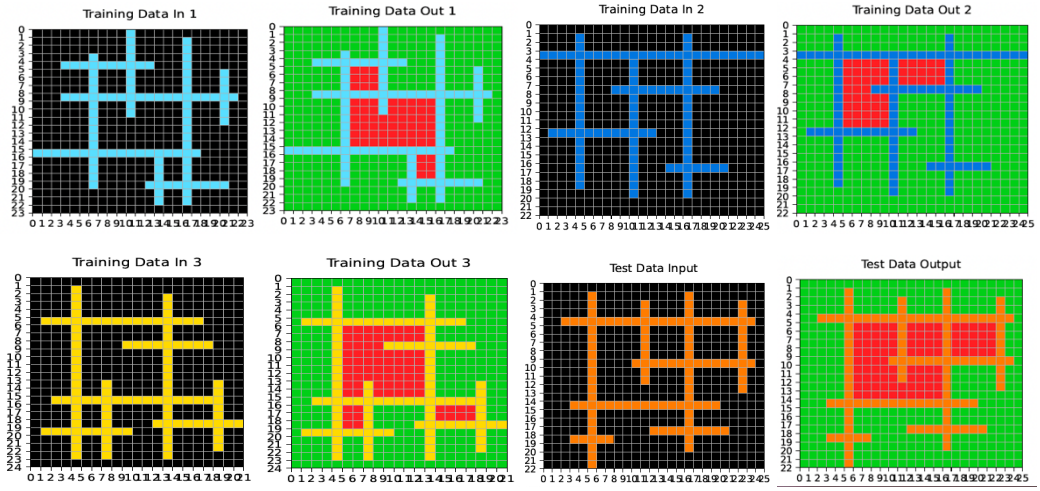

Figure 4—Success Example 4 showing three training pairs (top row and bottom left) and one test pair (bottom right).

4 AGENT PERFORMANCE – STRUGGLES

As mentioned in Section 2, my agent struggles with two test problems: 28e73c20 and e9b4f6fc. Although it successfully passes the corresponding training tasks, it fails to generalize to these two hidden test cases. The breakdown in its performance arises from the limitation of its rule-based reasoning.

4.1 Struggle Example 1 - 28e73c20

My agent approaches problem 28e73c20 (Figure 5) using a parameter-dependent rule. It first identifies the background color and the spiral color from the input-output pairs. Then, my agent constructs spiral patterns by combining orientation parameters (clockwise vs. counterclockwise and different flip operations). In this way, it generates spiral patterns that start from different corner cells and expand inward.

This rule design successfully reproduces all the training outputs but fails on the test case. **The breakdown lies in the fixed set of spiral candidates.** If the test pair does not match any of the defined patterns, my agent cannot produce the correct output. This illustrates the downside of rule-based reasoning: my agent has a limited capacity to generalize from the input data.

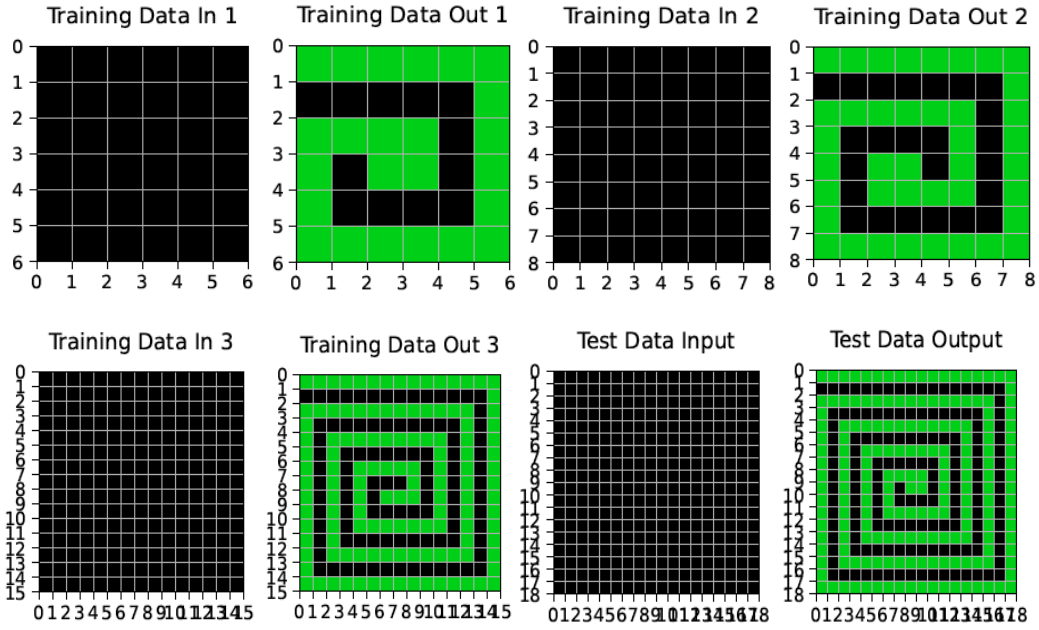

Figure 5 — Struggle Example 1 showing three training pairs (top row and bottom left) and one test pair (bottom right).

4.2 Struggle Example 2 - e9b4f6fc

My agent approaches problem e9b4f6fc (Figure 6) with BFS as its core method for structural identification. It extracts the largest connected component (LCC) of non-background cells and separates the input grid into LCC and non-LCC regions. My agent scans the LCC region to determine the unique colors (source colors). Then, it scans the non-LCC region for the source color and identifies the correct target color by examining the left and right adjacent neighbors of source-color cells. After constructing the color mapping for source-target color pairs, my agent recolors the LCC region accordingly.

This rule design reproduces all the training outputs but fails on the test case. **The breakdown likely arises from the color mapping identification step.** In the test input, there may be horizontally or vertically touching color pairs in the non-LCC region, requiring more flexible mapping logic. Because my agent relies on strict neighbor-based mapping, it fails to generalize to more complex edge cases. This demonstrates the limited capacity of the rule-based agent to handle complex pattern variations not explicitly covered during training.

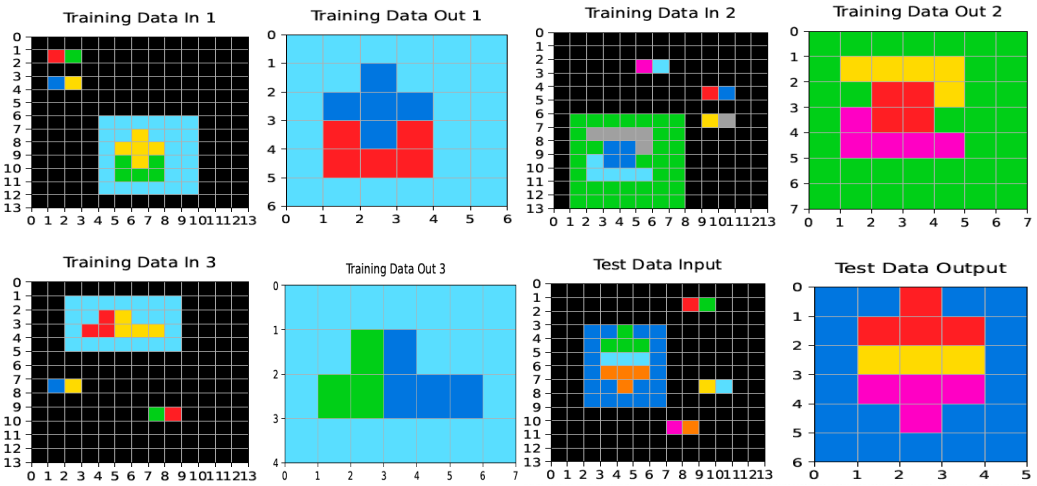

Figure 6— Struggle Example 2 showing three training pairs (top row and bottom left) and one test pair (bottom right).

5 AGENT DEVELOPMENT

The overall approach of my agent design is **a rule-based architecture with incremental expansion**. I built an **ever-expanding catalog of transformation rules** after inspecting the problem visuals. Each rule contains a detector that checks

whether the rule can explain all training pairs and a solver that applies the rule to generate the predicted output. When encountering a problem, my agent first analyzes the structural change and activates the corresponding category of transformations – either shape-altering or shape-preserving. It then evaluates the detectors in a fixed priority order, from the simplest to the most complex, and selects the first rule that fully matches the training set to produce the final output.

Parameter-dependent rules play a crucial role in improving generalization, requiring my agent to infer colors, orientation parameters, or structural relationships from the training pairs. Multi-stage rules that combine simpler operations and BFS-based structural analysis help solve more complex problems.

Throughout the milestones, my agent incrementally expands its rule catalog. I **refined the overall approach rather than replacing it with a different design**. During Milestone A, my agent solved one training problem using a predefined rule, which motivated the idea of developing a rule-based agent. During Milestone B, the rule-based agent used 12 rules and successfully solved 31 out of 32 problems, confirming the effectiveness of the rule-based strategy. During Milestone C, the catalog grew by 15 additional rules and achieved 32 out of 32 successes. Meanwhile, I introduced a finer rule categorization strategy that improved local runtime by 0.002 seconds across the 16 training problems. During Milestone D, the rule catalog was further expanded with 15 new rules, bringing the total to 42 rules across all milestones.

This heuristic-driven and rule-based design balances interpretability with incremental problem-solving ability, but it also limits generalization – my agent can only solve problems that match one of the rules in its catalog.

6 AGENT VS. HUMAN

My agent solves simple problems in a similar way I do, but it handles more complex tasks with different strategies.

6.1 Similarity

For simple problems, my agent applies straightforward rules, and the process is similar to the human approach. For instance, as analyzed in Section 3.1, to solve problem 22eboaco, my agent scans the input grid and fills the entire row if the leftmost and the rightmost cells share the same non-background color. I would

do something similar – manually inspect the two end cells in each row and connect them once they are in the same non-background color.

Additionally, my agent infers the correct rule for a problem by examining all training pairs and applying the transformation once the rule matches each pair. I follow a similar process to determine the correct transformation.

6.2 Difference

For more complex problems, my agent applies complex rules, including explicit algorithms that a human would not apply. For instance, as analyzed in Section 3.4, to solve problem 7b6016b9, my agent utilizes BFS to identify the connected components and separates the grid into the shape, the interior holes within the shape, and the exterior background region outside the shape. Then, it learns the color mapping and recolors the interior holes and exterior areas with the target colors. As a human, I would visually identify these regions rather than running a BFS traversal.

Additionally, my agent is rule-based and depends on explicit rule matching, whereas I rely more on intuition and visual inspection.

My agent is significantly efficient than a human. During the performance evaluation, Gradescope reports a runtime of 0.36 seconds to solve 94 out of 96 problems, overwhelmingly faster than human speed. However, humans have better generalization abilities. My agent cannot solve unseen tasks unless a matching rule is in its rule catalog, and it relies on manually added rules, while I can learn new rules autonomously. This contrast between "manually adding rules" and "self-learning new concepts" captures the fundamental difference between my rule-based agent and human reasoning.