Foundation Model Frontiers

CIS, LMU, SoSe 2025

Maximilian Seeth

Preprint.

Combining Induction and Transduction for Abstract Reasoning

Wen-Ding Li[*1] **Keya Hu**[*2] **Carter Larsen**[1] **Yuqing Wu**[1] **Simon Alford**[1] **Caleb Woo**[1]
Spencer M. Dunn[1] **Hao Tang**[1] **Michelangelo Naim**[3] **Dat Nguyen**[3] **Wei-Long Zheng**[2]
Zenna Tavares[†3] **Yewen Pu**[†4] **Kevin Ellis**[†1]
[1]Cornell [2]Shanghai Jiao Tong University [3]Basis [4]Autodesk [*]co-leads [†]co-advising
correspondence: {wl678,kellis}@cornell.edu, hu_keya@sjtu.edu.cn

Abstract

When learning an input-output mapping from very few examples, is it better to first infer a latent function that explains the examples, or is it better to directly predict new test outputs, e.g. using a neural network? We study this question on ARC by training neural models for *induction* (inferring latent functions) and *transduction* (directly predicting the test output for a given test input). We train

Overview

1.) The challenge
2.) The approach
3.) Results
4.) Outlook

The challenge

The challenge: The right benchmark

- “Testing for a skill that is known in advance to system developers [...] can be gamed without displaying intelligence in two ways: (1) unlimited prior knowledge (2) unlimited training data” (Chollet 2019, p. 20).

- Abstract Reasoning Corpus (ARC) is a benchmark to measure “human-like general intelligence” (Chollet 2019, p. 45).

The challenge: Intelligence

- What should such a benchmark measure to capture human-like intelligence? What family of tasks would it address?

The challenge: Intelligence

- François Chollet believes that humans are born with “cognitive priors”, which are used to deal with (novel) tasks.

The challenge: Intelligence

- François Chollet believes that humans are born with “cognitive priors”, which are used to deal with (novel) tasks.
- An intelligence benchmark should capture these priors (Cholet 2019, pp. 48-50):

a) Object priors:
    Object cohesion (e.g., color and space continuity)
    Object persistence (e.g., despite noise)
    Object influence via contact
b) Goal-directedness prior (similar to time)
c) Numbers and counting priors
d) Basic geometry and topology priors (e.g., symmetries, being inside or outside..)

The challenge: ARC-AGI-1

ARC benchmark addressing human priors (for comparison):

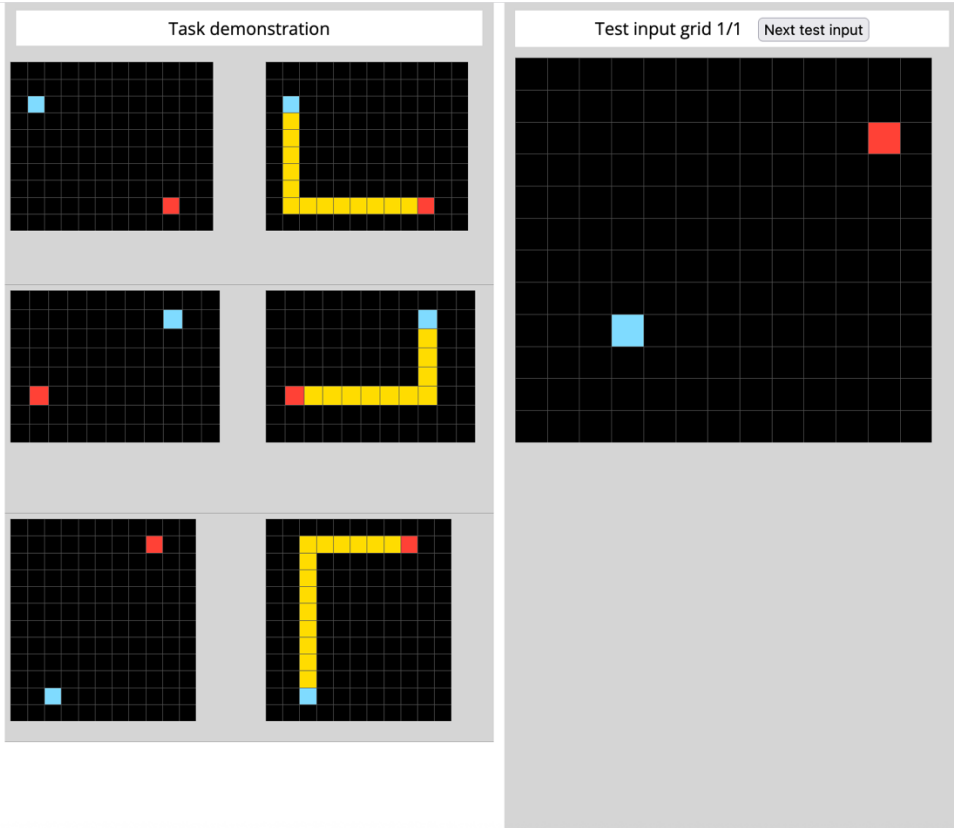
- The “intelligence” tests should be easy to solve for humans.
- The benchmark should control for knowledge about tasks, e.g., by tests unknown to the developers (private holdout set).
- The benchmark should be limited in terms of data (e.g., 600 training, 400 evaluation instances) and compute (12 hours of Kaggle notebook with GPU access).

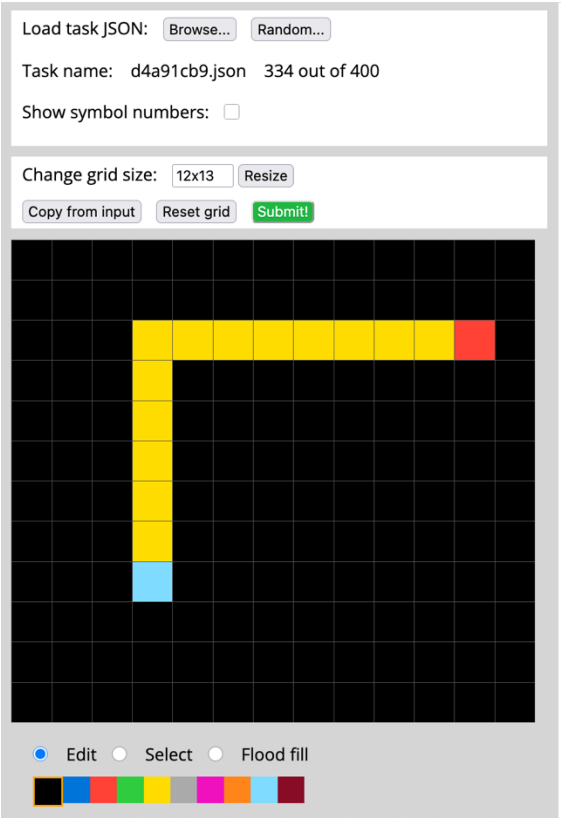
The challenge: ARC-AGI-1

examples

task

solution

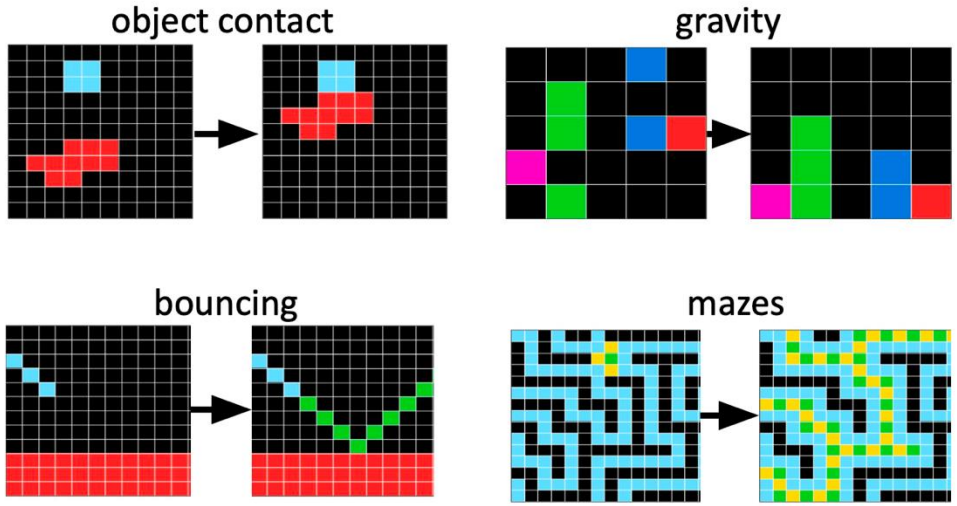




- Few-shot learning benchmark

The challenge: ARC-AGI-1

- Tasks are provided with numerical representations.
- Each task allows 3 attempts.



(Li et al. 2024, p. 1)


```
{"train": [{"input": [[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0], [0, 8, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0]], "output": [[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0], [0, 8, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 4, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0], [0, 4, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 4, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0], [0, 4, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 4, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0], [0, 4, 4, 4, 4, 4, 4, 4, 2, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0]]}, {"input": [[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 8,
0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]], "output": [[0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 8, 0, 0], [0, 0, 0, 0, 0,
0, 0, 0, 4, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 4, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0,
4, 0, 0], [0, 2, 4, 4, 4, 4, 4, 4, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]]}, {"input": [[0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0], [0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0,
0], [0, 0, 8, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]],
"output": [[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 4, 4, 4, 4, 4, 2, 0,
0], [0, 0, 4, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 4, 0, 0, 0, 0, 0, 0, 0, 0], [0,
0, 4, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 4, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 4, 0,
0, 0, 0, 0, 0, 0, 0], [0, 0, 4, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 4, 0, 0, 0,
0, 0, 0, 0, 0], [0, 0, 4, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 8, 0, 0, 0, 0, 0,
0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]]}], "test": [{"input": [[0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0,
0, 0, 0, 0, 0, 2, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0,
0, 0, 8, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0]], "output": [[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0], [0, 0, 0, 4, 4, 4, 4, 4, 4, 4, 2, 0], [0, 0, 0, 4, 0, 0,
0, 0, 0, 0, 0, 0], [0, 0, 0, 4, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 4, 0,
0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 4, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 4,
0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 8, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]]}]}
```

The approach

The approach

- Li et al. (2024) propose a combination of two models to solve ARC tasks:

  Model 1 performing **inductive program synthesis**.
  Model 2 performing **transductive prediction**.

- These two models are ensembled to make predictions on the test set.

The approach: Program synthesis

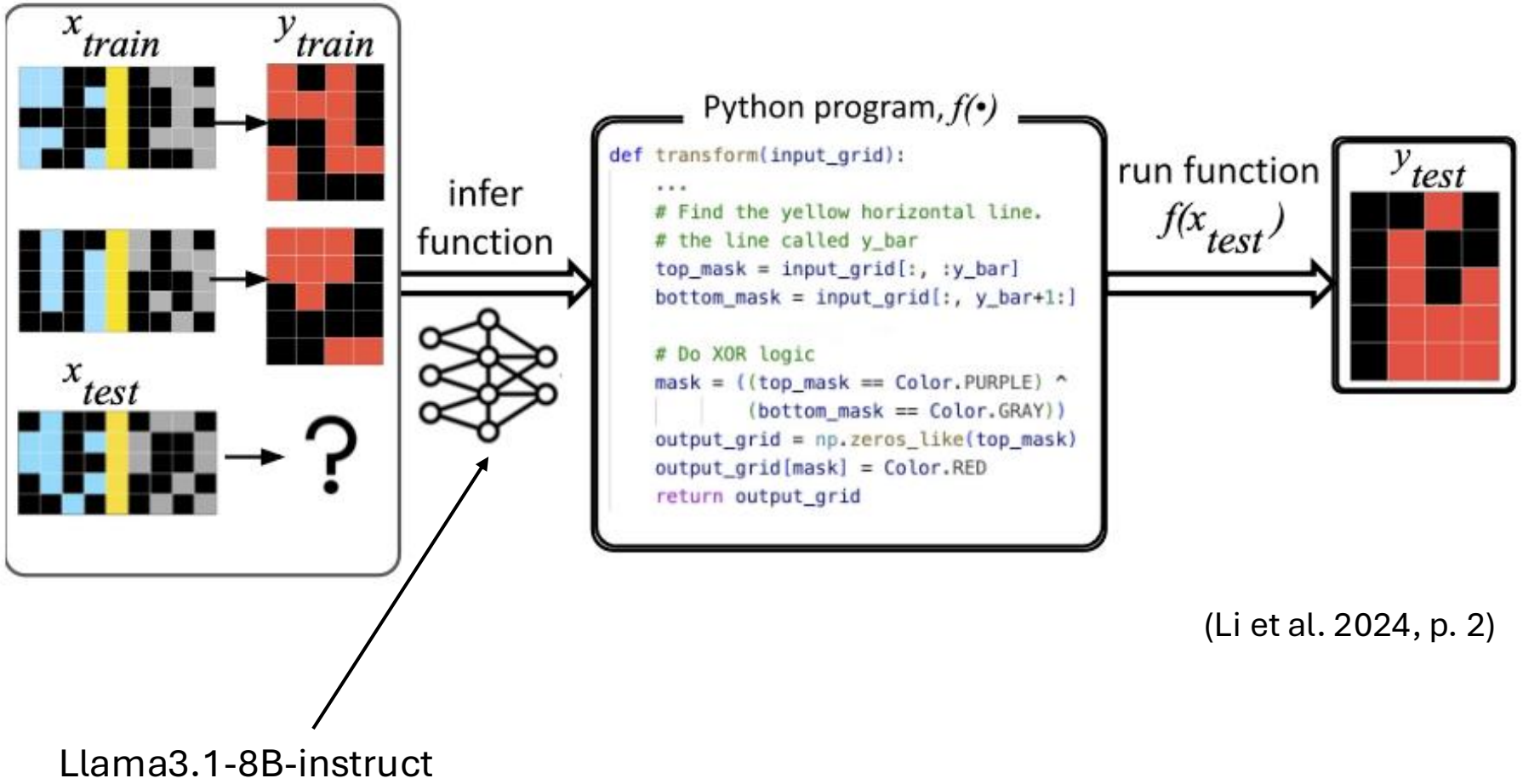
- From Devlin et al. (2017), auto-fill for Microsoft Excel:

Input	Output
I_1= January	O_1 = jan
I_2 = February	O_2 = feb
I_3 = March	O_3 = mar
I_n	O_n

P = ToCase(Lower, SubStr(1,3))

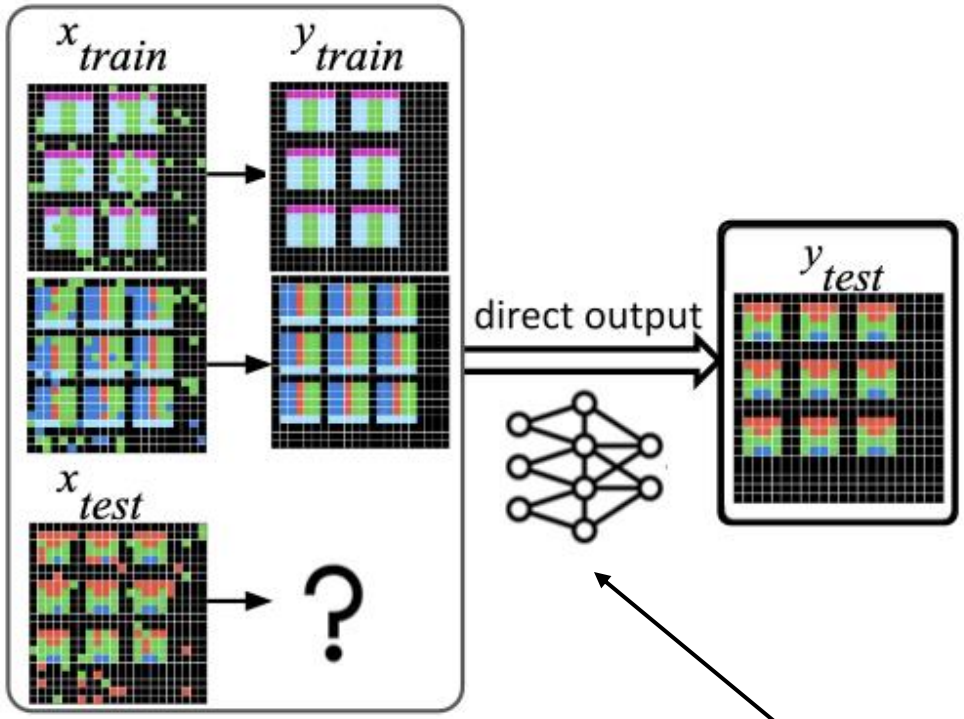
- "In the program synthesis approach, we train a neural model which takes (I_1, O_1), …, (I_n, O_n) as input and generates P as output, token-by-token." (Devlin et al. 2017, p. 3)

The approach: Induction



(Li et al. 2024, p. 2)

The approach: Transduction



(Li et al. 2024, p. 2)

Llama3.1-8B-instruct

The tasks are translated into lingual representations.

```
Example 2
Input:
Gray Black Black
Black Gray Black
Black Black Gray

Output:
Blue Red Black Black Black Black
Red Blue Black Black Black Black
Black Black Blue Red Black Black
Black Black Red Blue Black Black
Black Black Black Black Blue Red
Black Black Black Black Red Blue
```

(Li et al. 2024, p. 34)

The approach: Prediction

The two models are ensembled, to make predictions on ARC test sets:

1st step:

$$\textsc{Induction}: \hat{y}_{\text{test}} \sim \text{Uniform}\left(\mathcal{F}\right)$$
$$\text{where } \mathcal{F} = \{f_b(x_{\text{test}}) \ : \ \text{for } 1 \leq b \leq B \text{ if } f_b(\boldsymbol{x}_{\text{train}}) = \boldsymbol{y}_{\text{train}}\}$$
$$f_b \sim \mathtt{i}_\theta(f|\boldsymbol{x}_{\text{train}}, \boldsymbol{y}_{\text{train}}, x_{\text{test}})$$

2nd step if $F = \emptyset$:

$$\textsc{Transduction}: \hat{y}_{\text{test}} = \arg\max_{y \in \mathcal{Y}} \mathtt{t}_\theta(y|\boldsymbol{x}_{\text{train}}, \boldsymbol{y}_{\text{train}}, x_{\text{test}})$$

The approach: Prediction

The two models are ensembled, to make predictions on ARC test sets:

*P*rograms = *F*unctions

Test-time budget of B functions (e.g., 10k samples)

1st step:

$$\textsc{Induction}: \hat{y}_{\text{test}} \sim \text{Uniform}\left(\mathcal{F}\right)$$

$$\text{where } \mathcal{F} = \{f_b(x_{\text{test}}) \ : \ \text{for } 1 \leq b \leq B \text{ if } f_b(\boldsymbol{x}_{\text{train}}) = \boldsymbol{y}_{\text{train}}\}$$

$$f_b \sim \mathtt{i}_\theta(f|\boldsymbol{x}_{\text{train}}, \boldsymbol{y}_{\text{train}}, x_{\text{test}})$$

If function works on few shot examples, it can be used for the test.

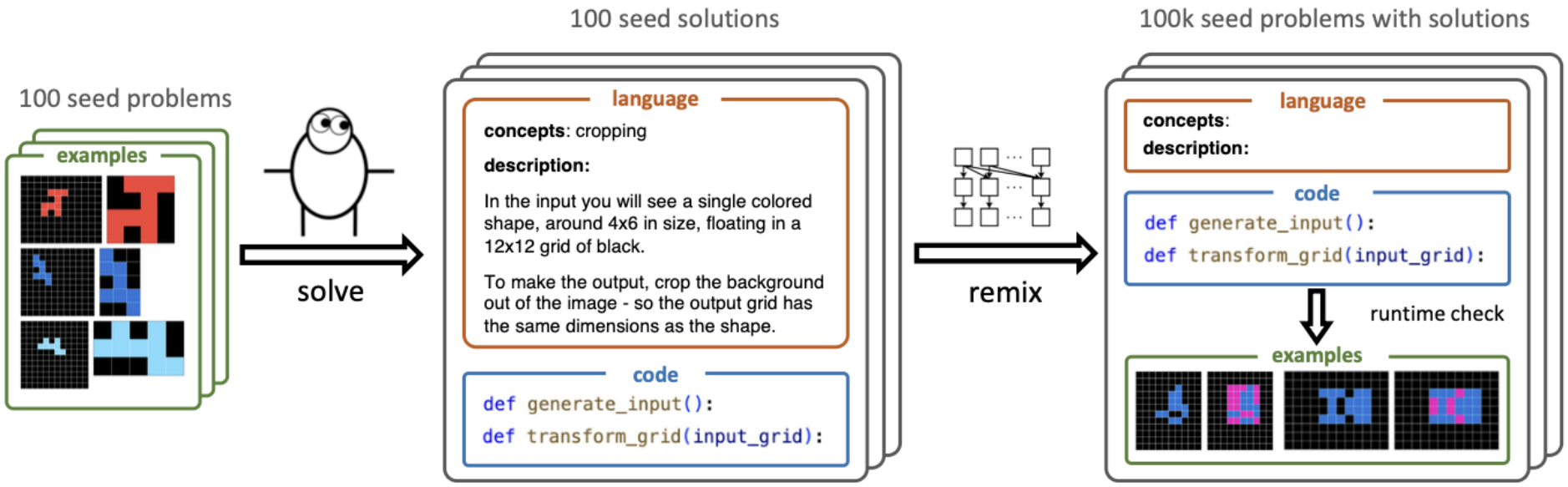
2nd step if $\boldsymbol{F} = \emptyset$:

$$\textsc{Transduction}: \hat{y}_{\text{test}} = \arg\max_{y \in \mathcal{Y}} \mathtt{t}_\theta(y|\boldsymbol{x}_{\text{train}}, \boldsymbol{y}_{\text{train}}, x_{\text{test}})$$

The approach: Synthetic data

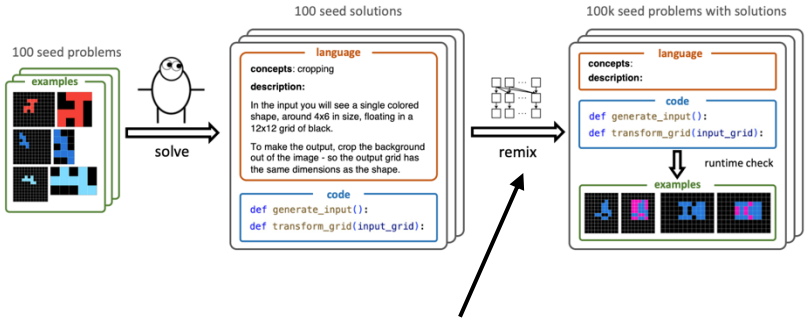
- To train their models, Li et al.(2024) need data!
- They generate a synthetic dataset, starting with 100 manually written solutions (seeds) for 100 ARC problems.

The approach: Synthetic data



(Li et al. 2024, p. 4)

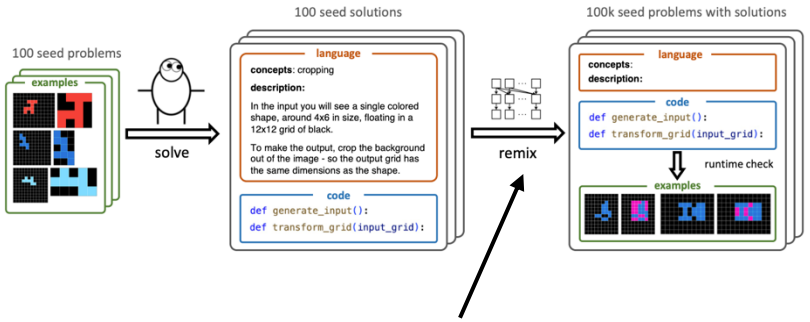
The approach: Synthetic data



New seeds are then generated by:

- prompting an LLM (GPT-4) with seed natural language description to create a new seed from in-context learning;
- retrieving with RAG similar descriptions from existing seeds to generate programs for new description (with GPT-4o-mini);
- generating new inputs for the function, producing new input-output samples.

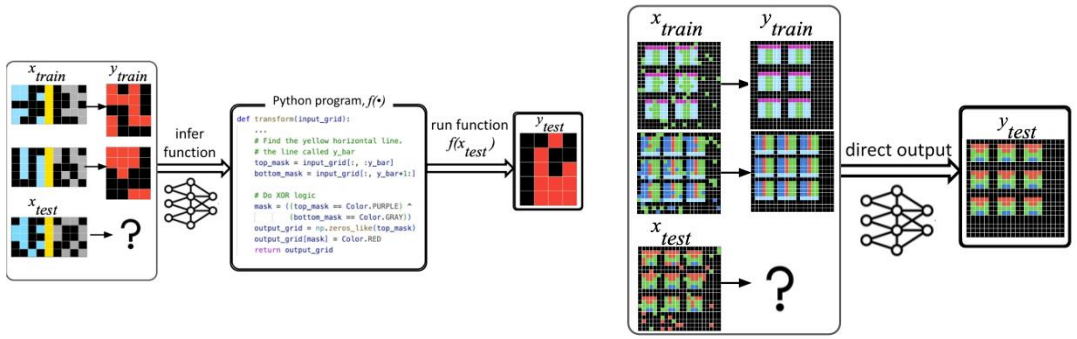
The approach: Synthetic data



Further scaling:

- Li et al. (2024) eventually synthesize 200k examples from 160 seeds.
- Li et al. (2024) add further synthesized input-output samples from other papers on ARC (adding up to 400k).

The approach: Training

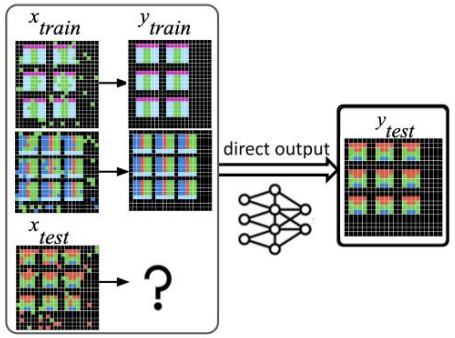


The models are trained based on the synthetic data with meta-learning minimizing few-shot problems:

$$\textsc{Transduction Loss} = \mathbb{E}_{(\boldsymbol{x}_{\text{train}}, \boldsymbol{y}_{\text{train}}, x_{\text{test}}, y_{\text{test}}, f) \sim \mathcal{D}} \left[-\log \mathtt{t}_\theta \left(y_{\text{test}} \middle| \boldsymbol{x}_{\text{train}}, \boldsymbol{y}_{\text{train}}, x_{\text{test}} \right) \right]$$

$$\textsc{Induction Loss} = \mathbb{E}_{(\boldsymbol{x}_{\text{train}}, \boldsymbol{y}_{\text{train}}, x_{\text{test}}, y_{\text{test}}, f) \sim \mathcal{D}} \left[-\log \mathtt{i}_\theta \left(f \middle| \boldsymbol{x}_{\text{train}}, \boldsymbol{y}_{\text{train}}, x_{\text{test}} \right) \right]$$

The approach: Training



- The transduction model is additionally trained with Test Time Training (TTT) and reranking.
  - TTT: self-supervised input transformation (rotation, permuting colors); essentially a second model-head that is added to the existing transductive model (Sun et al. 2020).
  - Reranking: predictions from multiple augmented (grid transposed, color permutated) training samples as further feedback signal to the model.

Results

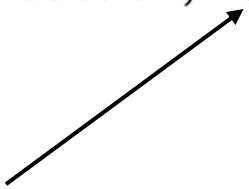
Results: Performance

The accuracy of humans, solving ARC-challenges is:

Average:	60.2%
Best:	97.8%

Li et al. (2024) models after 200k training samples:

Induction, 10k samples, majority vote	30.50%
Transduction (no TTT)	19.25%
Ensemble (no TTT)	37.50%
Transduction (TTT)	29.75%
Ensemble (TTT)	43.25%



Budget (B) of maximally 10k generated functions, pick most frequent (majority) solution.

Results: Performance

Li et al. (2024) models after 400k training samples:

Induction, 20k samples, majority vote	38.00%	
Transduction (no TTT, no reranking)	29.125%	
Transduction (reranking, no TTT)	35.25%	
Transduction (TTT, no reranking)	39.25%	
Transduction (TTT + reranking)	43.00%	
Ensemble (TTT + reranking)	56.75%	≈ average human (60.2%)

However, the ARC challenge comes with limited compute…

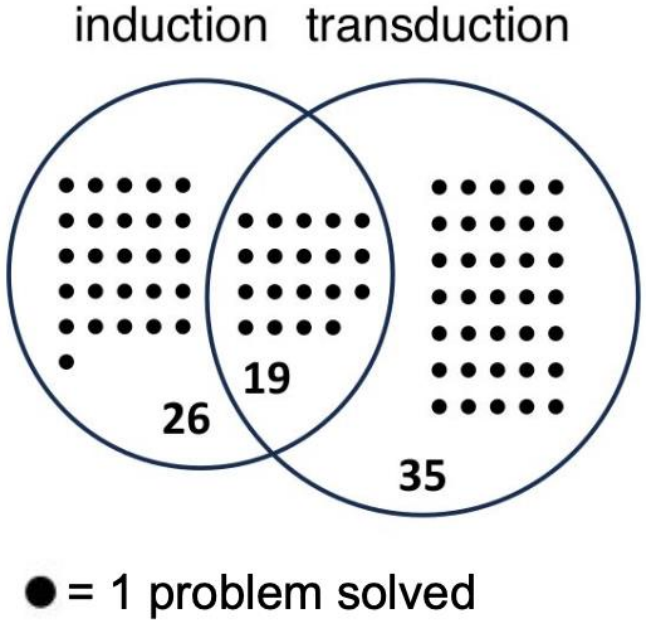
Results: Performance

Li et al.'s (2024) model performance with limited compute (12 hours of Kaggle notebook with GPU access):

	Private Test Set	Public Validation Set
Transduction (no TTT, beam size 3)	18%	32.25%
Induction, 384 samples	4%	14%
Ensemble	19%	36.5%

Results: Observations

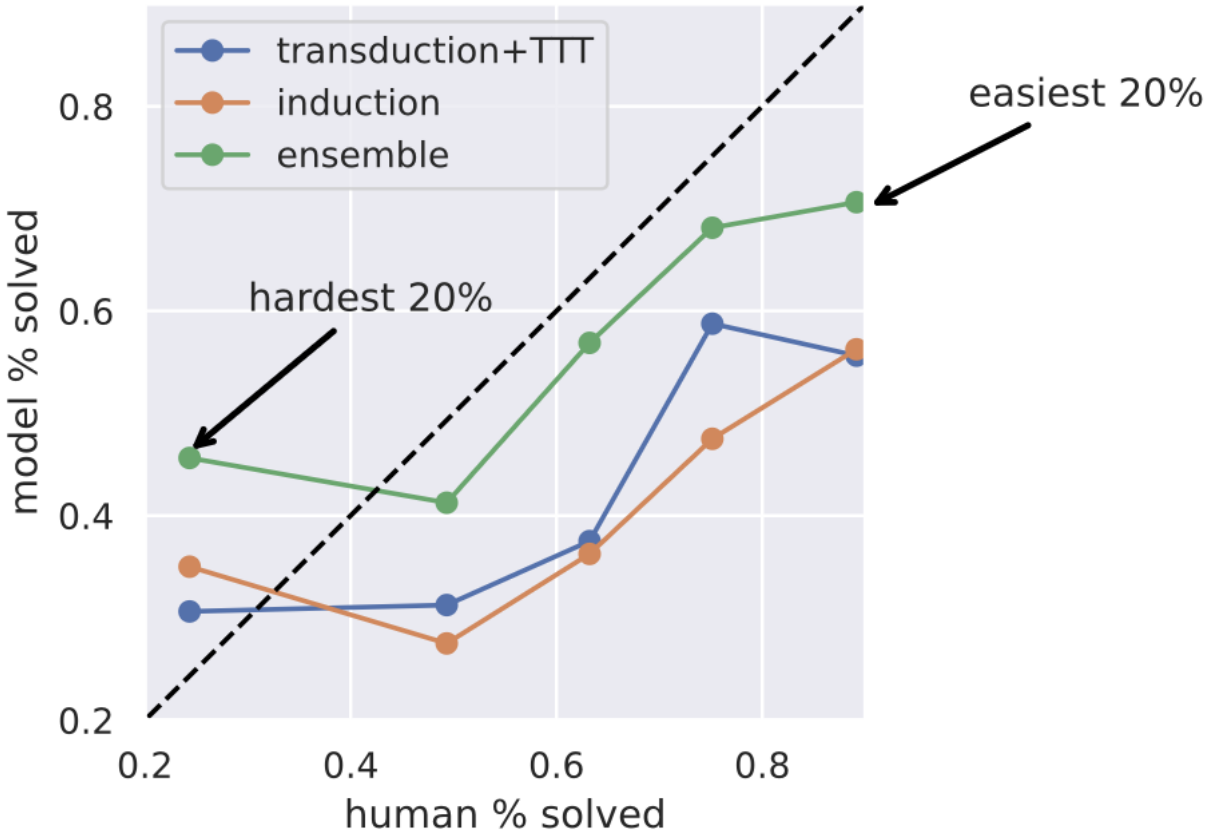
- Induction and transduction work complementary, even after further controlled experiments (e.g., different results due to different model initializations).



(Li et al. 2024, p. 5)

Results: Observations

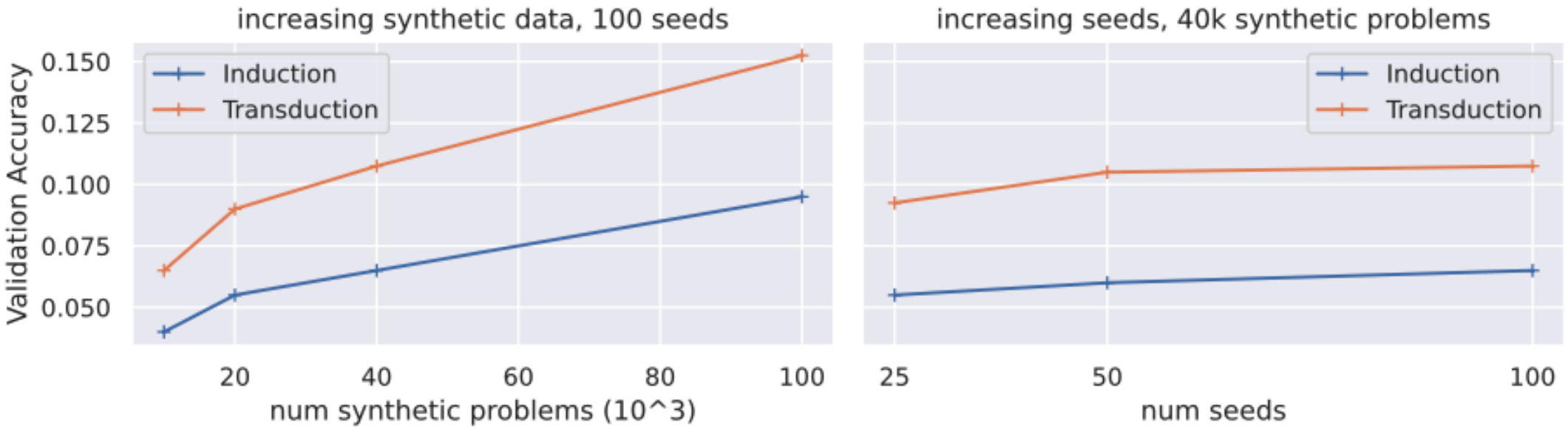
- Models surpass humans on harder problems, but struggle with “easy” ones:



(Li et al. 2024, p. 9)

Results: Observations

- More (>=160) seeds did not improve the results, but performance scales with synthetic data.



(Li et al. 2024, p. 6)

Outlook

Outlook: Advancements

- Li et al. (2024) combine neural with symbolic problem-solving strategies since the generated Python programs are deterministic and rule-based.

- Cognitive background: fast nonverbal intuitions vs. deliberative conscious thought (e.g., Kahnemann (2011)).

Outlook: Advancements

- Program synthesis with Python code instead of domain specific languages (DSL) as in Devlin et al. (2017) is introduced.

- Li et al. (2024) propose domain specific libraries that should be used for program synthesis.

Outlook: Critique

Surprising complementary results:

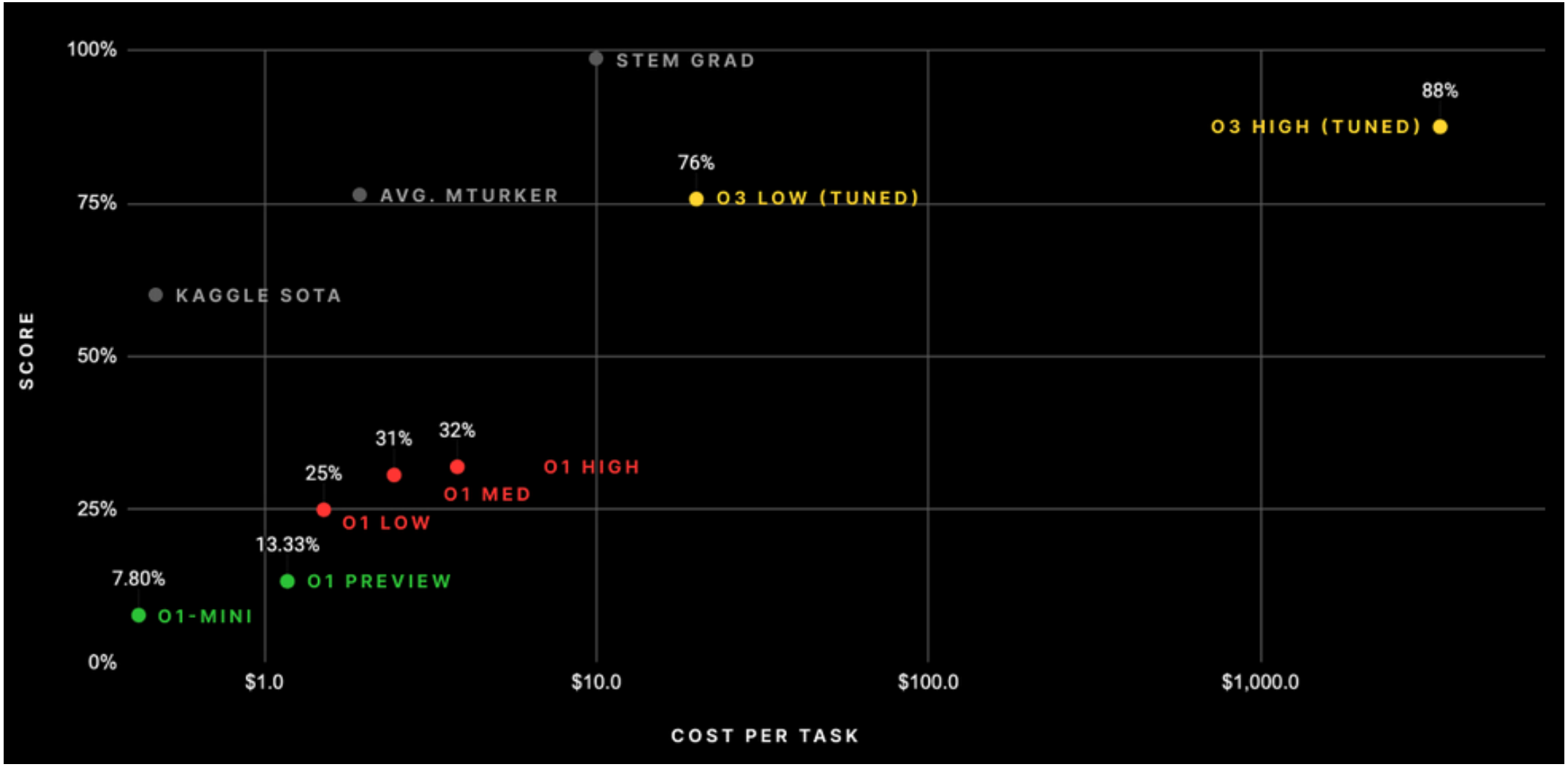
- Python (output from induction model) and the Transformer model (transduction model) are theoretically universal function approximators.

Outlook: Critique

- Li et al. (2024) have no formulated hypothesis of what a model should be like to succeed in ARC-like challenges.

- It seems that they ensemble a variety of techniques and models and still generate a lot of training data to achieve high performance.

Outlook: Unlimited compute



OpenAI O3 performance on the ARC challenge (arcprize.org/blog/oai-o3-pub-breakthrough)

Thank you

Supplementary material

- Critique of ARC: Given the few shot examples, is induction (in the classical sense) a good measure for intelligence?

- What about deductive thinking?

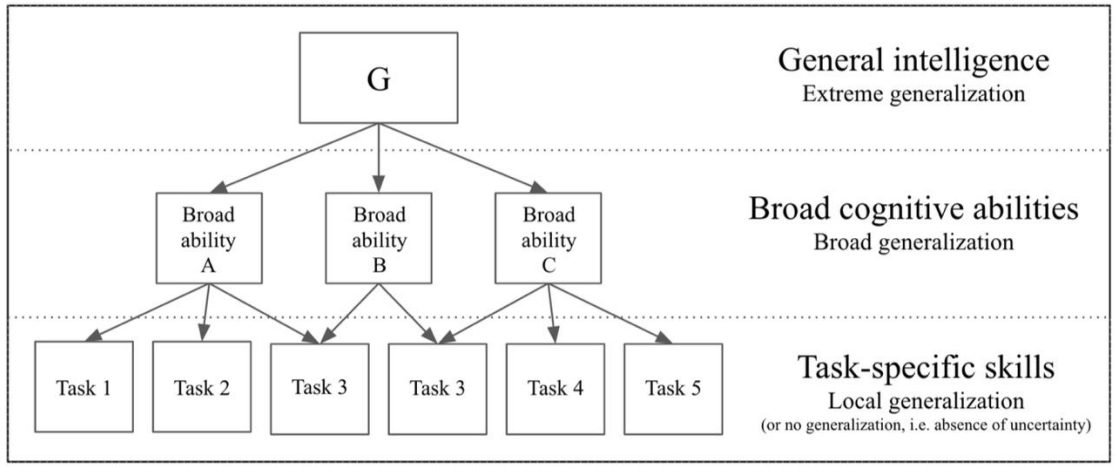
Supplementary material

- What about other models (VLMs or LRMs)?

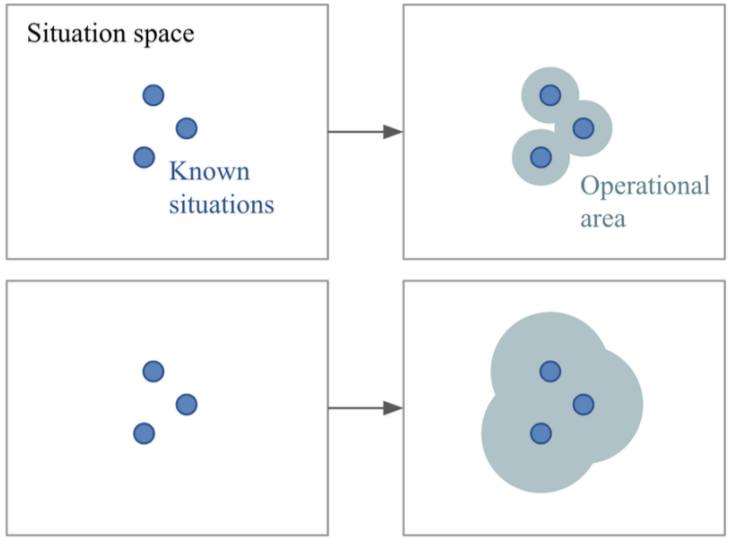
Supplementary material

- Intelligence according to Chollet:

  The intelligence of a system is a measure of its ***skill-acquisition efficiency*** *over a scope of tasks, with respect to* ***priors***, ***experience***, *and* ***generalization difficulty***.
  (my highlighting, Chollet 2019, p. 27)



(Chollet 2019, p. 12)



(Chollet 2019, p. 41)

Supplementary material

- Reranking in transduction:

“For ranking, we aggregate candidates across all transformations. For each unique candidate y, we track both its frequency of appearance freq(y) across different transformations and its average beam search score E [sT(y)]. These are then ranked with frequency taking precedence over average score.“(Li et al. 2024, p. 46)

References

Chollet, F. (2019). On the measure of intelligence.

Devlin, J., Uesato, J., Bhupatiraju, S., Singh, R., Mohamed, A.-r., and Kohli, P. (2017). Robustfill: neural program learning under noisy i/o. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, page 990–998. JMLR.org.

Kahneman, D. (2011). Thinking, Fast and Slow. Penguin Books Limited.

Li, W.-D., Hu, K., Larsen, C., Wu, Y., Alford, S., Woo, C., Dunn, S. M., Tang, H., Naim, M., Nguyen, D., Zheng, W.-L., Tavares, Z., Pu, Y., and Ellis, K. (2024). Combining induction and transduction for abstract reasoning.

Sun, Y., Wang, X., Liu, Z., Miller, J., Efros, A. A., and Hardt, M. (2020). Test-time training with self-supervision for generalization under distribution shifts. In *Proceedings of the 37th International Conference on Machine Learning*, ICML'20. JMLR.org.