January ARC Project Report

Group: Lost in Program Space
Dennis Lindberg, Kerem Yoner, Paul Weston, Yipu Li
Supervisor: Dr. Fausto Carcassi

Master students of Logic at the ILLC, Amsterdam

February 2, 2025

1 Introduction

The Abstract and Reasoning Corpus for Artificial General Intelligence (ARC-AGI), introduced by Chollet [1], is a dataset designed to evaluate progress towards Artificial General Intelligence (AGI). The tasks in ARC-AGI are structured as small grid-based puzzles requiring reasoning and pattern recognition, where the contestant must infer transformation rules from a few input-output examples.

The 2024 ARC Prize technical report, published by the ARC-AGI team [2], highlights three leading methodologies. The first, *deep learning-guided program synthesis*, employs deep learning architectures, such as large language models (LLMs), to generate executable programs that compute outputs given an input, and debug the programs that are promising [3]. The second approach, *transductive models with test-time training*, involves deep learning models that directly generate output grids based on input. The test-time training refers to fine-tuning the model on the training examples in the test dataset before it predicts the output. The third approach is a hybrid of the first two, leveraging their complementary strengths. Research suggests that inductive and transductive models excel at different task types, making their combination a promising direction [4].

Our approach aligns well with these observations, building on a transductive model with test-time training and attempting to combine this with deep-learning guided program induction.

2 Methodology

We employ the framework built by the first place winning team of the 2024 ARC competition. The ARChitects [5] makes use of fine-tuned LLMs, test-time training, and a novel method to generate and select candidate solutions. We add an auxilliary induction model to provide a first attempt on tasks and then further refine the transduction model's candidate selection.

Induction and transduction combined: In the literature revolving around ARC-AGI, induction methods predict the program on a domain general language or domain specific language and apply it to the problem; while transduction methods directly predict the output instead of the program. As discussed in the paper [4], induction and transduction method solves different types of problems in the ARC-AGI benchmark. In our model, we apply the induction model given by [4] to complement the main model, which focuses on transduction. Inspired by [3], our induction model does an additional debugging round on the promising programs to maximize its ability.

Candidate Selection: In the ARChitects model [5], many candidate solutions are generated and a solution is chosen that the model is most confident in across various data augmentations applied to the task. Across their models, the correct solution is generated much more than it is chosen. We explored alternative methods for candidate selection to capitalize on this discrepancy. We investigated using summation rather than multiplication to mitigate the negative effects of poorly performing data augmentations. Additionally, we applied methods that would limit the candidate selection to data augmentations the model was most confident on for a specific task.

3 Results

For the final result we ran the model on the first 100 tasks in the evaluation tasks, and here is the result:

	Tasks solved	**Applied on**	**Accuracy**
Induction model	1	9	88.9%
Transduction model	49.5	91	54.4 %
Combined model	56.5	100	56.5%

The high accuracy of the induction model is due to the fact that we are only using its submission when the program is positively verified on the training sets of a single task token. The result shows that for 2 out of 9 programs submitted by the induction model, the program produces the wrong output even though it fits the training examples. One of the error of the induction model actually roots in an error from our side. The induction model produces the correct code but we failed to execute it correctly.

We noticed that the transduction model incorrectly predicts the output grid size 5-7% of the time for the first attempt, and closer to 30% for the second attempt. Moreover, there were cases in which the first part of an output grid was correct, but that the array contained more cells, causing the submission to count as incorrect. This is a major drawback of transduction methods in general, as there is no obvious way to verify the correctness of a solution. We propose a method to mitigate this in the next section.

4 Discussion and Future Work

Better Integrating the Induction and Transduction Pipeline. One potential improvement is to have the induction model determine the expected output grid size. This information can then be passed to the transduction model, thereby reducing the search space and reducing the number of submissions with an incorrect grid size.

Architectures that Improve the Induction Result. It may be beneficial to sample more programs for problems where the induction model is close to solving the task, while abandoning efforts on tasks where the model consistently performs poorly. A challenge in this approach is devising a metric that accurately reflects how close a program is to solving the task.
Another strategy is to fine-tune a dedicated model to debug the promising programs generated during the induction phase.

References

[1] F. Chollet, “On the Measure of Intelligence,” Nov. 2019. arXiv:1911.01547 [cs].

[2] F. Chollet, M. Knoop, G. Kamradt, and B. Landers, “ARC Prize 2024: Technical Report,” Jan. 2025. arXiv:2412.04604 [cs].

[3] R. Greenblatt, “Getting 50% (sota) on arc-agi with gpt-4o,” 2024.

[4] W.-D. Li, K. Hu, C. Larsen, Y. Wu, S. Alford, C. Woo, S. M. Dunn, H. Tang, M. Naim, D. Nguyen, W.-L. Zheng, Z. Tavares, Y. Pu, and K. Ellis, “Combining Induction and Transduction for Abstract Reasoning,” Dec. 2024. arXiv:2411.02272 [cs].

[5] D. Franzen, J. Disselhoff, and D. Hartmann, “The LLM ARChitect: Solving ARC-AGI Is A Matter of Perspective,”