Playgrounds for Abstraction and Reasoning

Anonymous Author(s)
Affiliation
Address
email

Abstract

While research on reasoning using large models is in the spotlight, a symbolic method of making a compact model capable of reasoning is also attracting public attention. We introduce the Mini-ARC dataset, a 5x5 compact version of the Abstraction and Reasoning Corpus (ARC) to measure the abductive reasoning capability. The dataset is small but creative, which maintains the difficulty of the original dataset but improves usability for model training. Along with Mini-ARC, we introduce the O2ARC interface, which includes richer features for humans to solve the ARC tasks. By solving Mini-ARC with O2ARC, we collect human trajectories called Mini-ARC traces, which are potentially helpful in developing an AI with reasoning capability.

1 Mini-ARC: 150 Tasks for Measuring Intelligence

The first part of the paper introduces the Mini-ARC dataset, a set of visual reasoning tasks [1] that requires a complete understanding of human priors to solve each task correctly. To make the dataset, we set up certain principles that satisfy this purpose introduced in §1.1.

1.1 Principles

The primary reason to curate the compact 5x5 data set is to reduce the modeling budget. We conjecture that one of the approaches to tackle the ARC task is to train a reinforcement learning agent by following a human trace. However, the size of the original task varies from 1x1 to 30x30, the number of states and action space becomes inevitably large. We want to provide a lightweight training environment by minimizing the number of states and restricting the size of tasks. The reason for choosing the size 5x5 is as follows.

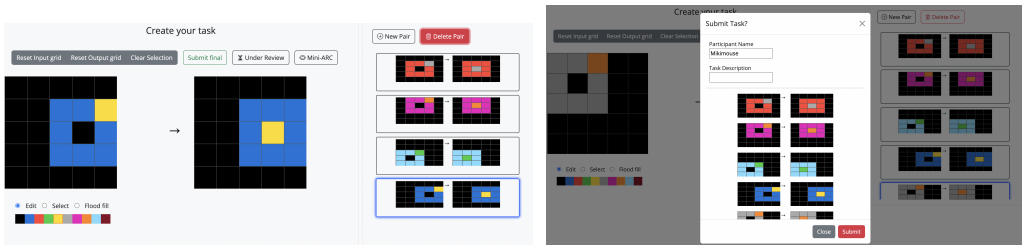
- When width and height are odd, the grid space has its central point.
- Square shape will encourage using basic primitives such as rotation to solve the task.
- 1x1 and 3x3 are too small, so rotation and symmetry were difficult to distinguished.

We invited 25 colleagues with a basic understanding of ARC to generate novel 5x5 tasks, including at least four demonstration pairs (input-output pairs). We instructed participants to build a task with a clear and unique solution to fit the intent of the ARC problem in this abbreviated size. During two day event, they spent two hours solving ARC problems and four hours creating new problems. As a result, we collected about 150 Mini-ARC tasks. Figure 1 illustrates the data collection interface.

As an example of this, various levels of problems can be created within 5x5 with different levels of complexity, such as an easy task consisting only of primary functions such as rotation and flip as the examples in the original research [2], or a difficult task where average description recorded in LARC [3] is lengthy.

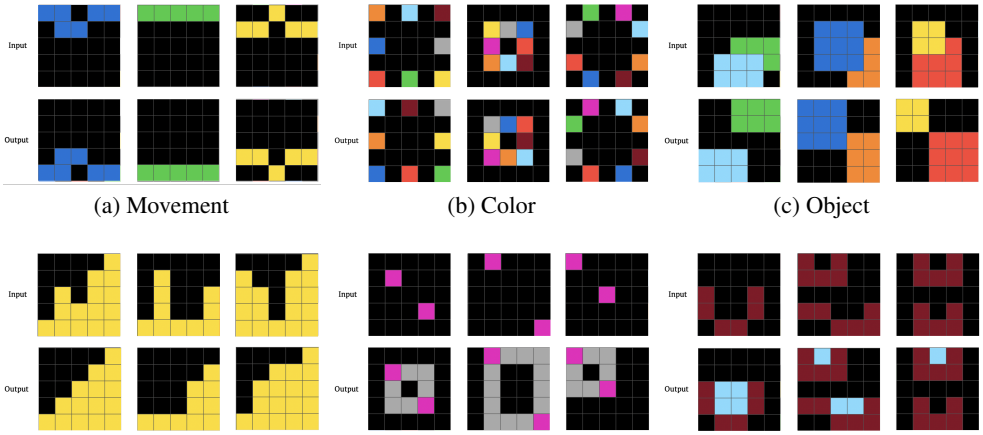
Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

Following the principles of ARC, participants are not encouraged to use prior external knowledge such as language, and the color option for each pixel is the same as the original ARC data set. The instructions were structured to respect the creativity of individual generators as much as possible.



(a) Task generation screen (b) Task submission screen

Figure 1: Interface for curating the Mini-ARC dataset. We allow participants to check other submissions while designing their tasks. Submitted tasks are queued in the under-review list. Administrators verified each task and notified participants about the approval. Resubmitting after editing the already made tasks is allowed. Finally, 150 approved tasks consist of the Mini-ARC dataset.



(a) Movement (b) Color (c) Object

(d) Number (e) Geometry (f) Common sense

Figure 2: Examples for each category: Movement, Object, Number, Geometry, and Common sense

1.2 Categorization

Mini-ARC tasks can be classified into the following six categories: Movement, Object, Number, Geometry, and Common-sense, based on the most prominent concepts for solving each task. Figure 2 shows a set of examples for each category.

Movement Tasks included in this category follow the rules based on dynamic modifications such as flip, rotation, and sliding sideways. The example shown in Figure 2(a) can be solved by flipping the 5x5 grid horizontally.

Color In this category, there are problems that are highly independent of the color aspect of each pixel, such as swapping colors. For example, the rule of Figure 2 is changing to the color of the 1x1 pixel adjacent in a clockwise direction.

Object The movement of the object or agent plays a vital role in solving the problem. An object refers to an area consisting of two or more adjacent pixels that can intuitively distinguish the object from the background.

Number This category includes a set of problems that could be solved using the number concept, such as counting the number of pixels of the same color and sorting the pixels accordingly or modifying the pixels so that the number of pixels could vary.

Geometry This category includes problems that require the concept of geometric structures. The problems involved mainly unifying the color of grids of the same row or column or swapping the same colored areas into other colors.

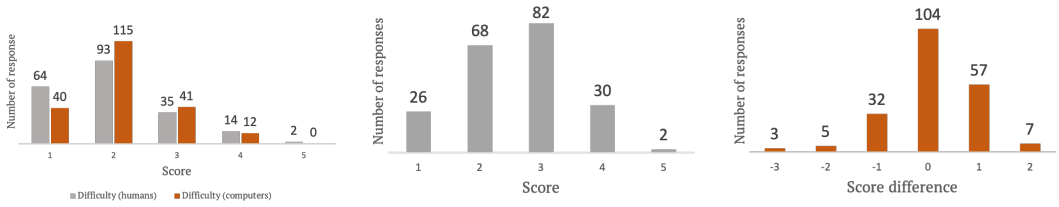
Common-sense This category includes tasks that do not fall into the five categories but are based instead on common sense. For example, The input-output pairs in Figure 2(f) demonstrate vertically dropping rainwater, and the brown-colored grid serves as a container for water. As such, problems that require high-level induction, although it can be intuitively clear for humans, belong to this category. Other issues include maze-pathfinder or Tetris.

1.3 Quality Evaluation

After curating the dataset, participants evaluate each other's dataset and give a score between 1 and 5 to measure task difficulty and novelty. The difficulty is divided into two, one for developing AI models and the other for human solvers. Higher scores mean being more complex (or novel). For convenience, evaluators were asked to evaluate random tasks. As a result, a total of 208 responses were collected and each task received at least one evaluation.

Difficulty As seen in Figure 3(a), Most of the problems were evaluated as easy for humans and computers. In particular, no response was scored as 5 for difficulty for computers. This is likely because the evaluators thought of one program solving only one individual problem at a time. The difficulty of one program solving multiple sets of data consisting of different rules is expected to be evaluated much higher.

Comparison between human and program Figure 3(c) shows the result of subtracting the score of difficulty for humans from the score of difficulty for computers for each evaluated problem set. If the difficulty for computer is greater than the difficulty for humans, it reflects the induction gap between the program and the human.



(a) Difficulty of tasks

(b) Novelty of tasks

(c) Difference between the difficulty for human and AI solvers

Figure 3: Survey results of Mini-ARC

1.4 Challenges and Suggestions

Challenges Mini-ARC limited the size of the input and output to 5x5, which may have restricted the creativity that the generator can exert when creating problems. Therefore, it may be challenging to measure general intelligence accurately using only Mini-ARC. This can be seen as an opportunity cost inevitably borne when space is constrained. After finding a way to solve the current Mini-ARC effectively, it will be possible to expand and switch to a form that does not restrict input and output grid size.

Suggestions There can be many ways to utilize in further research regarding Mini-ARC. First, since each generated Mini-ARC problem set should include at least four input-output pairs, four combinations of demonstration pairs and test pairs can be made for the same problem set in the training process. In addition, by using the six categories as mentioned in sec:category, it is possible to compare the characteristics of each category when developing models.

2 O2ARC: Tools for Collecting Expert Demonstrations

We re-design the browser-based interface for humans to solve the ARC tasks with their hands. We name it Object-Oriented ARC (O2ARC). The tool has six additional functions to ease the manual problem-solving process, and we will describe the principles and procedures hereafter.

2.1 Motivation of Redesigning the Original Tool

The functions included in the tool provided with the original ARC dataset were pretty basic. It contains three symbol controls: edit, select, and flood fill, and three grid controls: copy from input, resize grid, and reset grid to null.[1] All ARC problems can be solved by using these basic functions, but the lack of convenience makes the participant click the button repetitively when solving the problem. Therefore, the human trajectory saved by this tool is long and messy and often does not reflect the intuition of solving the problem at a glance.[2]

2.2 New Features: Six Primitives

First, we brainstorm additional functions grounded that many ARC problems are object-oriented, as illustrated in Figure 2. Inspired by editing tools such as Microsoft Powerpoint and Adobe Photoshop, functions such as cut, undo, redo, flip, rotate, and ctrl+select was implemented and added to the O2ARC tool. Through these functions, the O2ARC provided a higher level of convenience. The details of these functions were improved through alpha testing.

2.3 Layers

The function we concentrated on the most is a layering function. With this function, the user can save each object in one layer and control the object's position by manipulating the layer instead of modifying the final output grid. The user can change the order of the stacked objects easily by changing the order of the layers. This function enables the user to solve some problems concisely.[3] Aside from the convenience of the original ARC, the layer function was not necessarily needed to solve the 5x5 size of Mini-ARC. Therefore, in unraveling the 5x5 size Mini-ARC, we disabled this function and collected Mini-ARC Trace.

2.4 Interface Design

Other than that, we tried to change the layout of the interface by showing an input-output pair one by one instead of showing all training pairs together. This is to collect fine-grained user trajectories to answer some questions: How many input-output pairs do users see before making any action? How many seconds did the user look at each pair? Answering these questions helps us measure the task's difficulty or the wit and agility of the participants. Contrary to the intention, some participants complained that this layout takes away the advantage of observing pairs altogether. Therefore, we restored the original design.

3 Mini-ARC Trace: Compiled Expert Demonstrations

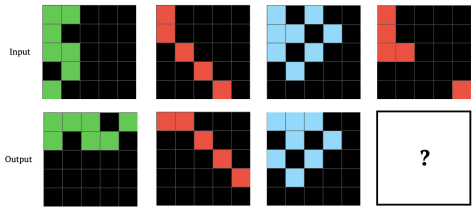
3.1 Principles of Collecting Mini-ARC Trace

Tasks used for gathering Mini-ARC trace were picked uniformly from six categories. We gathered 20 participants to solve ten sets of Mini-ARC within two hours. Each set contains five tasks, one from each category. We tried to exclude the 'common-sense' category as much as possible because the solutions for this category would seem difficult even for human participants. If the participant submits three consecutive wrong answers, the system determines that the user cannot not find the solution to the problem.

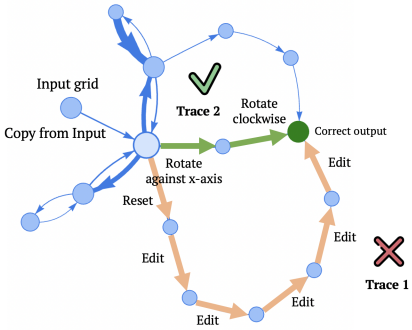
[1] Description of these functions is on the ARC repository: `https://github.com/fchollet/ARC`.

[2] See complex state-space graphs on `https://arc-visualizations.github.io/`.

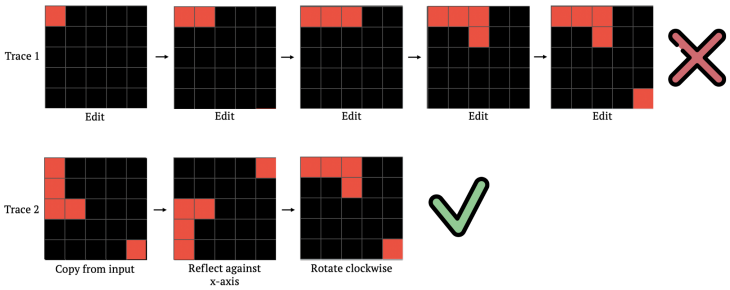
[3] See demonstrations on `https://youtu.be/zo1G4E720Co?t=108`.



(a) Task: Diagonal flip



(b) State-space graph



(c) Two human traces: One uses only edit button, the other uses flip and rotate.

Figure 4: Different traces are logged for the same diagonal flip problem. The first solution using only the edit button is not generalizable to other training pairs, which is not the desired answer. The logic behind the second solution is a diagonal flip, which is the correct answer. Selecting meaningful traces and training a model based on them will be the next challenge.

For each Mini-ARC task, we combined all traces to form a state-space graph. Figure 4(b) demonstrates the visualization of multiple traces collected from different participants. A sequence of actions can be identified from each trace. Each node represents the state of the output grid, and the green node represents the correct output grid.

3.2 Challenges and Suggestions

Challenges Figure 4(a) is a simple diagonal flip task, and we highlight two representative traces in Figure 4(b). As seen in Figure 4(b)–(c), not all traces contain the high-level process of human intuition. For example, in Trace 1, the task is solved through naive action sequences using only edit. The restricted input and output grid size of Mini-ARC may take part in increasing naive actions. However, Trace 2 reflects the intuition of diagonal flips using compact movements, and the process for solving the task is also more effective. The question of how to collect traces that mirror intuition remains

Suggestions The purpose of Mini-ARC trace is to collect reference trajectories for reference for the programs. We can utilize the traces as a replay buffer for training agents through imitation learning. We also plan to analyze the traces based on action sequences to find the new consequent series of actions that can be generally used for solving multiple Mini-ARC.

4 Conclusion

We present the Mini-ARC produced as part of challenging abductive reasoning problem [2, 4], and introduce O2ARC, which enables including high-level features of human intuitions when solving ARC tasks. With O2ARC, we collect Mini-ARC traces that can be used to advance the program synthesis [5] and RL agents [6, 7] to move one step toward general intelligence.

References

[1] Mikołaj Małkiński and Jacek Mańdziuk. Deep learning methods for abstract visual reasoning: A survey on raven's progressive matrices. *arXiv preprint arXiv:2201.12382*, 2022.

[2] François Chollet. On the measure of intelligence. *arXiv:1911.01547*, 2019.

[3] Samuel Acquaviva, Yewen Pu, Marta Kryven, Theodoros Sechopoulos, Catherine Wong, Gabrielle E Ecanow, Maxwell Nye, Michael Henry Tessler, and Joshua B. Tenenbaum. Communicating natural programs to humans and machines. *arXiv:2106.07824*, 2021.

[4] Aysja Johnson, Wai Keen Vong, Brenden M. Lake, and Todd M. Gureckis. Fast and flexible: Human program induction in abstract reasoning tasks. In *CogSci*, 2021.

[5] Kevin Ellis, Catherine Wong, Maxwell Nye, Mathias Sablé-Meyer, Lucas Morales, Luke Hewitt, Luc Cary, Armando Solar-Lezama, and Joshua B. Tenenbaum. Dreamcoder: Bootstrapping inductive program synthesis with wake-sleep library learning. In *PLDI*, pages 835–850, 2021.

[6] Simon Alford, Anshula Gandhi, Akshay Rangamani, Andrzej Banburski, Tony Wang, Sylee Dandekar, John Chin, Tomaso Poggio, and Peter Chin. Neural-guided, bidirectional program search for abstraction and reasoning. In *International Conference on Complex Networks and Their Applications*, pages 657–668. Springer, 2021.

[7] Lili Chen, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Misha Laskin, Pieter Abbeel, Aravind Srinivas, and Igor Mordatch. Decision transformer: Reinforcement learning via sequence modeling. In *NeurIPS*, 2021.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes] See Abstract.
   (b) Did you describe the limitations of your work? [Yes] See Challenges and Suggestions.
   (c) Did you discuss any potential negative societal impacts of your work? [N/A]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes] Yes, I read it and this paper conforms to them.
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? **[TODO]** The benchmark data and tool can be released upon the acceptance of the paper.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [N/A]
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [N/A]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [N/A]
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] Yes, we cited [2].
   (b) Did you mention the license of the assets? **[TODO]** In our Github Repository, which is currently hidden.
   (c) Did you include any new assets either in the supplemental material or as a URL? **[TODO]** Later, we will include our Github as a URL.
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [Yes] Yes, we got consent from participants.

(e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [Yes] Yes, we discussed, we can distinguish each trace, but cannot identify person.

5. If you used crowdsourcing or conducted research with human subjects...

(a) Did you include the full text of instructions given to participants and screenshots, if applicable? [Yes] It will be given with our codes.

(b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? **[TODO]** We will describe it on the camera-ready version.

(c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A] Recruited our lab members by serving nice lunch/dinner.