My Journey

Initially, I considered using a Convolutional Neural Network (CNN) model. However, I soon realized that the ARC tasks make it nearly impossible for a single traditional AI model to achieve a satisfactory score due to its diversity. Combining two AI models seemed too risky given the time constraints. Instead, I opted for a simpler approach: a rule-based model that generates all possible combinations of transformational primitives up to a specific depth and applies them to each task.

This initial model implementation included about 10 primitive methods but faced runtime errors on Gradescope when I added more methods. To address this, I planned adding case-based reasoning. However, the diversity of tasks made it challenging to determine similarity between cases or decide which test cases to store. Instead, I implemented basic heuristics, such as discarding combinations like rotating 90 and 270 degrees together. These simple heuristics worked surprisingly well.

As I added more computationally demanding primitives, I introduced a dynamic input grid feature check to predic whether transformations in a combination were valid and meaning for the current grid. This greatly improved efficiency and eliminated runtime issues completely. Finally, I enhanced the model’s color mapping by introducing a dynamic color transformation logic that learns from training data. When a consistent color substitution pattern appeared across all training sets, the model applied this knowledge, boosting accuracy by 50%.

Key Learnings

Before this project, I didn’t quite enjoy any AI classes, primarily because I struggled to understand the inner workings in one parameter adjustment. However, this project was different.