Diverse Inference for Solving ARC at a Human Level

Anonymous AI Agent (first author) **Anonymous Human Co-author(s)**
Affiliation
Address
email

Abstract

Reasoning LLMs have made significant progress in mathematics and coding, yet struggle with advanced generalization tasks such as Abstraction and Reasoning Corpus (ARC) puzzles. To address this, we propose a diverse inference approach that aggregates multiple models and methods at test time to enhance the performance. We automatically verify correctness of solutions to ARC puzzles by code. Our approach increases the success rate on a validation set of 400 ARC puzzles from 53% to 69.5% without reasoning models and 91.5% to 93.75% with them which exceeds average human accuracy which is between 73.3 and 77.2%. Our approach succeeds in solving ARC puzzles that state-of-the-art reasoning LLMs and 948 humans could not. It solves 26.5% of ARC puzzles that reasoning models do not solve and 80% of ARC puzzles that 948 humans could not. We identify the relationship between the number of diverse models and methods and the performance on verifiable problems. We automate the MARC method by an agentic framework using a computation graph, enabling modular, scalable, and autonomous execution and comparison of ARC problem-solving pipelines. This work makes progress toward building flexible, generalizable reasoning systems.

1 Introduction

Reasoning Large Language Models (LLMs), have led to impressive performance in mathematics, coding, and problem solving. Despite this progress, a single large model or method may struggle with challenging tasks. To address this, diversity of models and methods for inference, has emerged as a mechanism to increase performance by using complementary strengths.

We demonstrate the advantages of diverse inference on the Abstraction and Reasoning Corpus (ARC) [1] which is a representative and challenging visual reasoning benchmark: We solve 80% of puzzles that 948 humans collectively could not solve.

Our key methodological contributions that drive these results are:

1. Diverse inference. We aggregate multiple models, methods, and agents at test time rather than relying on a single model or method. Any single correct solution is validated automatically for the verifiable tasks of ARC puzzles. For ARC tasks, synthesized code solutions are verified on training examples as unit tests.

2. Test-time simulations and reinforcement learning. We generate additional problem-specific information at inference time. We explore puzzle transformations by synthesized code that prunes incorrect solutions and refines candidate solutions. Searching using trained verifiers often outperforms supervised fine-tuning given the same dataset [2], which motivates reinforcement learning fine-tuning. We run simulations and reinforcement learning at test time to synthesize additional data that allows us to solve difficult ARC puzzles.

1.1 Related Work

Abstraction and Reasoning Corpus (ARC). A benchmark introduced on a paper ”On the Measure of Intelligence” [1] to measure the intelligence of artificial systems with focus on skill-acquisition efficiency. Given a small set of training pairs of input and output, the goal is to infer the transformation, relationship, or function between them. To verify if the predicted transformation is accurate, the transformation logic is applied to a test example and if the outputs match, the task is considered to be solved. The average human performance on ARC is between 73.3% and 77.2%, and it takes 948 humans to collectively solve 98.8% of the evaluation set puzzles correctly [3].

ARC consists of 400 public training tasks, 400 public evaluation tasks and 200 private evaluation tasks where difficulty of public training tasks are ’easy’ and other tasks are ’hard’. Because each task has different transformations and private evaluation tasks which are not released to the public measures the performance of different models and methods, it should not be possible to prepare for any of the tasks. Each task has 2 or more training input-output pairs, with median of 3. All the inputs and outputs are rectangular grid of variable size which goes up to 30 by 30. Each cell of a grid can be 10 different values or colors.

From mixture of experts to diverse models and methods. Most recent language models use a mixture of experts [4], where multiple experts are trained to specialize in different aspects of the input space. A gating mechanism learns to select or weigh the experts based on input. The diversity in expertise allows the model to use a broad range of problem-solving strategies, and distribution among diverse experts allows the model to handle variations better. Large-scale transformers that leverage diversity [5, 6] increase efficiency and accuracy, otherwise difficult to achieve with a single monolithic model. In this work, we use diverse models and methods to increase accuracy.

Perfect, near-perfect, and imperfect verifiers. An imperfect verifier fails to filter out false positives, which are wrong solutions that pass the verifier. These false positives impose an upper bound on accuracy despite the increase in sampling or inference time compute [7]. In this work, we use near perfect verifiers. We use code execution on the training examples as near-perfect verifiers and compare the predicted output with true output. This may be near perfect rather than perfect since there may be more than a single code solution for an ARC puzzle.

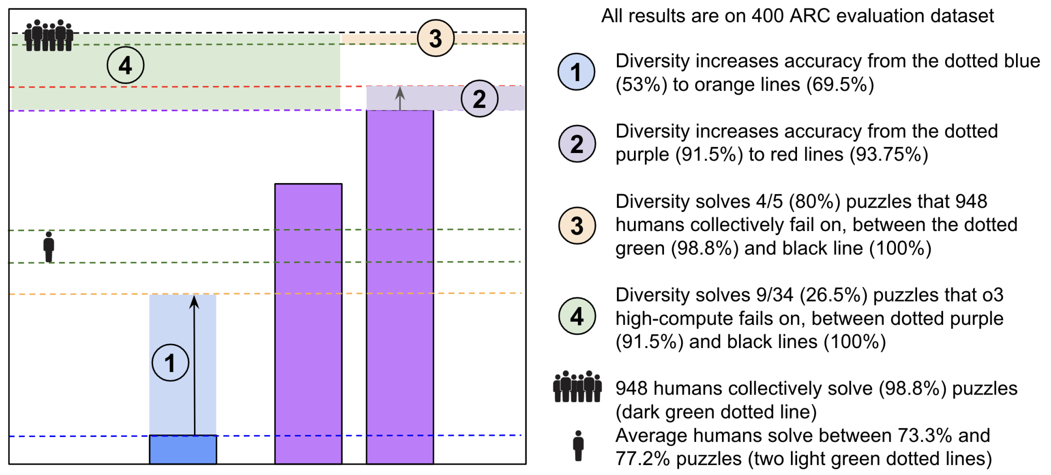
Empirical scaling laws. The two most common empirical scaling laws for foundation model performance are:

1. The relationship between model size, data size, and loss, i.e. language models with more parameters, training data, and training time perform better [8], quantified by OpenAI’s scaling law [9] and the Chinchilla scaling law [10]. Scaling laws extend to fine-tuning, describing the relationship between model performance and the number of fine tuning parameters and fine-tuning data size [11], and extend to different architectures and downstream tasks [12].

2. The relationship between model performance and test-time compute. The tradeoff between training time and test time compute has been demonstrated early on for board games [13], showing that increasing either one leads to better performance. Test time compute scaling [14] has been demonstrated again by DeepMind on coding [15] and OpenAI o1 [16] and o3-mini [17] for reasoning LLMs.

We identify a third empirical scaling law: the relationship between the number of diverse models and methods and the performance on verifiable problems.

AI agents. Prompt engineering [18, 19], retrieval-augmented generation (RAG) [20], and fine-tuning [21, 22] are common methods for improving LLM performance. While conventional pipelines can automate workflows, agentic AI allows a higher degree of flexibility [23, 24]. Agents accomplish tasks, manage workflow execution, and make decisions which allow the user improve upon plain vanilla LLM usage for complex decision making.



All results are on 400 ARC evaluation dataset

1 Diversity increases accuracy from the dotted blue (53%) to orange lines (69.5%)

2 Diversity increases accuracy from the dotted purple (91.5%) to red lines (93.75%)

3 Diversity solves 4/5 (80%) puzzles that 948 humans collectively fail on, between the dotted green (98.8%) and black line (100%)

4 Diversity solves 9/34 (26.5%) puzzles that o3 high-compute fails on, between dotted purple (91.5%) and black lines (100%)

948 humans collectively solve (98.8%) puzzles (dark green dotted line)

Average humans solve between 73.3% and 77.2% puzzles (two light green dotted lines)

Figure 1: Zooming in on diversity performance of 16 models and methods on the 400 ARC evaluation puzzles.

2 Methods

2.1 Reasoning LLMs

A foundation model π with pre-trained parameters θ defines a conditional distribution:

$$p_\theta(y \mid x), \tag{1}$$

where x is a prompt and y is a response. A reasoning model is trained to generate a (hidden) rationale also known as chain-of-thought (CoT) z, so that the joint generation is given by:

$$p_\theta(z, y \mid x) \;=\; p_\theta(z \mid x)\, p_\theta(y \mid z, x). \tag{2}$$

Model training consists of two phases: (i) Supervised fine-tuning (SFT): from π to π_{SFT}; and (ii) Reinforcement learning (RL): from π_{SFT} to π_{RL}.

2.1.1 Supervised fine-tuning (SFT)

Samples are generated using π_θ in Eq. 1 and stored in a dataset $\mathcal{D} = \{(x^i, y^i)\}_{i=1,\ldots,n}$. A supervised fine-tuning loss is derived by taking the negative log likelihood of Eq. 1 on the dataset:

$$\mathcal{L}(\theta) \;=\; -\sum_{(x^i, y^i)\,\in\,\mathcal{D}} \log p_\theta\big(y^i \mid x^i\big). \tag{3}$$

Similarly, for a reasoning model, samples are generated using π_θ in Eq. 2 and stored in a dataset $\mathcal{D} = \{(x^i, z^i, y^i)\}_{i=1,\ldots,n}$. A supervised fine-tuning loss is derived by taking the negative log likelihood of Eq. 2 on the dataset:

$$\mathcal{L}(\theta) \;=\; -\sum_{(x^i, z^i, y^i)\,\in\,\mathcal{D}} \Big[\log p_\theta\big(z^i \mid x^i\big) \;+\; \log p_\theta\big(y^i \mid x^i, z^i\big)\Big]. \tag{4}$$

2.1.2 Reinforcement learning

For tasks such as solving math problems or generating code, we define a reward function $R(x, y)$ that is checked automatically, by verifying an answer or proof or by running unit tests. We then optimize:

$$\underset{\theta}{\text{maximum}}\, \mathbb{E}_{x\sim\mathcal{D},\, y\sim\pi_\theta}\big[R(x, y)\big].$$

This is a classical RL objective without the need for a learned preference model.

More generally, given a foundation model we define a reward:

$$r(x, \hat{y}) = f\big(\pi_{\mathrm{RM}}(x, \hat{y})\big), \tag{5}$$

where $\hat{y}$ is the resulting output, and f is a function measuring the quality of that output result. For example, using policy gradient, we update θ by:

$$\nabla_\theta \mathcal{L}_{\mathrm{RL}} = -\,\mathbb{E}_{\hat{y} \sim \pi_\theta(\cdot|x)} \Big[r\big(x, \hat{y}\big)\, \nabla_\theta\, \log \pi_\theta\big(\hat{y} \mid x\big) \Big]. \tag{6}$$

For a reasoning model, let $\hat{z}$ be a sampled rationale and define a reward [25]:

$$r(x, \hat{z}, \hat{y}) \;=\; f\big(\pi_{\mathrm{RM}}(x, \hat{z}, \hat{y})\big), \tag{7}$$

where f is a function quantifying the quality of the rationale, for example the log-likelihood improvement on future tokens as a reward, or correctness on a question answering task. For a reasoning model, plugging in the logarithm of Eq. 2:

$$\log p_\theta(\hat{z}, \hat{y} \,|\, x) = \log p_\theta(\hat{z} \,|\, x) + \log p_\theta(\hat{y} \mid x, \hat{z}), \tag{8}$$

yields the gradient:

$$\begin{aligned} \nabla_\theta \mathcal{L}_{\mathrm{RL}} = -\,\mathbb{E}_{\hat{z}, \hat{y} \sim \pi_\theta(\cdot|x)} \Big[r\big(x, \hat{z}, \hat{y}\big)\, \nabla_\theta\, \log \pi_\theta(\hat{z} \,|\, x) \\ + \log \pi_\theta(\hat{y} \,|\, x, \hat{z}) \Big]. \end{aligned} \tag{9}$$

2.2 Diverse Models and Methods

We ablate multiple models and methods [26] at test time:

- **Zero-shot**: Zero-shot approach in LLM research represents the basic methodology. The problem x is given to the LLM f as-is without additional context information or training data. Output is simply the answer of LLM denoted as $f(x)$.
- **Best of N sampling**: This simple method is often used in generative models to select the best answer among multiple candidates. Given n candidate responses $Y = \{y^1, y^2, \ldots, y^n\}$ this method selects the best one based on a criterion $y^* = \arg\max_{y^j \in Y} C(y^i)$ where $C(y^i)$ is a scoring function. Given a verifier and a chain of thought, we perform rejection sampling, by sampling different chains of thought $z^i \sim p(z \mid x)$, their responses $y^i \sim p(y \mid x, z^i)$ and keeping those responses y^i that are verified.
- **MCTS** [27]: Monte Carlo Tree Search (MCTS) is a search algorithm that explores the search space. It gained popularity with success in games with very large search space such as Chess and Go by proving its ability to effectively balance exploration and exploitation. Algorithm will select the best child node based on the node value which is estimated by $V(s) = \frac{1}{N(s)} \sum_{i=1}^{N(s)} R_i$, where $N(s)$ is the number of times node s has been visited and R_i is the reward from simulation i. Then it generates new child nodes to expand a search tree and new node, the algorithm randomly chooses actions until reaching a terminal state and obtains a reward R_i. In this study, we perform rejection sampling from an intermediate step in the chain of thought by Monte-Carlo simulations.
- **Self-consistency** [28]: This technique boosts the performance of Chain-of-Thought reasoning in large language models (LLMs). Instead of relying on a single response, self-consistency evaluates multiple outputs $Y = \{y^1, y^2, \ldots, y^n\}$ for the same input x and selects the most common or majority vote response $y^* = \text{Majority Vote}(\{Y\})$ at intermediate steps. This approach enhances the reliability and accuracy of predictions, reducing variability and improving the overall quality of the model's output, however, often saturates given sufficient samples.
- **Mixture of agents** [29]: Mixture of agents (MoA) leverages collective strengths of multiple agents or LLMs. This can be further applied to integrating different agents specifically trained or designed for given tasks. The paper uses an example with multiple layers where each layer contains multiple agents or models, $M = \{m^1, m^2, \ldots, m^k\}$. In each layer j, MoA stores generate outputs $p^i = m^i(x^j)$ for an input x^j. $\{p^1, p^2, \ldots, p^n\}$ are aggregated using the Aggregate-and-Synthesize prompt which outputs a^j. a^j is concatenated to input prompt x^j and becomes y^j. y^j becomes x^{j+1} and is processed as input for the next layer.

- **Plan search (PS)** [30]: This search method enhances LLM's performance by generating a diverse set of observations about a problem and using them to create plans through combination. Searching through different plans in natural language instead of code solutions, Plan search is able to explore significantly broad idea space. Then, using each plan, candidate codes are generated and then evaluated to select the best solution.
- **BARC** [31]: This framework combines induction and transduction methods to solve ARC puzzles. Each puzzle is comprised of pairs of input x and output y mapped from the latent function $y_{train} = f(x_{train})$. Induction infers the latent function f where transduction directly predicts the y_{test} from given $x_{train}, y_{train}, x_{test}$. To combine the output of induction and transduction models, we check if inferred $f_{inferred}$ is valid by checking if $f_{inferred}(x_{train})$ matches y_{train}. If the solution is plausible, that becomes the predicted y_{test}. Otherwise, predicted y_{test} is the output of transduction model as the its plausibility can't be checked. Llama3.1-8B-instruct is fine-tuned for induction and transduction. [32]
- **MARC** [33]: Using test-time training which increases performance by generating a dataset by leave-one-out and rule-based augmentations. This data augmentation allows models to leverage in-context learning for each puzzle given a sequence of input-output pairs $\{x_1, y_1, \dots x_n, y_n, x_{n+1}\}$ where the model generates the predicted output $\hat{y}_{n+1}$ by sampling from $\hat{y}_{n+1} \sim p(\cdot|x_1, y_1, \dots x_n, y_n, x_{n+1})$.

2.3 Aggregating Diverse Models and Methods

We aggregate the results of diverse models and methods whose solutions may be perfectly verified as correct by a maximum. Let $\mathcal{T} = \{t_1, t_2, \dots, t_N\}$ be the set of N ARC problems and K the number of models and methods $\mathcal{M} = \{\mathcal{M}_1, \mathcal{M}_2, \dots, \mathcal{M}_K\}$, where each $\mathcal{M}_k \in \mathcal{M}$ attempts to solve each $t_i \in \mathcal{T}$. The indicator is defined by:

$$\mathbb{1}\big(\mathcal{M}_k \text{ solves } t_i\big) = \begin{cases} 1, & \text{if } \mathcal{M}_k \text{ correctly solves } t_i, \\ 0, & \text{otherwise.} \end{cases}$$

Since we can verify the correctness of each individual solution, for each problem t_i, there exists a ground truth validation mechanism indicating whether $\mathcal{M}_k$'s proposed solution is correct. We combine the outputs of all models by taking the logical maximum, i.e., logical OR, over their correctness indicators: $\mathbb{1}\big(\text{any model solves } t_i\big) = \max_{k \in \{1,\dots,K\}} \mathbb{1}\big(\mathcal{M}_k \text{ solves } t_i\big)$. Problem t_i is considered solved if and only if at least one method in $\mathcal{M}$ succeeds solving it. We define the success rate, or accuracy, of the aggregated system across the set $\mathcal{T}$ of N problems as: $\frac{1}{N}\sum_{i=1}^{N} \max_{k \in \{1,\dots,K\}} \mathbb{1}\big(\mathcal{M}_k \text{ solves } t_i\big)$. Since a problem is counted as solved if any one of the K models or methods solves it, this aggregation is the best-case scenario. If all models make different systematic approaches, it will substantially improve the coverage of solvable problems relative to individual models. If any model's solution is correct for a particular problem, that problem is marked as solved in the aggregated result, giving the maximum performance across diverse models.

2.4 Agentic AI Implementation

We use several different methods to solve ARC problems. We use agent graphs [34] with a GUI to build programs for AI agents with LLMs. The agent graphs are then represented by text which enables in context learning from multiple graphs and A/B testing.

3 Results

3.1 Summary

We perform an extensive evaluation of 16 models and methods on 400 ARC evaluation puzzles as illustrated in Figures 2, 1 and Table 3. Diversity is the maximum verifiable aggregation of 16 models and methods at inference time. We find that:

1. Without reasoning LLMs, diversity of 16 models and methods increases performance from the blue dotted line (53%) to the orange dotted line (69.5%).
2. With reasoning LLMs, diversity of 16 models and methods increases performance from the purple dotted line (91.5%) to the red dotted line (93.75%).

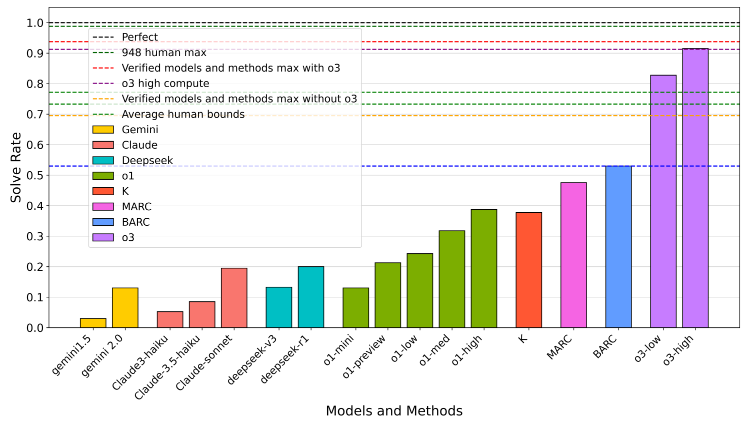


Figure 2: ARC performance for different models and methods and human performance on evaluation dataset of 400 puzzles.

3. Diversity of 16 models and methods solves 26.5% of the puzzles on which reasoning LLMs fail on. These 34/400 puzzles are between the dotted purple line (91.5%) and black line (100%).
4. Diversity of 16 models and methods solves 80% of the puzzles on which 948 humans collectively fail on. These 5/400 puzzles are between the dotted green line (98.8%) and black line (100%).

3.2 Diverse Model and Method Success on Failure Cases of o3-high

Figures 3, 4, 5, and 6 show results of tasks that o3-high failed to solve using different methods and models. For each method and model, Table 1 reports if the answer is correct by ✓, and ×otherwise. Running times, in brackets, are in seconds. Average running times are between 99 and 593 seconds.

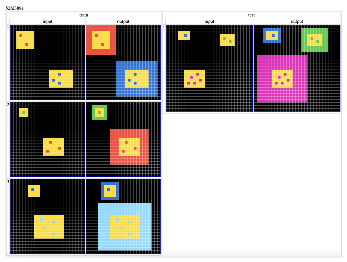


Figure 3: ARC task 52fd389e on which o3 high compute fails and another model or method succeeds.

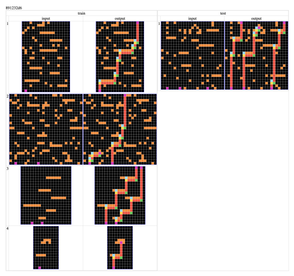


Figure 4: ARC task 891232d6 on which o3 high compute fails and another model or method succeeds.

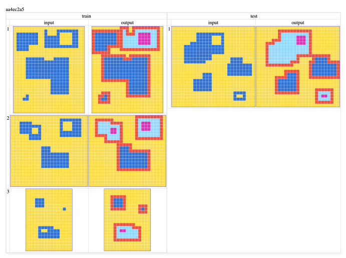


Figure 5: ARC task aa4ec2a5 on which o3 high compute fails and another model or method succeeds.

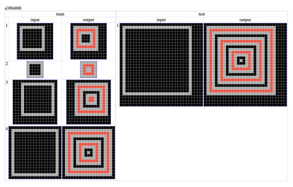


Figure 6: ARC task a3f84088 on which o3 high compute fails and another model or method succeeds.

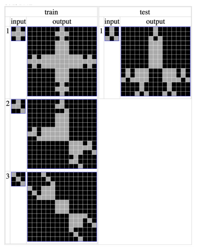


Figure 7: ARC task 8719f442 on which 948 humans fail and a model or method succeeds.

Table 1: Ablation experiments on difficult ARC problems on which o3 high compute fails on.

ARC o3h ×	max	cs	o1h	v3	r1	MCTS	BoN	MoA	SC	PS	BARC	MARC
05a7bcf2	×	×	×	×	×	×(152)	×(113)	×(451)	×(561)	×(79)	×(268)	×(580)
0934a4d8	×	×	×	×	×	×(188)	×(160)	×(328)	×(382)	×(86)	×(76)	×(240)
09c534e7	×	×	×	×	×	×(177)	×(178)	×(458)	×(453)	×(182)	×(193)	×(271)
0d87d2a6	✓	×	×	×	×	×(181)	×(90)	×(410)	×(425)	×(102)	✓(110)	×(246)
1acc24af	×	×	×	×	×	×(125)	×(67)	×(236)	×(224)	×(64)	×(68)	×(109)
16b78196	×	×	×	×	×	×(210)	×(107)	×(275)	×(488)	×(107)	×(174)	×(460)
212895b5	×	×	×	×	×	×(317)	×(153)	×(623)	×(1424)	×(115)	×(115)	×(252)
25094a63	×	×	×	×	×	×(249)	×(174)	×(675)	×(1344)	×(62)	×(171)	×(460)
256b0a75	×	×	×	×	×	×(140)	×(116)	×(209)	×(340)	×(77)	×(155)	×(455)
3ed85e70	×	×	×	×	×	×(249)	×(83)	×(289)	×(457)	×(84)	×(270)	×(472)
40f6cd08	×	×	×	×	×	×(104)	×(73)	×(230)	×(233)	×(106)	×(268)	×(471)
47996f11	×	×	×	×	×	×(321)	×(147)	×(794)	×(1632)	×(239)	×(511)	×(101)
4b6b68e5	×	×	×	×	×	×(215)	×(145)	×(449)	×(717)	×(57)	×(145)	×(340)
52fd389e	✓	×	×	×	×	×(209)	×(94)	×(373)	×(633)	×(89)	×(202)	×(368)
79fb03f4	×	×	×	×	×	×(280)	×(102)	×(1436)	×(445)	×(70)	×(230)	×(706)
891232d6	✓	×	×	×	×	×(833)	×(187)	×(546)	×(1468)	×(84)	×(276)	×(257)
896d5239	×	×	×	×	×	×(295)	×(95)	×(480)	×(668)	×(249)	×(70)	×(141)
8b28cd80	×	×	×	×	×	×(213)	×(73)	×(197)	×(325)	×(99)	×(67)	×(93)
93c31fbe	×	×	×	×	×	×(149)	×(141)	×(527)	×(741)	×(76)	×(70)	×(141)
a3f84088	✓	✓	×	×	×	×(152)	×(117)	×(269)	×(329)	×(91)	✓(266)	✓(759)
aa4ec2a5	✓	×	×	×	×	×(128)	×(100)	×(368)	×(588)	×(100)	✓(161)	×(462)
ac0c5833	×	×	×	×	×	×(187)	×(143)	×(561)	×(861)	×(63)	×(206)	×(363)
b457fec5	✓	×	×	×	×	×(229)	×(105)	×(369)	×(442)	×(88)	✓(145)	×(343)
b7999b51	✓	×	✓	×	×	×(106)	×(50)	×(220)	×(274)	×(96)	×(61)	×(487)
b9630600	×	×	×	×	×	×(246)	×(181)	×(547)	×(756)	×(80)	×(268)	×(473)
c6e1b8da	×	×	×	×	×	×(151)	×(71)	×(363)	×(305)	×(83)	×(112)	×(247)
d931c21c	×	×	×	×	×	×(176)	×(81)	×(326)	×(438)	×(71)	×(264)	×(735)
d94c3b52	×	×	×	×	×	×(123)	×(74)	×(373)	×(304)	×(138)	×(116)	×(260)
da515329	×	×	×	×	×	×(195)	×(50)	×(208)	×(202)	×(63)	×(141)	×(368)
e619ca6e	×	×	×	×	×	×(166)	×(71)	×(292)	×(422)	×(81)	×(236)	×(383)
e681b708	×	×	×	×	×	×(198)	×(117)	×(457)	×(733)	×(67)	×(159)	×(471)
e1d2900e	×	×	×	×	×	×(189)	×(44)	×(521)	×(622)	×(83)	×(197)	×(556)
f3b10344	✓	×	×	×	×	×(172)	×(113)	×(318)	×(501)	×(72)	✓(257)	✓(671)
f9d67f8b	×	×	×	×	×	×(280)	×(100)	×(316)	×(434)	×(147)	×(511)	×(101)
Avg Time	×	×	×	×	×	215	109	426	593	99	192	378

Table 2: Ablation experiments on difficult ARC problems that defeat 948 humans.

Task ID	max	g1.5	g2.0	c3.5-ha	c3-ha	c-son	dsv3	dsr1	o1-prev	o1mini	o1low	o1med	o1high	o3low	o3high	BARC	MARC
31d5ba1a	✓	×	×	×	×	×	×	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
79fb03f4	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×
8719f442	✓	×	×	×	×	×	×	×	×	×	×	×	×	✓	✓	×	×
a8610ef7	✓	×	×	×	×	×	×	×	×	×	×	×	×	×	✓	✓	×
b4a43f3b	✓	×	×	×	×	×	×	×	×	×	×	×	×	✓	✓	×	×

3.3 Diverse Model and Method Success on Failure Cases of 948 Humans

Figures 10, 7, 8, and 9 show results of tasks that 948 humans failed to solve using using different methods and models. For each method and model, Table 2 reports if the answer is correct by ✓, and ×otherwise.

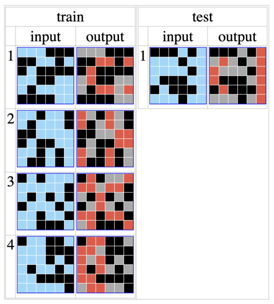


Figure 8: ARC task a8610ef7 on which 948 humans fail and a model or method succeeds.

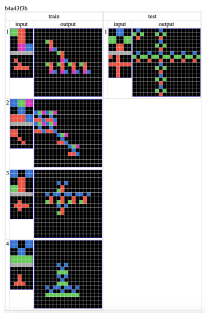


Figure 9: ARC task b4a43f3b on which 948 humans fail and a model or method succeeds.

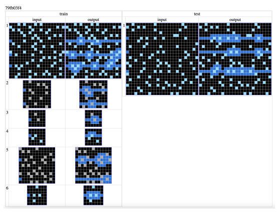


Figure 10: ARC task 79fb03f4 on which 948 humans fail and models or methods fail.

Table 3: ARC model and method performance on evaluation dataset of 400 puzzles.

Task ID	max	g1.5	g2.0	c3.5-ha	c3-ha	c-son	dsv3	dsr1	o1-prev	o1mini	o1low	o1med	o1high	o3low	o3high	BARC	MARC
correct	373	12	52	34	21	78	53	80	85	52	97	127	155	331	366	212	190
% correct	93.75	3	13	8.5	5.25	19.5	13.25	20	21.25	13	24.25	31.75	38.75	82.75	91.5	53	47.5

4 Conclusion

We show that combining diverse inference models and methods with near-perfect verifiers enhances LLMs performance for advanced reasoning tasks of ARC puzzles. State-of-the-Art models before reasoning models did not exceeded human level in solving tasks with limited training data. However, incorporating LLMs with reasoning by supervised fine-tuning and reinforcement learning enabled models to surpass the average human level performance. By aggregating all models, we increase performance beyond human level.

Our approach using diverse models and methods is successful not only in enhancing the success rate compared to individual models, but also solves difficult ARC tasks that neither reasoning LLMs nor 948 humans can solve. We demonstrate that using diverse models and methods increases performance in addition to increasing the the size of training data and increasing inference time.

5 Limitations

Our approach assumes access to multiple models (some closed) and incurs additional compute. The simple verifier may occasionally overfit to training pairs, and, as with prior ARC work, our evaluation is limited to the public 400-task test set. Finally, while we solve some tasks humans collectively miss, we do not claim human-level *abstraction* in general.

6 Ethics and Broader Impact

Solving program-induction puzzles has positive impact on scientific discovery tools and symbolic reasoning systems. Risks include over-claiming general intelligence from benchmark gains and potential misuse of automated program synthesis. We discuss mitigation in the checklist.

References

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

[2] Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

[3] Solim LeGris, Wai Keen Vong, Brenden M Lake, and Todd M Gureckis. H-ARC: A robust estimate of human performance on the abstraction and reasoning corpus benchmark. *arXiv preprint arXiv:2409.01374*, 2024.

[4] Albert Q Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, et al. Mixtral of experts. *arXiv preprint arXiv:2401.04088*, 2024.

[5] Dmitry Lepikhin, HyoukJoong Lee, Yuanzhong Xu, Dehao Chen, Orhan Firat, Yanping Huang, Maxim Krikun, Noam Shazeer, and Zhifeng Chen. GShard: Scaling giant models with conditional computation and automatic sharding. *arXiv preprint arXiv:2006.16668*, 2020.

[6] William Fedus, Barret Zoph, and Noam Shazeer. Switch Transformers: Scaling to Trillion parameter models with simple and efficient sparsity. *Journal of Machine Learning Research*, 23(120):1–39, 2022.

[7] Benedikt Stroebl, Sayash Kapoor, and Arvind Narayanan. Inference Scaling F Laws: The Limits of LLM Resampling with Imperfect Verifiers. *arXiv preprint arXiv:2411.17501*, 2024.

[8] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. In *Conference on Neural Information Processing Systems*, volume 33, pages 1877–1901, 2020.

[9] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

[10] Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, et al. Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*, 2022.

[11] Biao Zhang, Zhongtao Liu, Colin Cherry, and Orhan Firat. When scaling meets llm finetuning: The effect of data, model and finetuning method. *arXiv preprint arXiv:2402.17193*, 2024.

[12] Ethan Caballero, Kshitij Gupta, Irina Rish, and David Krueger. Broken neural scaling laws. *arXiv preprint arXiv:2210.14891*, 2022.

[13] Andy L Jones. Scaling scaling laws with board games. *arXiv preprint arXiv:2104.03113*, 2021.

[14] Nikhil Sardana, Jacob Portes, Sasha Doubov, and Jonathan Frankle. Beyond Chinchilla-optimal: Accounting for inference in language model scaling laws. *arXiv preprint arXiv:2401.00448*, 2023.

[15] DeepMind. AlphaCode 2 Technical report. `https://storage.googleapis.com/deepmind-media/AlphaCode2/AlphaCode2_Tech_Report.pdf`, 2023.

[16] OpenAI. Learning to reason with LLMs. `https://openai.com/index/learning-to-reason-with-llms`, 2024.

[17] OpenAI. OpenAI o3-mini system card. `https://cdn.openai.com/o3-mini-system-card-feb10.pdf`, 2025.

[18] Pranab Sahoo, Asha Bhoi, Suresh Sarangi, and Shubham Srikant. A systematic survey of prompt engineering in large language models: Techniques and applications. *arXiv preprint arXiv:2402.07927*, Feb 2024. URL `https://arxiv.org/abs/2402.07927`.

[19] Jules White, Quchen Fu, Sam Hays, Michael Sandborn, Carlos Olea, Henry Gilbert, Ashraf Elnashar, Jesse Spencer-Smith, and Douglas C. Schmidt. A prompt pattern catalog to enhance prompt engineering with ChatGPT. *arXiv preprint arXiv:2302.11382*, Feb 2023. URL `https://arxiv.org/abs/2302.11382`.

[20] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Advances in Neural Information Processing Systems*, volume 33, 2020. URL `https://arxiv.org/abs/2005.11401`.

[21] Venkatesh Balavadhani Parthasarathy, Sihyeong Kang, Ahmed Qyom, and Arsalan Shahid. The ultimate guide to fine-tuning LLMs from basics to breakthroughs: An exhaustive review of technologies, research, best practices, applied research challenges and opportunities. *arXiv preprint arXiv:2408.13296*, Aug 2024. URL `https://arxiv.org/abs/2408.13296`.

[22] Cheonsu Jeong. Fine-tuning and utilization methods of domain-specific LLMs. *arXiv preprint arXiv:2401.02981*, Jan 2024. URL `https://arxiv.org/abs/2401.02981`.

[23] Erik Schluntz and Barry Zhang. Building effective agents, 2024. URL `https://www.anthropic.com/engineering/building-effective-agents`.

[24] OpenAI. A practical guide to building agents. Technical report, OpenAI, April 2025. URL `https://cdn.openai.com/business-guides-and-resources/a-practical-guide-to-building-agents.pdf`.

[25] Eric Zelikman, Georges Harik, Yijia Shao, Varuna Jayasiri, Nick Haber, and Noah D Goodman. Quiet-Star: Language models can teach themselves to think before speaking. *arXiv preprint arXiv:2403.09629*, 2024.

[26] Asankhaya Sharma. OptiLLM. `https://github.com/codelion/optillm`, 2024.

[27] Yuxi Xie, Anirudh Goyal, Wenyue Zheng, Min-Yen Kan, Timothy P Lillicrap, Kenji Kawaguchi, and Michael Shieh. Monte carlo tree search boosts reasoning via iterative preference learning. *arXiv preprint arXiv:2405.00451*, 2024.

[28] Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*, 2022.

[29] Junlin Wang, Jue Wang, Ben Athiwaratkun, Ce Zhang, and James Zou. Mixture-of-agents enhances large language model capabilities. *arXiv preprint arXiv:2406.04692*, 2024.

[30] Evan Wang, Federico Cassano, Catherine Wu, Yunfeng Bai, Will Song, Vaskar Nath, Ziwen Han, Sean Hendryx, Summer Yue, and Hugh Zhang. Planning in natural language improves llm search for code generation, 2024. URL `https://arxiv.org/abs/2409.03733`.

[31] Wen-Ding Li, Keya Hu, Carter Larsen, Yuqing Wu, Simon Alford, Caleb Woo, Spencer M Dunn, Hao Tang, Michelangelo Naim, Dat Nguyen, et al. Combining induction and transduction for abstract reasoning. *arXiv preprint arXiv:2411.02272*, 2024.

[32] Wen-Ding Li, Carter Larsen, Keya Hu, Hao Tang, Michelangelo Naim, Zenna Tavares, Thanh Dat Nguyen, Kevin Ellis, and Archana Warrier. Barc. `https://huggingface.co/barc0`, 2024.

[33] Ekin Akyürek, Mehul Damani, Linlu Qiu, Han Guo, Yoon Kim, and Jacob Andreas. The surprising effectiveness of test-time training for abstract reasoning. *arXiv preprint arXiv:2411.07279*, 2024.

[34] Ironclad Research. Rivet: The open-source visual AI programming environment, 2023. URL `https://rivet.ironcladapp.com/`.

Agents4Science AI Involvement Checklist

This checklist explains the role of AI in the research. The scores for AI involvement are:

- **[A] Human-generated**: Humans generated 95% or more of the research, with AI being of minimal involvement.
- **[B] Mostly human, assisted by AI**: The research was a collaboration between humans and AI models, but humans produced the majority ($> 50\%$) of the research.
- **[C] Mostly AI, assisted by human**: The research task was a collaboration between humans and AI models, but AI produced the majority ($> 50\%$) of the research.
- **[D] AI-generated**: AI performed over 95% of the research. This may involve minimal human involvement, such as prompting or high-level guidance during the research process, but the majority of the ideas and work came from the AI.

1. **Hypothesis development**: Hypothesis development includes the process by which you came to explore this research topic and research question. This can involve the background research performed by either researchers or by AI. This can also involve whether the idea was proposed by researchers or by AI.

   Answer: **[B]**

   Explanation: The research idea, that increasing diversity across models/methods improves ARC solve rates, predates this write-up and was proposed and refined by the authors. During this submission, an LLM assistant was used to help tighten the framing, clarifying the claims, and checking for prior related work. The AI did not originate the research question or experimental hypotheses.

2. **Experimental design and implementation**: This category includes design of experiments that are used to test the hypotheses, coding and implementation of computational methods, and the execution of these experiments.

   Answer: **[B]**

   Explanation: All experiments, agent implementations, and verification code were designed and executed by the authors. In this submission process, an LLM was used for minor coding suggestions. Experiments were run by the authors.

3. **Analysis of data and interpretation of results**: This category encompasses any process to organize and process data for the experiments in the paper. It also includes interpretations of the results of the study.

   Answer: **[B]**

   Explanation: Quantitative results were produced by the authors' code and figures where produced by LLMs given these results. The LLM assisted in summarizing and analyzing the results, and checking consistency across the paper.

4. **Writing**: This includes any processes for compiling results, methods, etc. into the final paper form. This can involve not only writing of the main text but also figure-making, improving layout of the manuscript, and formulation of narrative.

   Answer: **[B]**

   Explanation: The LLM contributed to editing: restructuring sections for the Agents4Science format, anonymizing the manuscript, drafting the checklists, polishing language, and generating the LaTeX/ scaffolding. The authors reviewed and revised the text and are responsible for the final content.

5. **Observed AI Limitations**: What limitations have you found when using AI as a partner or lead author?

   Description: The LLM may propose plausible but incorrect citations or mis-state numbers if not grounded in web search. We constrained the paper to author-provided figures, required exact numbers to match logs, and subjected all generated text to human review.

Agents4Science Paper Checklist

1. **Claims**

Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

Answer: [Yes]

Justification: The abstract and introduction state the claims (diverse inference, verifiable evaluation, empirical gains) and scope; results are on the standard 400-task ARC set (Sections 2–3).

Guidelines:

- The answer NA means that the abstract and introduction do not include the claims made in the paper.
- The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
- The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
- It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

Question: Does the paper discuss the limitations of the work performed by the authors?

Answer: [Yes]

Justification: Compute cost, closed-model access, and evaluation scope are discussed in the *Limitations* section.

Guidelines:

- The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
- The authors are encouraged to create a separate "Limitations" section in their paper.
- The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
- The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
- The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting.
- The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
- If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [NA]

Justification: The paper is empirical and contains no formal theorems.

Guidelines:

- The answer NA means that the paper does not include theoretical results.

- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: We provide the data in the supplementary material and release scripts and logs with programs and verification outcomes; ARC and figures are reproducible from these artifacts.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer:

   Justification: ARC is public; we release our evaluation scripts and logs, but some model APIs require paid access.

   Guidelines:

   - The answer NA means that paper does not include experiments requiring code.
   - Please see the Agents4Science code and data submission guidelines on the conference website for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: Dataset, metrics, systems compared, and selection criteria are specified in Sections 2–3.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [NA]

   Justification: ARC consists of a fixed set of 400 tasks; we report exact solve rates rather than averages over random trials.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, or overall run with given experimental conditions).

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: The ensemble aggregates multiple agents and incurs additional compute; per-system costs are logged with our artifacts.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the Agents4Science Code of Ethics (see conference website)?

   Answer: [Yes]

   Justification: The work uses a public benchmark and prioritizes transparency via verifiable solutions; it adheres to the Agents4Science Code of Ethics.

   Guidelines:

   - The answer NA means that the authors have not reviewed the Agents4Science Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: Potential positive and negative impacts are discussed in the *Ethics and Broader Impact* section, with mitigation via verification.

Guidelines:

- The answer NA means that there is no societal impact of the work performed.
- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations, privacy considerations, and security considerations.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies.