Approaches on Abstract Reasoning Corpus

Woletemaryam Liyew
20231210
GIST
Gwangju, South Korea
woleteml@gm.gist.ac.kr

Dongjin Lee
20205130
GIST
Gwangju, South Korea
leedongjin@gm.gist.ac.kr

Donguk Ryu
20214072
GIST
Gwangju, South Korea
yujingon@gm.gist.ac.kr

ABSTRACT

Abstract and Reasoning Corpus is a measurement suggested to evaluate the AI. In each tasks, human-like intelligence is assumed, so that AI is requested to make a prediction from test input and train input-output pairs. We made three models with distinct approach and only decision tree model with augmented data solved the secret evaluation task. As we extract features explicitly in a form such that human can interpret, treating various tasks became more harder. Decision tree model implicitly guessed geometrical pattern or rule from augmented data and it scored 2% of correctness. Although its predictions are partially correct, whole correct predictions are a part of them, so that further improvement would be needed. More augmentation method or regularization by adding noise was suggested to improve the model further.

1 INTRODUCTION

1.1 Abstract and Reasoning Corpus

The Abstraction and Reasoning Corpus (ARC) is a benchmark in the field of artificial intelligence, focusing on measuring a system's general cognitive abilities. Developed by François Chollet, the ARC challenges AI models to solve tasks that involve recognizing patterns and abstract reasoning. Each task in the ARC is presented as a small number of input-output grid patterns, where the model must deduce the underlying rule to transform the input grid into the output grid. The ARC aims to assess the generalization power of AI beyond specialized tasks, emphasizing the capability to understand and manipulate abstract concepts. This benchmark is considered a significant step in evaluating and advancing AI towards more human-like reasoning and problem-solving abilities.

1.2 Intelligence

In "On the Measure of Intelligence" [4] by François Chollet, intelligence is defined as the ability to adapt to a wide variety of environments with a limited scope of prior knowledge and limited resources.

- Generalization Ability: Chollet emphasizes that true intelligence should not be limited to the capacity to perform well on familiar tasks but should instead be measured by the ability to generalize to novel scenarios. An intelligent system should be able to find solutions in situations it hasn't encountered before.
- Skill Acquisition Efficiency: Intelligence is also characterized by the efficiency with which a system can acquire new skills. This involves the ability to learn quickly and with fewer resources, which is indicative of a system's flexibility and adaptability.
- Resourcefulness: A key aspect of Chollet's definition is the idea that intelligence involves accomplishing goals across a wide range of situations with limited resources. This includes not only the data available for learning but also computational resources and prior knowledge.

So Francis chollet introduced a bechmark data set called the sbatraction and reasong corpus(ARC) that will measure the intelligence of artificial intelligent agents. Below Figure 1 shows a sample task of the ARC given the input and output pair the agent is then required to predict the output.

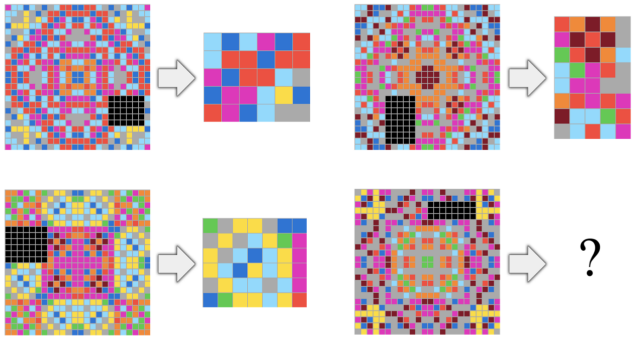

Figure 1: Sample task: A task where the implicit goal is to complete a symmetrical pattern. The nature of the task is specified by three input/output examples. The test-taker must generate the output grid corresponding to the input grid

In ARC tasks, the objective is to discern the pattern or rule that the input grid follows to produce the output grid. The rule might involve color, shape, the number of occurrences of certain elements, symmetry, or other abstract features. Solving these tasks requires a form of intelligence that can generalize from the given examples and apply the inferred rules to new, unseen situations.

1.3 Human versus AI

Humans typically achieve 80 percent success rate in solving the ARC tasks, but existing algorithms have only solved 30.5 percent of the ARC tasks which is current world record achieved through combinations of various algorithms. Humans, including young children, often excel at ARC tasks because they can intuitively grasp abstract concepts and patterns. They can quickly identify relationships and apply them to new situations with minimal data. Also Humans are highly adaptable and can apply learned concepts to a wide range of scenarios. This flexibility allows them to generalize from a few examples and understand new rules or patterns without needing vast amounts of data. AI systems, particularly those based

Machine learning and deep learning , GIST,
2023. ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

on machine learning, typically require large amounts of data to learn. The few-shot learning scenario presented by ARC, where only a few examples are provided to infer the rule, is challenging for many AI models.

1.4 Dataset

The training and public evaluation sets can be downloaded on the ARCathon Website and it contains 800 json files (400 tasks for the training dataset and 400 tasks for the public evaluation dataset). And there is a secret evaluation set which has the same form as the public one, except that the test outputs are replaced by trivial entries, like an array of zeros. Each task has a dictionary with two fields; train and test field. The train field shows 2-5 example input-output pairs (typically 3 pairs) and the test field shows one test pair. Given datasets contain input-output pairs of different shapes. The task data is availble at github.com/fchollet/ARC.

training / eb5a1d5d.json	Input shape	Output shape
Train 0	(23,27)	(5,5)
Train 1	(22,25)	(3,3)
Train 2	(21,22)	(7,7)
Test 0	(26,27)	(9,9)

Table 1: Input-output shapes

For example, eb5a1d5d in the training dataset like Figure 2. It has 3 input-output pairs of 'train' (Train 0,1,2) and 1 pair of 'test' (Test 0). As shown in Table 1, all of the input and output shapes are different.

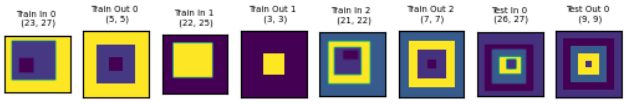

Figure 2: Sample task with different shapes

It shows the 2d array form as above. This 2d array consists of digits 0-9. Digits 0 to 9 means colors dark (navy) to bright (yellow). The core knowledge priors assumed by the ARC are as below.

Objectness: being able to parse grid into objects.

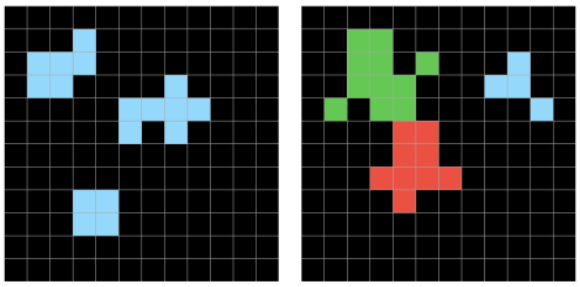

Figure 3: parsing grids into objects

Goal-directness:the solution to any given problem should not involve arbitrary or overly complex sequences of actions. Instead, the solution is typically straightforward and involves recognizing patterns,relationships and applying transformations that directly lead to the goal state.

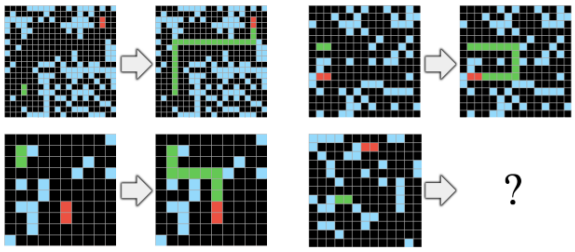

Figure 4: A task involves changing obstacles and reaching goals.

Numbers and Counting: Being able to count digit numbers for pixels or objects.

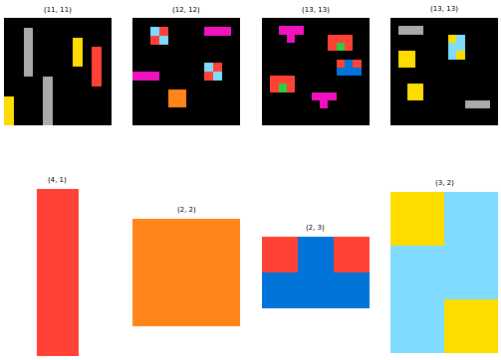

Figure 5: Task which requires AI to find the below objects which appear once in each above inputs.

Basic Geometry and Topology: AI should discern simple geometrical objects such like line or square. Also, AI would be assumed that it might understand topolgical relationship between objects.

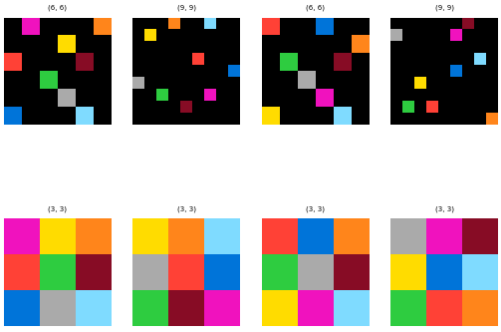

Figure 6: Topological task which requires below color ordering as outputs from their relative positions in above inputs.

Along with the data set there is a website interface provided for users to try solving problems by their hands and the interface looks in the figure below. on the middle users see the input and their respective outputs and on the right is current grid. users use controls on the right to construct the output grid.

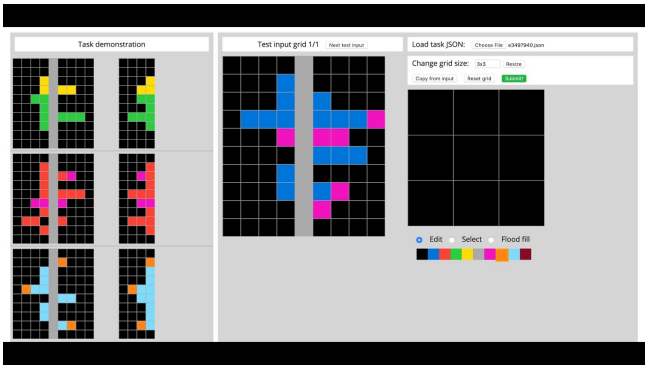

Figure 7: Website interface for ARC

1.5 Related Works

The paper called "Communicating Natural Programs to Humans and Machines" [1] by Samuel Acquaviva and colleagues introduced a novel approach to solve the ARC. This methodology focused on the integration of natural language descriptions into the ARC tasks, which creates their own corpus called language complete abstration and reasoning corpus(LARC). LARC is created by augmenting the original ARC tasks with language by adding natural language instructions.They genertaed the instructions in equivalence way in which humans could correctly produce the exact outputs with out needing the original input output examples.

The paper entitled with "Neural-guided, Bidirectional Program Search for Abstraction and Reasoning" [3] introduces two novel approaches: first, applying DreamCoder for program synthesis to create symbolic abstractions, enhancing the capability to solve complex ARC tasks. Second, a human-inspired reasoning algorithm is designed, which uses a search graph and deductive reasoning based on function inverse semantics. This leads to a neural-guided bidirectional search algorithm, demonstrated to be effective in ARC, 24-Game tasks, and a 'double-and-add' arithmetic puzzle. Aslo the authors mentioned that their approaches was not brute force attack.

The paper titled "A Neurodiversity-Inspired Solver for the Abstraction & Reasoning Corpus (ARC) Using Visual Imagery and Program Synthesis" [2] by James Ainooson and colleagues from Vanderbilt University presents a system for reasoning and solving tasks from the Abstract Reasoning Corpus (ARC). The approach uses a program synthesis method, employing a domain-specific language called Visual Imagery Reasoning Language (VIMRL) for reasoning about ARC tasks. The system incorporates high-level functions that determine their arguments through local searches on a given task item.

The paper entitled "Unraveling the ARC Puzzle: Mimicking Human Solutions with Object-Centric Decision Transformer" [7] employ an approach that combines the decision transformer with an object detection algorithm known as the push and pull method.This research showed that the need for better data collection tools and robust training data sets to further improve the decision transformer in the AGI.

Deep Learning Models: Some researchers have attempted to use deep learning models like Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs). These models, while powerful in pattern recognition, often struggle with the ARC tasks due to their requirement for large amounts of data and difficulty in extrapolating general rules from specific examples.

Abstract Reasoning with Graph Abstractions (ARGA), which represents images using graphs and then searches for the correct program within a Domain-Specific Language (DSL) based on the abstracted graph space.

The paper entitled "An Approach to Solving the ARC Challenge" [6] by John Chong Min Tan tried to solve the abstraction and reasoning corpus problem using Large Language Models(LLMS) like GPT-4 through prompt engineering to perform few-shot learning tasks. Few-shot learning involves understanding tasks and generating correct outputs from a limited number of input-output examples.. It focuses on generating detailed descriptions for the input and the output pairs and their respective mapping relations descriptions.The paper showed some success in solving some of the ARC tasks and also the paper recommended that combining LLMS with multi agent system could solve most of the ARC tasks.

There is also a paper called "Solving Abstract Reasoning Tasks with Grammatical Evolution" [5] which introduces a method called grammatical evolution a type of genetic algorithm to provide solutions for the ARC tasks.To apply this grammatical evolution the authors have developed DSL suited for image transformation.Using this method they solved 3

2 TRIALS AND MODELS

In pursuit of advancing AI's proficiency in these areas, this paper presents a comprehensive study employing three distinct approaches to tackle the ARC challenge. Because the output would be the form of matrix with discrete numbers and they have not mathematical relationship, predicting each integer might be thought as a classification problem for each integers. We first approached the tasks in human-likely method as if each distinct integers represent just distinct colors. Then we approached in more implicit way to make AI find the rule of the task itself.

The first approach, termed "One by One" is a methodology focusing on the analysis of each task. This method meticulously dissects individual elements of the tasks, applying a sequential reasoning process. It is predicated on the hypothesis that a detailed, step-by-step analysis can unveil underlying patterns and rules in the data, allowing for effective problem-solving even with the limited data typical of ARC tasks.

Our second approach, "Decision Tree and Data Augmentation," uses machine learning techniques with contemporary data augmentation strategies. The decision tree algorithm, known for its simplicity and interpret ability, is employed to formulate basic reasoning paths. Concurrently, data augmentation is utilized to artificially expand the data set, aiming to enhance the decision tree's ability to generalize from limited examples.

Lastly, the "U-Net" [8] approach applies a convolutional network architecture commonly used in image segmentation tasks. U-Net's architecture, characterized by its 'U'-shaped symmetrical structure, enables precise localization and is adept at handling the variety of patterns and layouts present in ARC tasks.

2.1 Classifying and Solving One-by-One

Chollet [4] presented four types of prior knowledge for AI. We first extracted features based on them and made a model for solving specific tasks. Suggested prior knowledge types are as below.

- Objectness
- Goal-directness
- Numbers and Counting
- Basic Geometry and Topology

Integer zero is regarded as a background and connected nonzero integers form an object. To discern objects, we applied DFS-algorithm. Each objects have its own attributes to save features. The number of distinct integers or the number of each integers may be a feature for counting. Geometrical features such like symmetric property was also included. Equality between two objects was defined so that we can check whether certain object in input repeats in the output. Other relationships between objects were concerned, so that each objects have a list indicating objects adjacent to it and horizontally or vertically apart from it.

To make a prediction based on goal-directness, we needed to classify tasks. Shapes of input and output pairs were effective in dividing a task categories. In most cases, output shape were fitted by ordinary linear model regarding input shape as a predictor. Tasks that such linear relationship does not hold would require topological representation or the specific object as an output. Simply decision tree model was enough to solve object-selecting task, in that case.

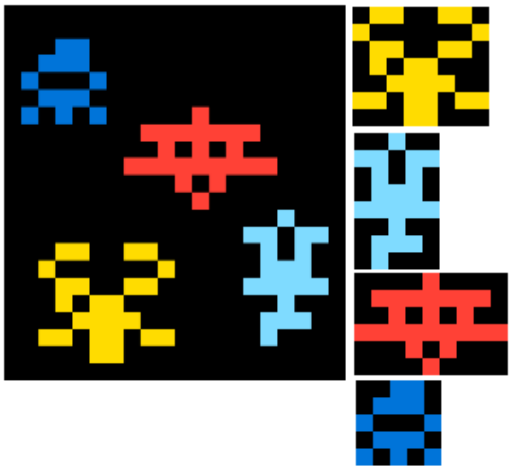

Figure 8: Left is a test input of one object-selecting problem and right ones are its objects discerned. Desired output is the red object, which has a symmetric property.

Model solved 5 tasks in training set and 4 tasks in evaluation set. However, did not make a proper prediction for hidden evaluation tasks. Objects and features were extracted explicitly in this way, however tasks were so various and some outputs request to do more than one sub-task. In addition, some outputs request to modify object in its input, so that we have to find where does modification occurs, not just which object would be modified. Specific classified tasks were done correctly, but it contains only few predictions.

Just following human's decision algorithm does not give effective process for finding goal of the encountered new task. Clearing other types of prior knowledge made explicitly guessing goal harder because each tasks consider distinct prior knowledge. Therefore we needed to make AI find the goal itself from the input.

2.2 Decision Tree and Data augmentation

This method basically consists of three steps. These are feature extraction, model training and finally making predictions. We have to make each of the tasks ready for making predictions for the unknown given input tasks.

2.2.1 Feature extraction. We have used different functions for extracting features from each of the tasks and how they work. These functions will give us a new extracted feature which is basically the main purpose of data prepossessing.

Feature extraction Functions

- getAround: This function will return the values that are around a specific pixel. Given pixel coordinates (i , j) in the grid input the function will return the values that are around this specific pixel within a specified size. then we will store this extracted around values for a specific pixel. this function will help to capture local patterns.
- getx: this function will create a feature vector for a given cell in the grid. for each cell coordinates (i,j) this function will compile features like the row and column lengths , unique values in the row and the column, and the unique values around the cell(using get around). this function will transform each pixel and its context into a numerical vector that will be used for the machine learning model to make predictions.
- getxy: this function generates the feature vectors(X) and the labels(Y) for cells in a grid. the function iterates over all cells in an input gird using the getx to create feature vectors and collecting the corresponding output values as labels. it prepares the training data for the machine learning model where X is the feature matrix and y is the label vector.

So below we have listed all of the features that are extracted in the extraction step.

Extracted Features

- Pixel Features: The coordinates of the cell (i, j).
- Grid Dimension Feature: The total number of rows and columns in the grid.
- Unique Value Feature: The count of unique values in the cell's row and column, and in the cells around it. This helps in understanding the diversity of values in the immediate vicinity of a cell.
- Surrounding Cell Value feature: The actual values of the cells surrounding the current cell, as obtained by 'getAround'.

  In order to increase the diversity of the training data by creating different variations of the input-output pair we used different data augmentation techniques. we removed specific columns and then create combinations and permutations of the remaining columns. we applied the permutations and combinations to all the input and output metrics. Also we augment the data by flipping and rotating the input and output metrics. Flips used are horizontal and vertical and

then we rotated this flip-ed metrics in three different angles(90,180,and 270 degrees). Each transformation is applied to both the input and the output metrics.

Below figure shows sample code snippet that we have used for the augmentation using flip and clockwise rotation.

```
def get_flips(inp,oup):
    result = []
    n_inp = np.array(inp)
    n_oup = np.array(oup)
    result.append((np.fliplr(inp).tolist(),np.fliplr(oup).tolist()))
    result.append((np.rot90(np.fliplr(inp),1).tolist(),np.rot90(np.fliplr(oup),1).tolist()))
    result.append((np.rot90(np.fliplr(inp),2).tolist(),np.rot90(np.fliplr(oup),2).tolist()))
    result.append((np.rot90(np.fliplr(inp),3).tolist(),np.rot90(np.fliplr(oup),3).tolist()))
    result.append((np.flipud(inp).tolist(),np.flipud(oup).tolist()))
    result.append((np.rot90(np.flipud(inp),1).tolist(),np.rot90(np.flipud(oup),1).tolist()))
    result.append((np.rot90(np.flipud(inp),2).tolist(),np.rot90(np.flipud(oup),2).tolist()))
    result.append((np.rot90(np.flipud(inp),3).tolist(),np.rot90(np.flipud(oup),3).tolist()))
    result.append((np.fliplr(np.flipud(inp)).tolist(),np.fliplr(np.flipud(oup)).tolist()))
    return result
```

Figure 9: Sample code snippet for flip and rotation.

Next we trained the machine learning model(Decision Tree) on the augmented data set which increases its generalization ability to new unseen tasks.

Our model makes correct predictions for 33 tasks on the evaluation set and 2 tasks on the hidden task set.

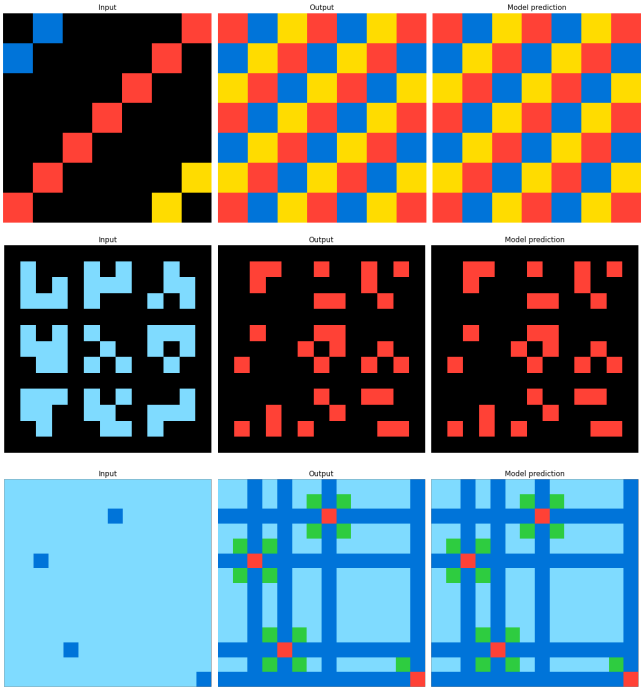

Figure 10: tasks solved by decision tree model. The model itself found the pattern and filled it.

2.3 Image Training Model with U-Net

We use padding to make the input and output 2D arrays the same size. This size is adjusted to the size of one width of padding for the largest shape of each input and output. Predicted shape of Test 0 output is used for the secret evaluation set. In the case of eb5a1d5d, the shape of test 0 is the largest among the input shapes, so it is determined as (28,29) by padding it with 1. And the output shape also sets the padded shape of (11,11). Figure 11 shows the examples of the padding.

Since both input and output fields are 2D arrays, we considered models based on encoder-decoder and decided U-Net among them. Usually, in the encoding stage, the dimensionality is reduced while increasing the number of channels to capture the characteristics of the input image, and in the decoding stage, only low-dimensional encoded information is used to reduce the number of channels and increase the dimension to restore a high-dimensional image. However, detailed location information about image objects is lost as dimensionality is reduced in the encoding step, and since only low-dimensional information is used in the decoding step, the loss of location information cannot be recovered. The basic idea of U-Net is to extract image features using not only low-dimensional but also high-dimensional information and at the same time enable accurate location determination. To achieve this, a method of concatenating the features obtained from each layer of the encoding stage to each layer of the decoding stage is used. The direct connection between the encoder layer and the decoder layer is called a skip connection.

Therefore, the 2D array is converted into an image for training. A 3 layer U-Net was used for image training and it gives predicted image output. The predicted image output was cropped and rescaled to the previously predicted test output shape like Figure 12. The model solved 9 tasks in training set and 3 tasks in evaluation set.

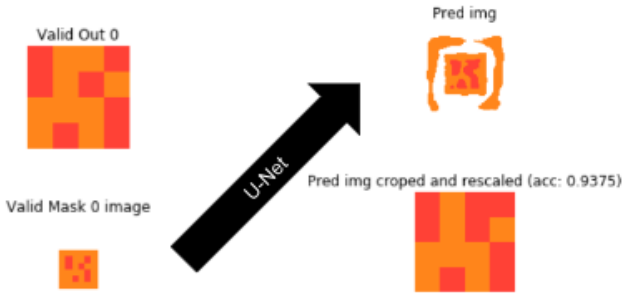

Figure 12: Image training with U-Net example

3 RESULT AND DISCUSSION

Although each model solved some tasks in public data set, only decision tree made a correct prediction for secret evaluation set. Because the metric is whole correctness of the prediction, partially correctness of some integers was not enough and models failed in many tasks. Therefore other models did not make a proper prediction and even decision tree model got 2% correctness score. Most effective model, decision tree would be improved by further feature extracting and augmenting methods. Also, making arbitrary noise is expected to be a regularization method.

3.1 Leaderboard

We were ranked in 9th position. Much of participants did not succeed to make a correct prediction and only few participants made near the 30% of correct predictions.

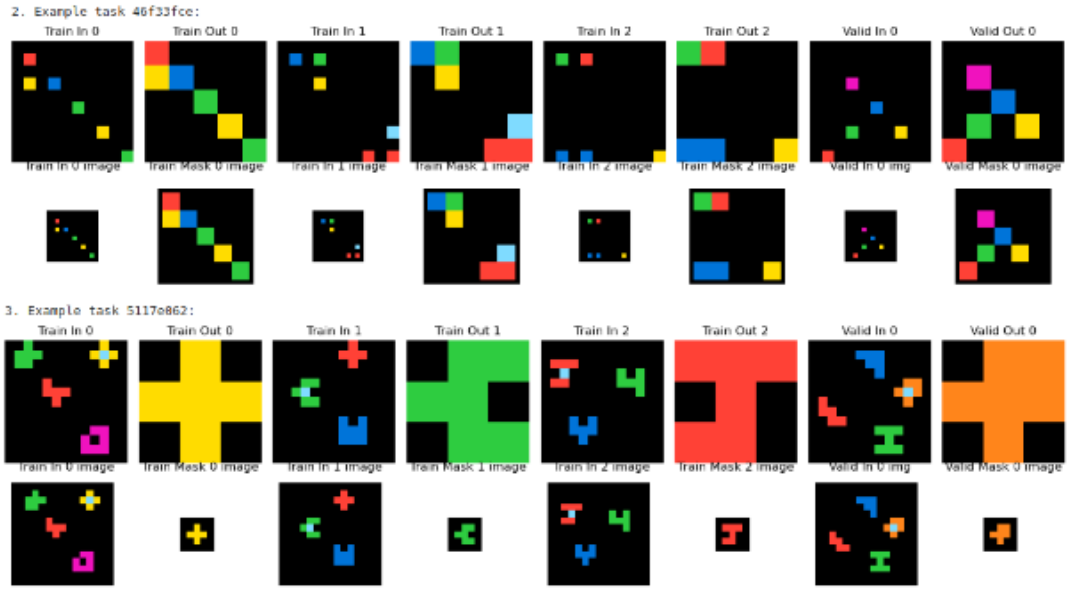

Figure 11: Sample task with padding

#	Team	Country	%	Entries
1	SM	Iran	30%	1
2	MindsAI	USA	30%	12
3	armo	Belgium	29%	27
4	notXORdinary	Denmark	8%	64
5	pablo	Switzerland	5%	1
6	MADIL	France	2%	5
7	AIVAS	USA	2%	5
8	ocaml_is_beautiful	France	2%	10
9	GIST22	South Korea	2%	4
10	ASA	Georgia	1%	6

Figure 13: Leaderboard last updated in December 3, 23:45 p.m. CET. GIST22 is ranked in 9th.

3.2 Discussion

In ARC tasks, metric is only the correctness of whole integers in the test output. Therefore the prediction regarded to be failed if it contains at least one wrong integer or its prediction has a different shape from desired output. Even our most effective model, decision tree achieved low metric score, although its many predictions are partially correct. To improve the further model, more feature augmentation or regularization by adding noise would be required.

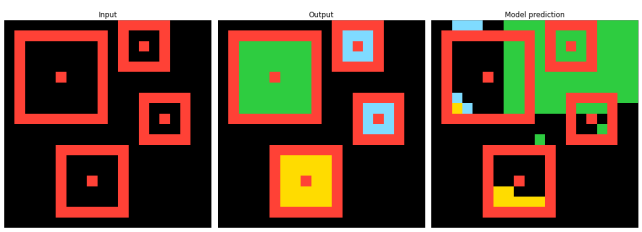

Figure 14: One failed prediction of the decision tree model from augmented data.

Solving tasks one-by-one results low metric only solving specific few tasks and even can not make a prediction for other unclassified tasks. Because model itself can not find the goal of new encountered task, it was failed to adapt goal-directness in many cases.

Decision tree method from augmented data made a much more prediction. Predicted many integers or patterns are partially correct. However, it does not give whole corrected prediction in many cases. Because the model predicts integer from its surroundings, model does not discern objectness directly, although it usually conserves boundary of such objects.

U-Net focuses on catching the object and makes a prediction from it. Therefore although it made a proper results for object selecting or re-scaling task, its solvable tasks are narrow, so that tasks which does not assume objectness would not be predictable.

Below we have listed some of the reasons why our model achieved only 2 percent in solving the hidden task:

- Advanced feature extraction method should be employed
- Only flip rotation might not be help full in capturing the underlying patterns
- Introducing random noise or change to the images will improve performance
- Task specific approach (domain specific knowledge): because each task has specific pattern, developing a model that can solve task specific patterns may show better performance.

To gather more features, features earned from each objects might be re-given to each nonzero pixels again. If we discern whether each pixels are from specific geometrical object, such as line or rectangle, then decision tree model would overcome the ignorance of objectness.

In some tasks, certain color appears on only one or two input-output pairs. This might make AI confused in predicting step. Therefore, augmentation on colors would be effective in some tasks. Because certain color would repeatedly appear and have a important roll in some other tasks, augmentation on colors must be carefully treated.

Because ARC tasks assume human-like intelligence, AI should make a prediction from certain rule or pattern. Small differences non-related to such rule should not be overrepresented and thus making more data including arbitrary noise would be a regularization method.

REFERENCES

[1] Samuel Acquaviva, Yewen Pu, Marta Kryven, Theodoros Sechopoulos, Catherine Wong, Gabrielle E Ecanow, Maxwell Nye, Michael Henry Tessler, and Joshua B. Tenenbaum. 2023. Communicating Natural Programs to Humans and Machines. arXiv:2106.07824 [cs.AI]

[2] James Ainooson, Deepayan Sanyal, Joel P. Michelson, Yuan Yang, and Maithilee Kunda. 2023. A Neurodiversity-Inspired Solver for the Abstraction & Reasoning Corpus (ARC) Using Visual Imagery and Program Synthesis. arXiv:2302.09425 [cs.AI]

[3] Simon Alford, Anshula Gandhi, Akshay Rangamani, Andrzej Banburski, Tony Wang, Sylee Dandekar, John Chin, Tomaso Poggio, and Peter Chin. 2021. Neural-guided, Bidirectional Program Search for Abstraction and Reasoning. arXiv:2110.11536 [cs.AI]

[4] François Chollet. 2019. On the Measure of Intelligence. arXiv:1911.01547 [cs.AI]

[5] Raphael Fischer, Matthias Jakobs, Sascha Mücke, and Katharina Morik. 2020. Solving Abstract Reasoning Tasks with Grammatical Evolution.

[6] Tan John Chong Min. 2023. An Approach to Solving the Abstraction and Reasoning Corpus (ARC) Challenge. arXiv:2306.03553 [cs.AI]

[7] Jaehyun Park, Jaegyun Im, Sanha Hwang, Mintaek Lim, Sabina Ualibekova, Sejin Kim, and Sundong Kim. 2023. Unraveling the ARC Puzzle: Mimicking Human Solutions with Object-Centric Decision Transformer. arXiv:2306.08204 [cs.AI]

[8] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. 2015. U-Net: Convolutional Networks for Biomedical Image Segmentation. In *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, Nassir Navab, Joachim Hornegger, William M. Wells, and Alejandro F. Frangi (Eds.). Springer International Publishing, Cham, 234–241.