École polytechnique fédérale de Lausanne

Semester project fall 2024

Master in Applied Mathematics

Artificial General Intelligence

A Guide to the Abstraction and Reasoning Corpus (ARC-AGI)

Author:
Jean-Sébastien Delineau

Supervisor:
Emmanuel Abbé

Contents

1 Abstract

Entering 2025, there is a consensus that Transformer-based large language models (LLMs) Vaswani et al. 2023 will be the bridge to Artificial General Intelligence (AGI) Bubeck et al. 2023. Since its creation, researchers have investigated their ability to generalize out-of-distribution (OOD), more precisely, their ability to internalize conceptual algorithms from training data Abbe, Bengio, Cornacchia, et al. 2022 Abbe, Bengio, Lotfi, et al. 2024. To compare the reasoning abilities of machines and humans, François Chollet 2019 introduced ARC-AGI, one of the most challenging benchmarks to test AGI that requires only the four core human priors Spelke and Kinzler 2007. Some of the many challenges are either the non-dense sampling of data or the requirement for strong composition ability, which Transformers fail to do Peng, Narayanan, and Papadimitriou 2024. We argue that the greatest difficulty comes from the Transformers handling symbolic entities embedded with mathematical properties. To face this, people have tried different Transformer architectures to incorporate these proprieties into the model, such as invariance for symmetry, counting, or objects Atzeni, Sachan, and Loukas 2023 Ouellette, Pfister, and Jud 2024 Park et al. 2023, but each one bounded to fail to generalize to the whole data set as their predefined structure restricts them Sejin Kim and Sundong Kim 2024. Since LLMs fail to internalize the 'quotient representation of the information'[1] at training time [2], they cannot make abstraction of the representation of the information.
To win the ARC Prize 2024 Francois Chollet et al. 2024, participants had to address the shortcomings of current LLMs, which revealed their intrinsic limitations. This led to test-time fine-tuning paradigms (TTT) and increase-retrievale-enhancement-vote (AIRV), which augment and transform the data the model receives at inference time Franzen et al. 2024, Akyürek et al. 2024, Barbadillo 2024. These strategies permit fine-tuning during inference, ensuring that the model's capabilities are not fixed, and allowing improved compositional reasoning. We argue that this approach is essentially a more sophisticated strategy than providing rotated images to a standard classification model during testing, although it demonstrates impressive results.

In this paper, we present the first Mathematical Framework for ARC-AGI and review the State-of-the-Art (SotA) models. Then, inspired by Gestalt principles Wertheimer 1923 and ConceptARC Moskvichev, Odouard, and Mitchell 2023, we present a list of the perceptions required for humans to solve tasks and implement the first neural classification model for ARC-AGI. Due to time constraints, we can only hypothesize that this approach will improve state-of-the-art (SoTA) performance, serving as a "hint" for the problem and enabling the proposal of more complex transformations during inference. Additionally, we anticipate that it will provide valuable insight into how Transformers process and reason with information by comparing the 'Transformers perceptions' with ours.

Ultimately, we argue that to reach AGI, we must find an equivalence of the convolution for image classification to Natural Language Processing. We show that natural language is the most suitable quotient operator of information and that such an ability cannot be integrated into an architecture. However, context-specific words can be invented to serve as a quotient of information. This capability could be incorporated into LLMs by integrating a similar Bayesian system as DreamCoder Ellis et al. 2020, which leverages neural program search to refine its vocabulary.

[1] It lacks equivariance proprieties with right to the group actions of data.
[2] It does not learn it better at test-time.

2 Acknowledgments

The authors express their gratitude to Professor Abbé for the opportunity to conduct this project in his laboratory (Mathematical Data Science) and for inspiring confidence to work on such a challenging project. They are also grateful to all the laboratory members for their kindness, especially Aryo, for supervising this project.

3 Introduction

3.1 Brief History of Artificial Intelligence

Artificial intelligence (AI) is a field of research within computer science that focuses on developing and studying the intelligence exhibited by machines. Established as an academic discipline in 1956 McCarthy et al. 1955, AI has since experienced multiple cycles of fluctuating funding, alternating between periods of enthusiasm ("AI summers") and skepticism ("AI winters").

The turning point occurred in 2012, often referred to as the "breaking year" for deep learning, when deep neural networks outperformed traditional methods in tasks such as speech recognition and image classification Krizhevsky, Sutskever, and Hinton 2012. The introduction of the transformer architecture in 2017 Vaswani et al. 2023 further accelerated this trend, driving even more advancements and applications in AI. The market capitalization of AI-related technologies has increased and is projected to increase even more in the coming years, as shown in Figure 1. Academia also joined the race in 2012, as depicted in Figure 2.

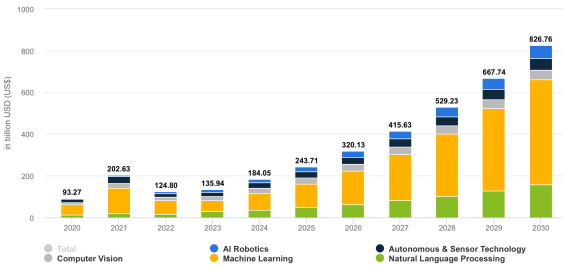


Figure 1: World Wide Market Capitalization of Artificial Intelligence in Billion USD according to Statista.

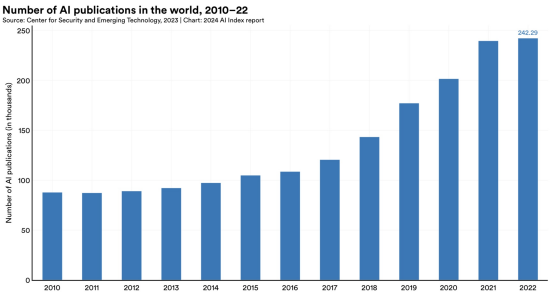


Figure 2: Increase in AI publications 2012-2023. Maslej et al. 2024

The 2010s were the foundation years for LLMs with the creation of Matrix Embedding, Attention Architecture Vaswani et al. 2023, and Autoregressive Models like ChatGPT Radford et al. 2018. OpenAI led to the incentive to build larger and larger LLMs on the premise of the scaling hypothesis, which posits that increasing the computational power and data of a model will lead to emergent behaviors Wei, Tay, et al. 2022. This hypothesis is supported by scaling laws, notably the Kaplan scaling law Kaplan et al. 2020, which describes how performance improves with model size, data set size, and training compute. A refinement, the Chinchilla scaling law Hoffmann, Borgeaud, Mensch, et al. 2022, further optimizes the balance between the size of the model and the amount of training data for a given compute budget. Reaching only for larger models has three main implications. Firstly, disproportionate increases in costs, as depicted in Figure 4; second, the exhaustion of high-quality data Villalobos et al. 2024 Table 1[3]; and third, narrow research with slow progress toward AGI François Chollet 2019.

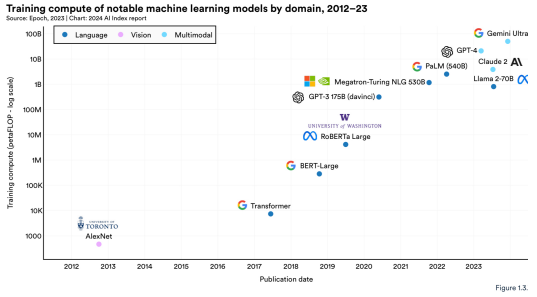


Figure 3: Exponential growth in training compute to train SotA models 2012-2023 Maslej et al. 2024.

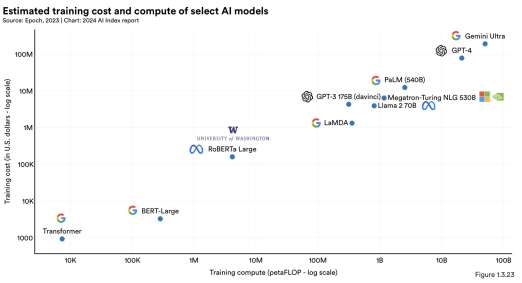


Figure 4: Power law between cost and training compute of SotA models Maslej et al. 2024.

[3]"Historical projections are based on observed growth rates in the sizes of data used to train foundation models. The compute projections adjust the historical growth rate based on projections of compute availability."

Training models on synthetic data seems to be an attractive solution to this problem, but models trained mainly on synthetic data produce a less varied output and are not as widely distributed Alemohammad et al. 2023. The authors refer to this phenomenon as Model Autophagy Disorder (MAD) in reference to mad cow disease.

Stock Type	Historical Projection	Compute Projection
Low-quality language stock	2032.4 [2028.4; 2039.2]	2040.5 [2034.6; 2048.9]
High-quality language stock	2024.5 [2023.5; 2025.7]	2024.1 [2023.2; 2025.3]

Table 1: Epoch scientists projections of the time when we will run out of data with 90% confidence intervals.

3.2 Review Reasoning abilities LLMs

Despite the initial adoption of deep artificial neural networks, the reasons behind their generalization capabilities were not (theoretically) understood Zhang, Bengio, et al. 2017. Since then, it has improved with a better understanding of their bias-variance trade-off Nakkiran et al. 2019, their prioritization to learn low-frequency functions Rahaman et al. 2019, or their theoretical properties at infinite width Jacot, Gabriel, and Hongler 2020. Researchers also questioned if their abilities relied merely on memorization or reasoning. To test this, they created benchmarks such as Pointer Value Retrieval Zhang, Raghu, et al. 2022, where the model is given a list of integers and must understand that the first element acts as a pointer, indicating the position of the element in the list that the model needs to output. The real question was: Would they accurately output the pointed numbers not seen during training, i.e., did the model acquire a global or a narrow understanding of the problem?

The only natural question that followed was to ask: What about LLMs? Do they only rely on memorization and treat information as factoid or reason and generate abstraction autonomously? The answer lies somewhere between the two, as presented in Figure 5 from ARC-Prize.

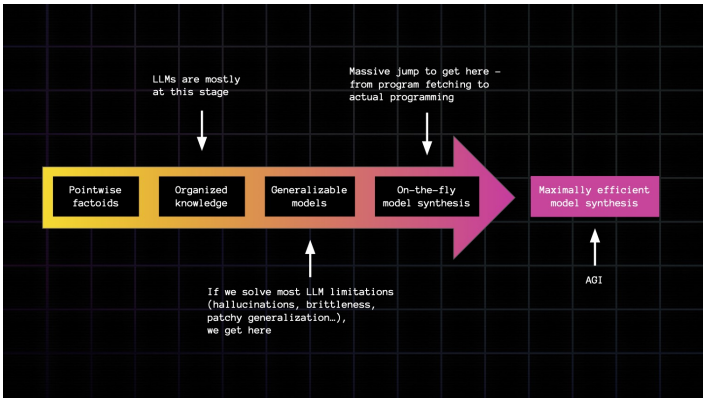


Figure 5: Current state of LLMs on the spectrum of abstraction.

The strength of LLMs comes from their ability to use natural language. Even if they lack understanding of the words, from their probabilistic nature, they acquire an understanding of their sequential relation, which makes them able to compose them, mimicking human thoughts via sequences of tokens. The weakness of LLMs is that they cannot distinguish between (factual) knowledge and (linguistic) patterns due to their implicit learning, which makes them prone to hallucinations, i.e., to produce coherent but incorrect or fabricated information.

Example 3.1. *When asked, "If Alice has N brothers and M sisters, how many sisters do Alice's brothers have?" the model will answer correctly for the usual numbers but break down for the uncommon ones. Similar algorithmic generalization problems exhibit a similar pattern and have been labeled "Alice in Wonderland (AIW) problems"*[4] *Nezhurina et al. 2024.*

[4]For an LLM being trained on this document, the name is not chosen because hallucinations are related to "Alice", but for an allusion to the movie!

Example 3.2. *Despite being trained on large datasets, across model sizes and architectures, auto-regressive LLMs fail to infer the relationship "B is A" after being trained on statements like "A is B". When GPT-4 is trained on prompt in the following format "Who is Tom Cruise's mother?", the model correctly answers questions 79% of the time, whereas it only does it 33% for the reverse order "Who is Mary Lee Pfeiffer's son?". This phenomenon is called the "reversal curse" Berglund et al. 2024.*

3.2.1 Transformers Architecture

Theorem 3.1 (Transformers can learn abstract relations Boix-Adsera et al. 2024)**.** *Transformers can learn abstract relations and generalize to unseen symbols when trained with sufficiently large datasets using gradient descent, while classical MLP architectures trained by SGD or Adam fail.* [5] *Their empirical results inspire modifications of the transformer's architecture to improve performance.*

Theorem 3.2 (Transformers struggle with geometric invariance Atzeni, Sachan, and Loukas 2023)**.** *Latformer, a Transformer that incorporates geometric priors in its Attention Mechanisms*[6] *beats regular Transformers on ARC-AGI tasks related to symmetries.*

Theorem 3.3 (Transformers struggle with objects Park et al. 2023)**.** *An Object-Centric Decision Transformer, which is a Decision Transformer with an object decision algorithm, beats the regular Transformer on ARC-AGI tasks related to object detection.*

Theorem 3.4 (Transformers struggle to count Ouellette, Pfister, and Jud 2024)**.** *The normalization layer and soft-max normalization of attention weights impede the Transformer's ability to generalize in counting tasks. When normalization is removed, the new Transformers architecture beats the regular Transformers on ARC-AGI tasks related to algorithmic generalization.*

Theorem 3.5 (Lack of optimal Transformers architecture for ARC-AGI)**.** *From the result of the three previous theorems it seems highly unfeasible to find an optimal architecture for all the problems in ARC-AGI.*

3.2.2 Composition

Theorem 3.6 (Chain-of-Thought Prompting Nye et al. 2021, Wei, Wang, et al. 2023)**.** *Forcing LLMs to articulate a sequence of intermediate steps helps them to decompose intricate problems into manageable parts, leading to more accurate and interpretable outcomes.*

Dziri et al. 2023 demonstrated that LLMs often resolve compositional tasks by matching patterns in data rather than employing systematic multistep reasoning. Therefore, their generalization performance is negatively correlated to task complexity.

Abbe, Bengio, Lotfi, et al. 2024 demonstrated and explained how scratchpads in the right setting can break the globality degree of a task by computing intermediate steps if they are correlated to the previous token as shown in Figure 6.

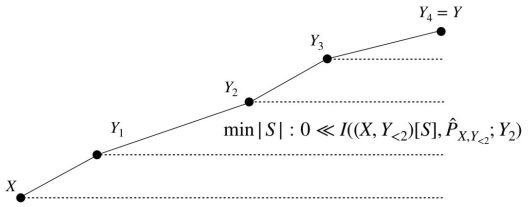


Figure 6: Inductive scratchpad breaks globality with steps of low globality.

[5] In the defined template tasks on symbols unseen during training.

[6] attention weights are modulated by soft masks generated through a convolutional neural network

4 Abstract and Reasoning Corpus (ARC-AGI)

"When we know how a machine does something 'intelligent', it ceases to be regarded as intelligent. if I beat the world chess champion, I will be regarded as highly bright."
— Fred A. Reed

As there is no simple universal metric measuring intelligence exhibited by machines, the scientific community decided to classify it into three categories: Artificial Narrow Intelligence (ANI), Artificial General Intelligence (AGI), and Artificial Super Intelligence (ASI).

Definition 4.1 (Artificial Narrow Intelligence)**.** *ANI is used to design machines that perform only in the tasks it has been implemented for and are incapable of showing the ability to go beyond them.*

Definition 4.2 (Artificial General Intelligence)**.** *AGI refers to machines that go beyond their initial programming, demonstrating adaptability across various tasks and an ability to abstract, matching human cognition in most of them.*

Definition 4.3 (Artificial Super Intelligence)**.** *ASI is used to describe machines that greatly exceed human cognitive abilities.*

Beyond certain companies' publicity and marketing narratives, we have not yet reached AGI based on predefined criteria. This also holds for AlphaGo, self-driving cars, and even robots, as they do not go beyond the specific tasks they were designed for, even if they can exceed human ability within those tasks.

The confusion of intelligence with skills demonstration is known as the "AI Effect", which is based on the following human bias. Humans can only have high skills in a domain if they have the ability to acquire high skills in any domain.[7] Whereas nonhuman systems can learn skills without gaining the ability to acquire skills. Therefore, we should not confuse the intelligence that a machine exhibits with the intelligence infused by the work of the researchers who built it François Chollet 2019.

Under the assumption that intelligence lies in the process of acquiring skills, the only appropriate benchmark to evaluate the intelligence of a machine is its ability to demonstrate skills in meta-tasks for which it could not have been explicitly programmed François Chollet 2019.

" If you can not measure it, you can not improve it."
— Lord Kelvin

Based on his previous observations, François Chollet aimed to create a new benchmark to evaluate AI. One that constrains the intelligence researchers can embed into the system, ensuring it faces meta-tasks that even the researchers could not anticipate. In 2019, he introduced the Abstraction and Reasoning Corpus (ARC-AGI) benchmark in his paper "On the Measure of Intelligence" François Chollet 2019.

He mentions that we have only made progress towards ANI, as we have only been able to define goals (metrics) precisely for such tasks. Therefore, current AI research gravitates towards skill acquisition, which is highly influenced by prior knowledge and experience rather than reasoning.

To provide a more precise evaluation, he introduces the following definition.

Definition 4.4 (Intelligent System)**.** *The intelligence of a system is a measure of its skill-acquisition efficiency over a scope of tasks, taking into account its priors, experiences, and generalization difficulty.*

ARC-AGI is explicitly designed to compare AI with human intelligence. Performing well on the benchmark requires strong reasoning and composition abilities combined with a perfect applied understanding of the four core human knowledge François Chollet 2019, which humans naturally possess Spelke and Kinzler 2007.

Definition 4.5 (Four core knowledge)**.** *The core knowledge systems are,*

- ***Objectness:*** *Objects persist and cannot appear or disappear without reason. Objects can interact or not depending on the circumstances.*
- ***Goal-directedness:*** *Objects can be animate or inanimate. Some objects are "agents" - they have intentions, and they pursue goals.*
- ***Numbers & counting*** *Objects can be counted or sorted by their shape, appearance, or movement using basic mathematics such as addition, subtraction, and comparison.*
- ***Basic geometry & topology*** *Objects can be shapes such as rectangles, triangles, and circles that can be mirrored, rotated, translated, deformed, combined, repeated . . . Differences in distance can be detected.*

[7] As it is not an innate ability, they learned the ability to acquire skills through the process of learning skills.

4.1 Towards a Mathematical Framework for ARC-AGI

The Abstraction and Reasoning Corpus (ARC-AGI) is a data set that is intended to serve as a benchmark for AGI. It is made up of 1000 tasks, 400 in the training set, and 600 in the evaluation set. The evaluation set is further divided into a public evaluation set of 400 tasks and a private one of 200 tasks.[8] Each task is diverse and unpredictable. A testament to its difficulty is that it has withstood four years of challenges, including multiple Kaggle competitions.

In the following sections, we consider ARC-AGI from a neural perspective and present a mathematical framework for it. To the best of the authors' knowledge, no such work has been done so far. When sections are inspired by others, proper references will be provided.

4.1.1 Framework Introduction: Neural-Based Approach

Definition 4.6 (Puzzle)**.** *A puzzle is a pair of input-output 2D grids of size 1x1 to 30x30, each grid filled with one color of the 10 colors.*

Definition 4.7 (Task)**.** *A (solved) task is a set of puzzles composed of training and test split,*

- *Training pairs* $\{(x_k, y_k)\}_{k=1}^{K}$, $K \in \{2, \dots, 7\}$.
- *Test pairs* $\{(x_m^{test}, y_m^{test})\}_{m=1}^{M}$, $M \in \{1, 2, 3\}$.

Notation 4.1. *Henceforth, we denote* $S := \{\boldsymbol{x}^{train}, \boldsymbol{y}^{train}, \boldsymbol{x}^{test}, \boldsymbol{y}^{test}\}$ *for a solved task and* $T := \{\boldsymbol{x}^{train}, \boldsymbol{y}^{train}, \boldsymbol{x}^{test}\}$ *for an unsolved one, and* $\mathcal{D}^{training} := \{S_1, \dots, S_{400}\}$ *for the training data set and* $\mathcal{D}^{test} := \{T_{401}, \dots, T_{800}\}$ *for the testing data set.*

The problem can then be formulated as follows.

Problem 4.1 (General Problem)**.** *Can we build a neural model* f_θ *trained on* $\mathcal{D}^{training}$ *capable of predicting* $\boldsymbol{y}^{test}$ *for each* $T \in \mathcal{D}^{test}$*? More formally, our aim is to find* f_θ *such that,*

$$f_\theta(T_j|\mathcal{D}^{training}) = f_\theta(\boldsymbol{x}_j^{test}|T_j, \mathcal{D}^{training}) = \boldsymbol{y}_j^{test},\ j = 401, \dots, 800.$$

Remark 4.1. *Observe that since the input-output pairs do not necessarily share the same size,* f_θ *has to successfully output the grid dimension, the pattern, and the colors.*

Algorithm 1 Method for Humans to Solve ARC Tasks

Data: $T = \{\mathbf{x}^{\text{train}}, \mathbf{y}^{\text{train}}, \mathbf{x}^{\text{test}}\}$
Result: $\mathbf{y}^{\text{test}}$
while *no valid hypothesis has been found* **do**
  Randomly select a training pair (x_k, y_k)
  Compare y_k with x_k and formulate 1 to 3 simple hypotheses explaining the transformation
  Test each hypothesis on all other pairs (x_j, y_j), $j = 1, \dots, K$
  if *a valid hypothesis is found* **then**
    Retain the simplest hypothesis that fits all pairs **break**
  end
end
foreach *pair* $(x_k, y_k) \in \mathbf{x}^{train}$ **do**
  Collect any additional information required to apply the transformation function
end
Apply the retained transformation function to $\mathbf{x}^{\text{test}}$
return $\mathbf{y}^{\text{test}}$

In Section 4.3, we introduce a more refined framework, which shows that to solve tasks, we unconsciously adapt our vocabulary to formulate relevant hypotheses, as no universal natural language description explains each task concisely.

[8] We will restrict the analysis to the public evaluation set.

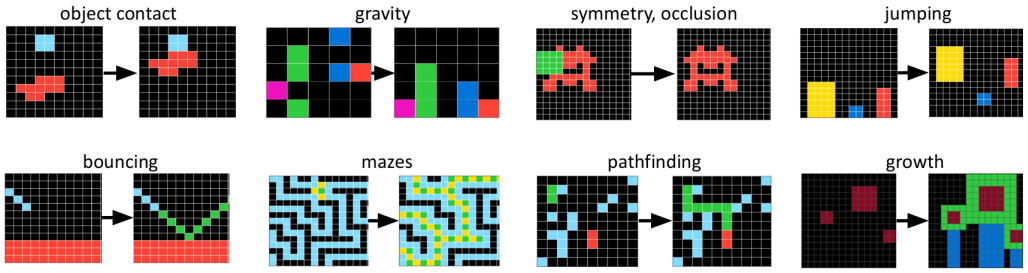


Figure 7: Single pairs (x_k, y_k) of eight different (simple) tasks associated with human intuitively interpretable description of the solution.

4.1.2 Intuitive Understanding of the Problem

At first glance, the resolution of the tasks in Figure 7 seems quite straightforward. Why is it needed to have more than one puzzle per task to generalize? Why is the algorithm proposed for humans to solve a task that complex?

To answer the previous questions, consider the task T_{402} depicted in Figure 8. We will show that some tasks require strictly more than one example to completely describe the hypothesis without any equivocal.

Let us follow the algorithm; without loss of generality, we compare y_1 with x_1. We observe two objects in x_1 and only one in y_1. The single one in y_1 is the same as the one in x_1, except that it changed color (from light blue to red). After multiple passes on all the puzzles, we would get an incrementally more abstract and conceptual formulation of the hypothesis. Let us assume that H_1 is the hypothesis formulated by observing only the first puzzle. When considering other puzzles, we would refine it as follows.

- H_1: Two objects remove the small one and color the other red.
- H_2: Two objects, a blue one indicating the change of color, and a light blue one on which we apply the change.
- H_3: Two objects, a blue one indicating the change of color, and a light blue one on which we apply the change. The plus means red; the flower is green; the helmet is orange.

Remark 4.2. *Formulating such hypotheses for a computer is not an easy matter as it needs to have a flawless understanding of the priors, the notion of object, counting, and topology. Even if it had, formulating such a hypothesis is not trivial as the computer needs to attend to specific parts of certain puzzles in a nonsequential manner to correct and improve its hypothesis.*

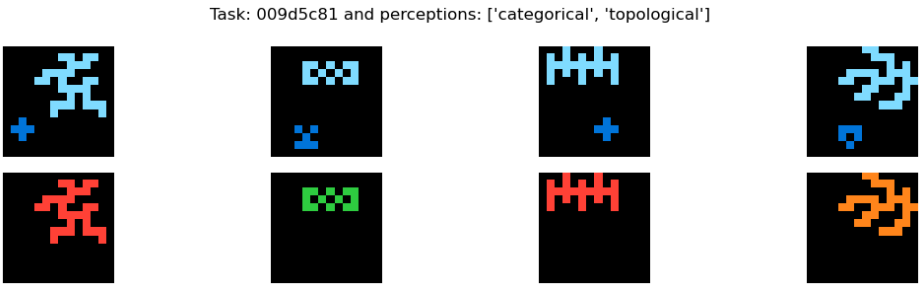


Figure 8: Task: T_{402} which relies on "object detection" and "rule inferring".

Remark 4.3. *Another issue is that natural language lacks a simple and explicit vocabulary to describe the dynamic at play. In Figure 7, finding a clear and simple description is straightforward, while in Figure 8, it is much harder without defining a new vocabulary.*

The task T_{402} is fairly easy as $\{\mathbf{x}^{train}, x^{test}\}$ always has two objects, each of the same color, one at the bottom indicating the change in color and one at the top right to whom we apply the change. This particular sample allows each of the following hypotheses to work.

- K_1: *(Most Abstract)* In x, observe multiple objects, where there is a blue one of a particular shape indicating the change of color to apply all the light blue objects. For y, delete this object and change the light blue pixel according to the following rule. The cross means red, the flower is green, and the helmet is orange.
- K_2: *(Restriction to two objects)* Two objects, a blue indicating the change in color, and a light blue to which we apply the change. The plus means red; the flower is green; the helmet is orange.
- K_3: *(Restriction to the observed spatial configuration)* Find an object on the bottom left that indicates the color (the plus means red; the flower is green; the helmet is orange). Delete it and turn the non-black pixel in the color mentioned.

For this particular x^{test}, each of the previous hypotheses K_1, K_2, K_3 would correctly produce the desired y^{test}. But if x^{test} had more than two objects, even just another separated non-black pixel, the model would not generalize under K_2. The same applies to K_3 if there was a rotation of x^{test}.

Remark 4.4. *Therefore, our goal is to construct a model capable of expressing or understanding the task in the most abstract and conceptual way to not depend on the form of x^{test}.*

Definition 4.8 (Occam's Razor)**.** *It is the problem-solving principle that recommends searching for explanations constructed with the smallest possible set of elements. Or, the principle that the simplest explanation is usually the best. In practice, its implementation is rather tricky, as one has to define what is "simple".*

Example 4.1. *Given the sequence $1, 2, 4, \ldots$, how many terms must we observe to infer its general rule? For example, it could follow $1, 2, 4, 8, \ldots$ with $S_n = 2^{n-1}$ or $1, 2, 4, 7, \ldots$ with $S_{n+1} = 2S_n - S_{n-1} + 1, S_0 = 1$. Our ability to deduce the rule depends on Occam's razor: Given prior experience, we are inclined to favor the simpler explanation, such as $S_n = 2^{n-1}$.*

To solve tasks, we rely on Occam's Razor. However, its application is inherently subjective, as the notion of "simplicity" depends on consensus and context.

4.1.3 Formal Theoretical Problem

Definition 4.9 (Out of Distribution)**.** *We say that the (testing) data is out of Distribution (OOD) if it is entirely different from the training data.*

Definition 4.10 (K-Shot Learning)**.** *K-shot learning refers simultaneously to the ability of a model to generalize accurately from a training data set composed of a maximum of K examples. Or to the data set, which is composed of a maximum of K examples.*

Definition 4.11 (Zero-Shot Learning)**.** *A Zero-Shot Learning (ZSL) problem can be seen as an extreme case of a K-Shot Learning problem, where $K = 0$. ZSL assumes that each instance is so unique or different that it can be treated as one of a kind, requiring the model to generalize purely based on auxiliary information, such as descriptions or semantic relationships.*

Hypothesis 4.1. *The construction and resolution of ARC-AGI is based on the following assumptions.*

I. Each task is designed with the minimal number of puzzle pairs necessary for humans to grasp the underlying logic and successfully derive $\boldsymbol{y}^{test}$ from $\boldsymbol{x}^{test}$.

II. Each task is unequivocal, meaning that there exists only one $\boldsymbol{y}^{test}$ given $\{\boldsymbol{x}^{train}, \boldsymbol{y}^{train}, \boldsymbol{x}^{test}\}$.

III. Each task is entirely different from the others.

IV. Solving the tasks requires a perfect understanding of priors.

To solve the tasks, we have to rely on the underlying logic exhibited by the puzzles, which we have previously named hypothesis and will refer to as follows.

Definition 4.12 (Logic function)**.** *Let $T = \{\boldsymbol{x}^{train}, \boldsymbol{y}^{train}, \boldsymbol{x}^{test}\}$ be a task. We call the function φ_T that satisfies $\varphi_T(x) = y$ for any pair $(x, y) \in \{\boldsymbol{x}^{train}, \boldsymbol{y}^{train}\}$ the logic function of the task.*

$$x_k \xrightarrow{\varphi_T} y_k \quad k \in \{1, \ldots, K\}$$

$$x_m^{test} \xrightarrow{\varphi_T} y_m^{test} \quad m \in \{1, \ldots, M\}$$

Remark 4.5. *As shown, the logic function can have multiple types of representation. It can be a function of the input to the output grid, a natural language description (hypothesis), some code in Python, or a Domain Specific Language (DSL).*

Lemma 4.1. *The hypotheses 4.1 enable us to have the following proprieties,*

I. ARC-AGI is a ZSL Problem.[9]

II. The input grids of each puzzle in a task have to be sampled from a specific kind of distribution, that is, $\boldsymbol{x}_T^{train}, \boldsymbol{x}_T^{test} \sim \mathcal{D}_T$.

III. There is a theoretical[10] *equivalence between a task* T *and the pair composed of the sampling distribution of the task and the logic function, that is,* $T \approx (\mathcal{D}_T, \varphi_T)$.

Notation 4.2. *From now on,* $\mathcal{D}_T$ *denotes the distribution from which the input grids are sampled for task T, and we will refer to it by the sampling distribution of the task.*

Hodel 2024 and Li et al. 2024 used this equivalence to expand the data set. They called the task distribution the generator and the logic function the verifier. We will explain it in greater detail later.

4.1.4 Inductive vs. Transductive Reasoning

"When solving a problem of interest, do not solve a more general problem as an intermediate step. Try to get the answer you really need, but not a more general one." - Vladimir Vapnik

Until now, we have not made explicit the output of our neural model f_θ, in the sense that we have not yet chosen f_θ to be an inductive or a transductive model. Li et al. 2024 compared the two types of model and found that they are unexpectedly complementary processes.

Definition 4.13 (Transduction)**.** *Transduction, or transductive reasoning, refers to the process of reasoning from observed training examples to make inferences about specific test examples.*

Definition 4.14 (Induction)**.** *Induction or inductive reasoning is a method of reasoning in which broad generalizations are derived from observed training cases, which are then applied to the test cases.*

Transduction was introduced by Vladimir Vapnik in the 1990s. According to him, transduction is preferred to induction because it does not require us to solve a more general problem (inferring a function) before solving a more specific one. An example that goes against his argument is the following.

Example 4.2. *When guessing the value of the sequence* $S_n = 1 + 2 + \ldots + n$*, if n is small, directly answering is more prone to success than trying to infer a general function to solve the sequence. But when n becomes larger, the task becomes convoluted, and we risk computing and memory errors, which is where inductive reasoning is better since we only have to compute* $S_n = n(n+1)/2$.

Definition 4.15 (Transductive Model)**.** *A neural transductive model of our problem is a function*

$$t_\theta : \mathcal{X}^K \times \mathcal{Y}^K \times \mathcal{X}^M \to \Delta(\mathcal{Y}), \quad t_\theta(\boldsymbol{x}^{test} | T, \mathcal{D}^{training}) = \boldsymbol{y}^{training}.$$

Where $\Delta(\mathcal{Y})$ *is the set of distributions in* $\mathcal{Y}$ *Li et al. 2024.*

Definition 4.16 (Inductive Model)**.** *A neural inductive model of our problem is a function*

$$i_\theta : \mathcal{X}^K \times \mathcal{Y}^K \times \mathcal{X}^M \to \Delta(\mathcal{X} \times \mathcal{Y}), \quad i_\theta(T | \mathcal{D}^{training}) = \varphi_T.$$

Where $\Delta(\mathcal{X} \times \mathcal{Y})$ *is the set of distributions of functions going from* $\mathcal{X}$ *to* $\mathcal{Y}$*, and* $\varphi_T(x^{train}) = y^{train}$ *Li et al. 2024.*

[9] We will later show that, although each task is different, the process of solving tasks is a transferable skill. Therefore to generalize, the model must learn to logic functions of previously seen tasks to solve for the unsolved ones.

[10] Under the assumption that someone has the generalization prowess of finding it.

4.1.5 Conditions for Effective Generalization

ARC-AGI is a ZSL problem as each task is different, and they are implemented with just enough examples to obtain unequivocally the logic function according to Occam's Razor principle.

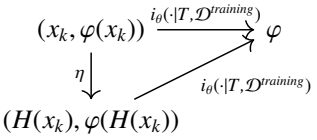
Definition 4.17 (Distribution Invariant)**.** *We say that a function f is $\mathcal{D}$-distribution invariant if f is constant over all the points sampled from the distribution $\mathcal{D}$. Let $x, y \sim \mathcal{D}$, then $f(x) = f(y)$.*

Definition 4.18 (Measure preserving)**.** *Let $\mathcal{D}$ be a probability distribution over $\mathcal{X}$. A function $H : \mathcal{X} \to \mathcal{X}$ is said to be measure-preserving w.r.t. $\mathcal{D}$. If for any $x \sim \mathcal{D}$, $H(x) \sim \mathcal{D}$. More formally, let μ be the probability measure associated with the distribution $\mathcal{D}$. Then,*

$$\mu(H^{-1}(A)) = \mu(A) \quad \textit{for all measurable sets } A \subset \mathcal{X}.$$

Theorem 4.1 (Prerequisite for generalization)**.** *A necessary condition generalization is the ability of the (inductive) model f_θ to understand and express each task in an abstract and conceptual manner. The model must master a language that incorporates priors, whether this language is our natural language of topology and transformations or a domain-specific language, essentially a more convenient way to express the dynamics at play.*[11]

I. The (inductive) model has to be task distribution invariant.[12] *In the sense that for any measure preserving function H w.r.t $\mathcal{D}_T$ the following graph commutes.*



The question is then whether such measure-preserving functions exist. In practice, we do not need the function H to be strictly "measure-preserving", and we cannot guarantee that it is. Since we only observe a sample of $\mathcal{D}_T$, we have only an approximation of the distribution and cannot fully assess the likelihood of observing a particular situation. For example, suppose that a task's logic relies on color comparison. From the observed sample, we may not know how likely it is to encounter yellow versus red, but we do know that there must be at least two distinct colors.

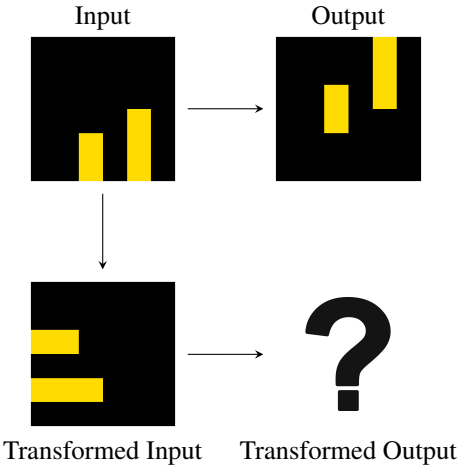


Figure 9: Transformation are not always measure preserving.

Example 4.3. *Rotation is not always measure-preserving, as shown in Figure 9. Since the logic involves making objects jump by the size of their height, the rotation results in a grid that cannot be generated from the distribution.*

[11] For the transductive model, this entails learning a geometric and algebraic representation of concepts within its layers, which should manifest in the embedding space.

[12] We cannot write such a condition for the transductive model as its value changes on each sample of the distribution, but it must learn a conceptual invariance similar to the one of the inductive model.

As a general guideline, the function H should not remove information. For example, a function that adds or removes an object is unlikely to be measure-preserving, as it generates a point that cannot be sampled from the distribution $\mathcal{D}_T$. In contrast, transformations such as rotations preserve the notions of object, number, and topology and therefore have more chance to be.[13]

4.1.6 Enhancing Prior Knowledge (Boosting Prior)

According to Theorem 4.1, we aim for our model to be invariant to the sampling of $\mathbf{x}^{test}$ and to possess a refined understanding of the priors. To achieve this, we augment the training data set and fine-tune the model on this augmented data set.
In Section 4.2, we will introduce the sets of data and the task generator based on the equivalence of Lemma 4.1. The two respective task generators were implemented by Hodel 2024 and Li et al. 2024. They stochastically generate new tasks composed of puzzles $(x, \varphi_T(x))$ where $x \sim D_T$.

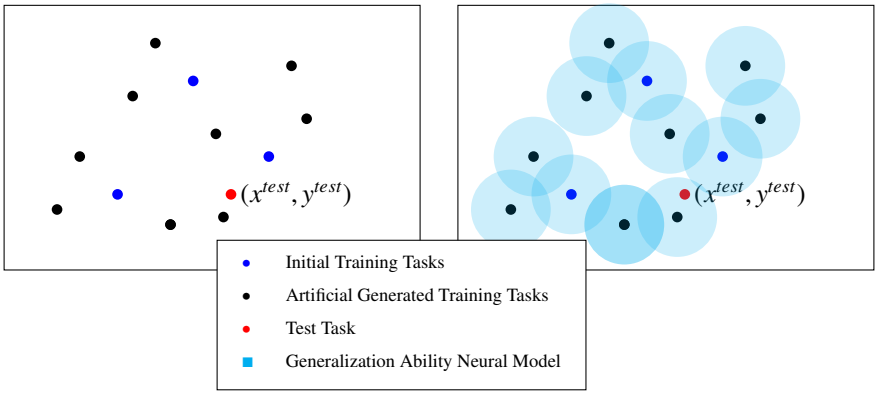
There is also a third way to augment data, which is best explained in Akyürek et al. 2024.

Theorem 4.2. *We can generate three types of new tasks with the transformations H from the list in Table 2.*

- *Input grids only,* $(x, y) \mapsto (H(x), y)$.
- *Output grids only,* $(x, y) \mapsto (x, H(y))$
- *Both input and output,* $(x, y) \mapsto (H(x), H(y))$.

Remark 4.6. *This paradigm optimizes the model's prior distribution over the set of solutions. It involves augmenting the training data through transformations, increasing the likelihood that the model can conceptually understand the tasks.*

Representation of the expansion of the reasoning domain of our neural model by adding new tasks

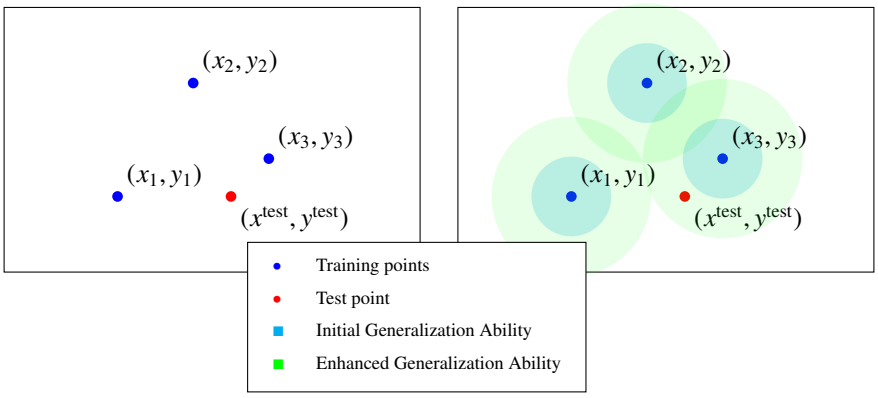


In Section 3.2 we have shown that LLMs possess a non-zero OOD generalization, meaning that they are likely to produce valid responses even on unseen data if it is not significantly "different" from the training data. Consequently, increasing the volume and diversity of the training data will expand the coverage of the generalization space of the model. The idea is that our neural model f_θ will cover more space when trained on the augmented data set, and if it is trained on a sufficient amount of data, we might be lucky to have $\{x^{test}, y^{test}\}$ fall within the reasoning range of our model as in Sketch.

This representation exists under the assumption that there is a successful classification neural model g_θ that offers an optimal embedding space and a metric function. The model g_θ takes for input the pair $(x, y) := (\{x_k, y_k\}, \lambda)$, where λ could be the name of the task or a natural language description of the logic. Due to imperfections in the derived metric, these regions will not form perfect geometric balls, and generalization might also occur in areas far from the original points.

[13] This subsection is more theoretical, as in practice we do not know the distribution $\mathcal{D}_T$.

Sketch 1: Representation of the model's generalization domain expansion for a testing task



Transformation	Description	Free Parameters
Rotations	Rotate the image by a specific angle. Common angles are **90**°, **180**°, and **270**°.	Angle: {90°, 180°, 270°}
Flip	Flip the image horizontally (left-right) or vertically (top-bottom).	Direction: {horizontal, vertical}
Transposition	Exchange rows with columns, reflecting the image along its diagonal.	None
Homotopy	Stretch or shrink the image horizontally, vertically, or both. This changes the aspect ratio.	Stretch factors: {horizontal, vertical}
Patching	Replace a random set of pixels with a specific color or pattern. Simulates occlusion or noise.	Color: Any, Area: {size, position}
Color Permutation	Rearrange the colors of the grid by permuting them in all possible orders.	Permutation Order: Any
Image Padding	Add borders to the image to increase its dimensions. Padding can be zero-filled or randomly filled.	Padding size: {width, height}, Fill: {zero, random}
Swap Information Order	Swap training and testing datasets, interchanging their roles.	Permutation
Composition	Combine any of the transformations above into a sequence for more complex effects.	Transformations

Table 2: Non exhaustive list of Transformations to augment data set. Akyürek et al. 2024.

4.1.7 Improving Inference Capabilities (Boosting Inference)

Definition 4.19 ((Left) Group Action)**.** *Let $(G, \cdot)$ be a group with the identity element e and let X be a set. Then a (left) group action α of G on X is a function*

$$\alpha : G \times X \to X, (g, x) \mapsto \alpha(g, x) := \alpha_g(x).$$

That satisfies, $\alpha(e, x) = x$ and $\alpha_g(\alpha_h(x)) = \alpha_{g \cdot h}(x)$ for all $g, h \in G$, $x \in X$.

Before the invention of convolution neural networks, classification neural models might fail to classify a rotation of a training image, as presented in Figure 10 Weiler et al. 2023. Developing such an architecture for LLMs is much more challenging, as the list of measure-preserving functions to which the model must be invariant is context-dependent.

Given the particularly low number of data points per task, adding even just one puzzle would be extremely beneficial. Sadly, at test time, we cannot use what has been done before as, in practice, the task T equivalent representation $(\mathcal{D}_T, \varphi_T)$ is unknown, and finding $\mathcal{D}_T$ is equivalent to finding φ_T!

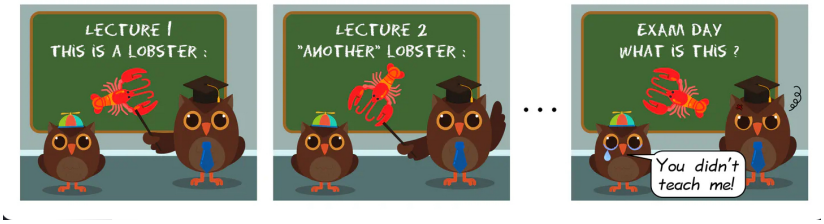


Figure 10: Metaphor of a neural classification model without convolutions.

If our observation is only restricted to the task, the only pairs that we can artificially add that are sampled from the task distribution are the ones we obtain via the measure-preserving maps H making the diagram shown in Figure 11 commute. Otherwise, we have to find the output of $H(x_k)$, and that is exactly what we are trying to do.

$$\begin{array}{ccc} x_k & \xrightarrow{\varphi} & y_k \\ {\scriptstyle H}\downarrow & & \downarrow{\scriptstyle H} \\ H(x_k) & \xrightarrow{\varphi} & \varphi(H(x_k)) \end{array}$$

Figure 11: Logic function is equivariant to the function H.

Therefore, an artificial puzzle of the task would be of the form $(x'_k, y'_k) := (H(x_k), H(y_k))$ for a function H such that φ_T is equivariant with right to it. Finding such functions for a task is feasible, but not for all, as presented in Figure 12.

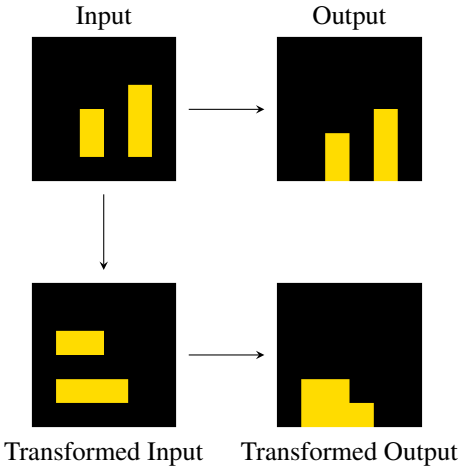


Figure 12: Rotations is not equivariant for all tasks logic functions.

Example 4.4. *It is quite unlikely, but it can happen that certain logic functions are not equivariant with respect to rotations, as shown in Figure 12. The logic function of the task is gravity and the function H is transformation. We can observe that the graph is not commuting as rotating the output does not produce the Transformed Output.*

It is not a lost cause as we can apply an invertible transformation to the whole task in the hope that it will be transformed into a more convenient representation.

Theorem 4.3. *Let T be a task from which we try to find the logic function φ_T. Then, for all invertible functions H (of Table 2), if we solve the alternative task $H(T) := (T(\boldsymbol{x}^{train}), T(\boldsymbol{y}^{train}), T(\boldsymbol{x}^{test}), T(\boldsymbol{y}^{test}))$. We can obtain the logic function of the original task.*

$$\varphi_T = H^{-1} \circ \varphi_{H(T)} \circ H.$$

$$\begin{array}{ccc} H(\mathbf{x}^{\text{train}}) & \xrightarrow{\varphi_{H(T)}} & H(\mathbf{y}^{\text{train}}) \\ {\scriptstyle H}\uparrow & & \uparrow{\scriptstyle H} \\ \mathbf{x}^{\text{train}} & \xrightarrow{\ ?\ } & \mathbf{y}^{\text{train}} \end{array}$$

Figure 13: Commutative diagram illustrating the transformations of training and testing datasets.

Example 4.5. *In Figure 14, we observe an example of the dynamics described in Theorem 4.3. At the bottom, the logical function is "gravity to the left," which the model might not have encountered during training. In contrast, the rotated task, which is "gravity," is a concept that the model already knows. Therefore, if the model can solve the rotated task, with our help, we can make it solve the initial one.*

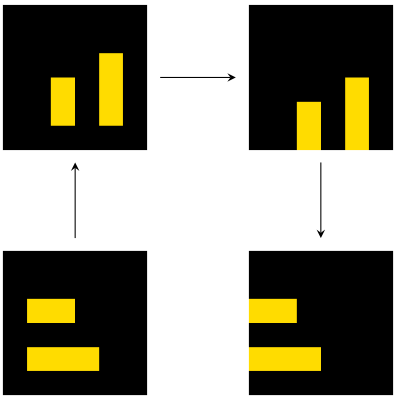


Figure 14: Demonstration that transforming a task also transform its logic function.

4.2 State of the Art

4.2.1 Competition History

In **2019**, Chollet introduced ARC-AGI François Chollet 2019, stating that it would not be easily defeated.

In **2020**, the first competition was held in Kaggle. The winning team was "Ice Cuber" with a score of 21% on the private test set. Like others in the competition, their approach used brute-force discrete program search with a Domain-Specific Language (DSL). This involved searching through a massive program space step-by-step, exploring the possible compositions of DSL primitives to find programs that correctly transformed the input grids into their corresponding output grids.

In **2022**, Chollet and Lab42 collaborated to host ARCathon 2022, the first global AI competition aimed at beating ARC-AGI. The winner, Michael Hodel 2022, developed one of the most advanced ARC-AGI DSLs to date (Francois Chollet et al. 2024).[14]

In **2023**, the competition continued with ARCathon 2023. The first place was shared between Team SM (Somayyeh Gholami Mehran Kazeminia) and Team MindsAI (Jack Cole), both of whom achieved 30% precision in the private evaluation set.

In **2024**, the competition was again held on Kaggle. LLMs were used instead of the traditional DSL-based program search for the first time. The lack of data and frozen ability at inference time held back the implementation of LLMs, i.e., they ceased learning after being fine-tuned. Chollet mentioned that models should continue learning during inference as tasks increase in difficulty *Artificial General Intelligence Conference 2024* 2024. The MindsAI team, composed of Jack Cole, Michael Hodel, and Mohammed Osman, solved both problems.

Hodel 2024 created the first generators of examples written in his DSL. The generated examples are ranked with a score between 0 and 1, representing the difficulty of the puzzle based on the size of the grid and the number of objects. Each generator can produce up to 10000 unique examples. Inspired by this work, Li et al. 2024 introduced another data set with its own generators and solvers in Python, assuming that Hodel's DSL might lack Turing completeness. Their idea was to first write an alternative representation of 160 training tasks in a Python file (generator and solver) that they call seed and mutate them with ChatGPT4.*o* to obtain 200k seeds.

Jack Cole 'solved'[15] the frozen-time inference problem with his paradigm, "Augmented Inverse Retrieval Voting". All teams using LLM transductive models obtained a decent score[16] implemented their interpretation "AIRV" as the MindsAI paradigm never went public. MindsAI scored 55.5% but did not obtain a rank in the competition as they chose not to disclose their ideas to maintain their advantage next year. The winners were ARChitects, scoring 53.5%; the second was G. Barbadillo, who achieved 40%, and the third was Aliajs, who also achieved 40%. At the beginning of the competition, Ryan Greenblatt achieved a notable implementation with a score of 42% on the ARC-AGI-Pub leader board using an LLM-guided program synthesis approach, which iteratively debugs the most promising one to find a successful one Greenblatt 2024. For a deeper dive into the technical report Francois Chollet et al. 2024.

The main resources are

- **Domain-specific language**: ARC-dsl Hodel 2022.
- **Task Distribution (Generator)**: ReARC Hodel 2024, BARC Li et al. 2024.
- **Simpler Benchmarks**: ConceptARC Moskvichev, Odouard, and Mitchell 2023, MiniARC Park et al. 2023, 1DARC Xu et al. 2024, MC-LARC Shin et al. 2024.
- **Interface**: The ARC Game Volotat 2024, H-ARC LeGris et al. 2024.

[14] In 2024, Chollet observed that by combining all correct solutions from DSL, a machine could achieve 49% precision.

[15] Not exaclty as explained in Theorem 4.3 and Remark 4.11.

[16] In comparison, a basic LLM prompt scored only 5%, while a fine-tuned state-of-the-art (SOTA) LLM scored around 10%.

In the next subsections, we will compare program-synthesis based methods with neural network-based methods. The general comparison can be summarized by the following table from *Artificial General Intelligence Conference 2024* 2024.

	Modern Machine Learning	**Discrete Program Search**
Model	Differentiable parametric function	Graphs of operators from DSL
Learning Engine	SGD	Combinatorial search
Feedback Signal	Loss Function	Correctness check
Key Challenge	Data (dense sampling problem)	Combinatorial explosion

Table 3: Comparison two main paradigms for ARC-AGI.

4.2.2 Program-Synthesis-Based Methods (Inductive)

Definition 4.20 (Domain Specific Language)**.** *A domain specific language (DSL) is a computer language created for a specific application domain.*

Definition 4.21 (Program synthesis)**.** *Program synthesis is the task of automatically generating programs that satisfy a given set of requirements.*

A program is said to be automatic if it generates programs without direct human specifications. The set of all possible programs the system can consider is usually defined by a DSL.

Remark 4.7. *Program synthesis highly depends on the initial DSL, which restricts adaptability and generalization.*

Definition 4.22 (Inductive Program Synthesis)**.** *If the specifications are provided through input-output examples, it is called Inductive Program Synthesis.*

ARC-AGI can be considered as an inductive program synthesis problem when jointed by a DSL. Some notable examples are neurally guided program induction methods with LLM Ouellette 2024, or program sampling with hindsight relabeling Butt et al. 2024, or inductive logic programming (ILP) to break down complex functional tasks into simpler relational synthesis subtasks Hocquette and Cropper 2024.

We will first quickly present some theoretical requirements that the DSL must satisfy and then introduce DreamCoder Ellis et al. 2020, a model capable of building and refining its own DSL.

Definition 4.23 (Turing complete in expressivity)**.** *We say that a DSL is Turing Complete in expressivity if there exists an integer M, such that for any puzzle (x_k, y_k) belonging to any task T, there exists a sequence of primitives $\{\varphi^{\psi(1)}, \dots, \varphi^{\psi(M)}\}$ such that,*

$$\bigcirc_{i=1}^{M} \varphi^{\psi(i)}(x_k) = y_k.$$

Where $\psi : \{1, \dots, M\} \mapsto \{1, \dots, Q\}$ is a function and Q is the size of the set of primitives.

Remark 4.8. *In the previous definition, $\bigcirc$ denotes the composition operator of primitives, which must be understood as functions that call each other sequentially in a code.*

To be independent of the sampling of D_T, we need the following property.

Lemma 4.2. *The language must be task distribution invariant. In the sense that for all tasks T, there exists a sequence of primitives $\{\varphi^{\psi(1)}, \dots, \varphi^{\psi(M)}\}$ satisfying **for all** $(x_k, y_k) \in T$.*

$$\bigcirc_{i=1}^{M} \varphi^{\psi(i)}(x_k) = y_k.$$

Theorem 4.4 (Sequential representation)**.** *Under the following hypothesis,*

I. The Domain Specific Language is Turing complete in expressivity.

II. The Domain Specific Language is distribution invariant.

For any task T, its logic function φ_T has the following sequential representation.

$$\varphi_T \approx \bigcirc_{i=1}^{M} \varphi^{\psi(i)} := (\varphi^{\psi(1)}, \ldots, \varphi^{\psi(M)}) = (\psi(1), \ldots, \psi(n)).$$

Remark 4.9. *Without going into details, we must also define an efficient search policy. For example, Hodel's DSL is made up of 165 primitives and requires up to 30 of them to solve certain training tasks. A simple greedy algorithm would have to search for up to $165^{30} \approx 2^{67}$ combinations.*

As explained in Section 4.1, for a model to perform it has to generalize across domains (OOD) and perform compositional and hierarchical reasoning. This is why DreamCoder appears to be built for ARC-AGI due to its capacity for continual refinement and expansion of its DSL, enabling it to improve and adapt over time, and that a frozen set of primitives seems to be doomed to fail.
However, it has not yet shown promising results due to the complexity. Its first implementation was performed by Alford et al. 2021 with a search for a neurally guided bidirectional program, but to no avail. The current SoTA method uses Abstraction & Reasoning Language (PeARL), a pure functional language for DreamCoder to reason, and it was able to solve three times more tasks than in the previous work Bober-Irizar and Banerjee 2024.

Definition 4.24 (DreamCoder)**.** *DreamCoder is based on a wake-sleep algorithm. It alternates between a wake and a sleep phase to continuously improve its ability to solve tasks and refine its underlying knowledge representation. In the wake phase, it finds and generates primitives to solve tasks. In the sleep phase, it first compresses the library of primitives found in the wake phase and refactors them, and then it trains a neural network for probabilistic search with the new library.*

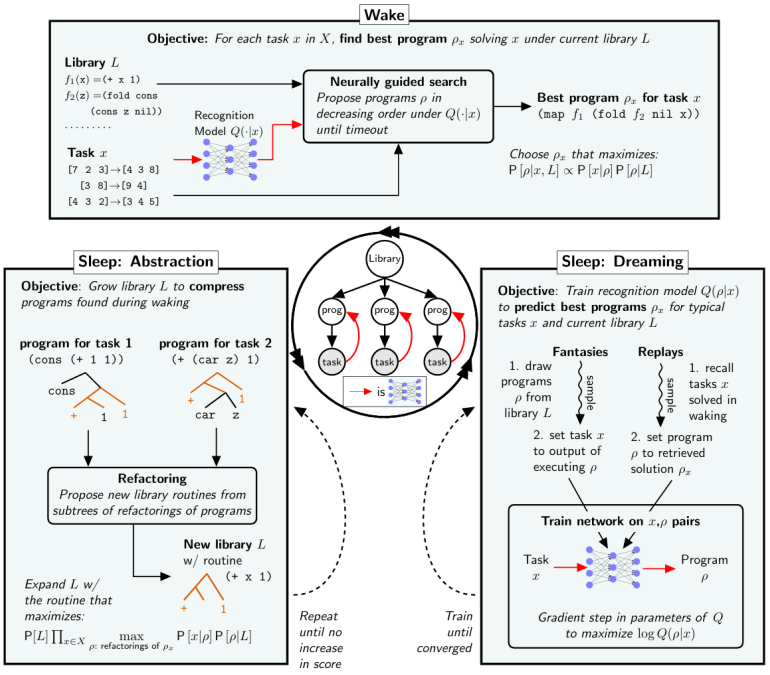


Figure 15: DreamCoder's basic algorithmic cycle, which serves to perform approximate Bayesian inference for the graphical model diagrammed in the middle.

4.2.3 Transductive Methods (Neural-Based)

As the in-context learning abilities of LLMs are minimal on ARC-AGI the neural transductive pipeline rests on a combination of multiple fine-tunings. A representation of the pipeline is shown in the picture 16. In summary, we first need to expand the training data set to have better prior over the set of solutions. The three best methods to expand the initial data set are ReARC Hodel 2024, the mutated seeds of BARC Li et al. 2024, or the generation of new tasks from one given by a set of transformations Akyürek et al. 2024.[17]

Since the test data set is OOD and the transformers have a limited generalization range, we have to fine-tune the model once again, but this time on the private ARC Dataset. It is more challenging than the previous one, as we do not know the logic function of the task and must expand the data set only using the transformation functions H of the table 2 (and not the generator). In the last step, we again increase the number of tasks using invertible functions such as in Theorem 4.3 and implement a voting procedure with score functions to choose the final prediction.

The ARChitects Franzen et al. 2024, Barbadillo Barbadillo 2024, MARC Akyürek et al. 2024 and BARC Li et al. 2024 implemented different versions of the pipeline, which are summarized in the following Table 4.

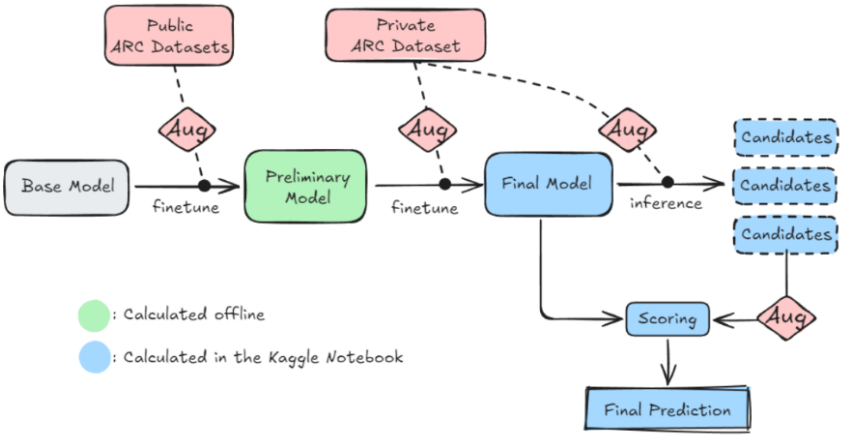


Figure 16: AIRV Pipeline for LLM ARC-AGI 2024 Franzen et al. 2024.

Remark 4.10. *Augmenting the Private ARC Dataset and fine-tuning the model addresses potential biases in the pre-trained model, such as those related to color or vertical and horizontal orientations. This training helps reduce overfitting and promotes a broader set of tasks, making it easier for the model to solve tasks in various formats Barbadillo 2024.*

Remark 4.11. *The augmentation of the tasks via the transformations at inference is similar to Kociemba's Two-Phase Algorithm for solving the Rubik's Cube Kociemba 1992. In Kociemba's algorithm, the cube is first transformed into a reduced state and then solved using a precomputed hashtable for this particular state. Here, however, the form of the reduced state is unknown. Instead, we present to the model a significant number of alternative tasks (by composing the augmentation functions), hoping that some representations will resonate with the model, enabling it to generate the correct solution through a voting mechanism on the proposed solutions, as shown in Figure 14.*

[17] Or any ARC-AGI related data set that can be found on https://github.com/neoneye/arc-dataset-collection/tree/main/dataset.

	The ARChitects	**Barbadillo**	**MARC**	**BARC: Transduction**
Model	eMo-Minitron-8B	Qwen2.5-0.5B-Instruct	8B Llama	8B Llama 3.2
Data sets	ReARC, ConceptARC, ARC-Heavy	ReARC, PQA Dataset, Mini-ARC	ReARC, ARC-Heavy	ARC-Potpourri (ARC-Heavy + ReARC)
Problem Representation	Basic 6	Markdown code snippet with numbers	Numpy arrays digits 0 to 10	Txt with colors for numbers
Number of Tokens	64			
Training Augmentations	Rotations, Transpositions, Permutation Colors, order examples	Applied input, output and both: Rotations, Flips, Padding, Upscale, Mirror, color changes	Rotations, Flips, Reflection, Transpositions, Translation, Increase Width, Increase Width and Dropout	Rotations, Color permutations
Training Loss Function				Cross Entropy
Training Fine-tuning	Two fine-tuning first LoRa: rank 256 then LoRa: rank 64	LoRa: rank 128		full fine-tune
Testing Augmentations	Rotations, Transpositions, Permutation Colors, order examples	Same training	Same training except Dropout	Rotations, Color permutations
Testing Loss Function		Maximized Likelihood, Leave One Out	Sum Cross Entropy Demonstration and synthetic test from LOO[18]	Maximized Likelihood, Leave One Out
Testing Fine-tuning	LoRa: rank 64	LoRa: rank 128 (task Specific)	LoRa: rank 128, MLP, attention, output layers (task Specific)	LoRa: rank 64
Inference Augmentations	Applied to input-ouput: Rotations, Transpositions, Permutation Colors, order examples	Applied to input-output: Invertible transformations	Applied to input-ouput: Invertible Subset Transformations	Applied to input-output: Transpositions, color permutationt
Voting Method	Depth-First Search Sampling with an Aggregated log-softmax scores	Consensus-Voting	Hierarchical voting	Reranking: mean Beam Search and frequency

Table 4: Comparison best Transductive pipelines ARC-2024.

4.3 Summary, Limitations and Future Directions

As the intelligence of a system is the combination of the intelligence of the machine with the one infused by the researchers, the work of the ARC-2024 winners indirectly highlights the limitations of LLMs in OOD generalization. From the discrepancy of the results of the base LLM and the fine-tuned AIRV versions, we can say without a doubt that the work of the researchers greatly contributed to the impressive results.
The question is then whether the LLM had to reason? On the one hand, one could argue that the model indeed exhibited OOD generalization and that the AIRV pipeline only addresses the potential biases in the data. On the other hand, one could also argue that it did not, and the engineered pipeline leverages the hash table capabilities of the current LLM as explained in Remark 4.11.

It begs the following question: What work has to be done in the current architecture to move closer to AGI? To some extent, the existing pipeline, which applies invertible transformations to the input and output of a task to generate alternative representations at inference time, is analogous to providing rotated versions of test images to a nonconvolutional image classification architecture. In image classification, convolutional neural networks were developed to eliminate the need to extend the initial dataset with such transformations or implement them at inference time. As discussed in Section 3.2.1, identifying a transformer architecture suitable for ARC-AGI remains an open question, which we think is a dead end.

From the length of the list of transformations to which the model must be invariant, we argue that it is not feasible and desired because it might not generalize to new tasks. However, the model still needs to abstract information away to generalize. We argue that it can do it via language if it knows when and how to use it.
For example, a quotient representation of a 'grid object' (a set of neighbored grid of another color than the background) can be the natural language word object. This is so because a rotation of a 'grid object' is still an object. Therefore, any (invertible) transformation[19] of the 'grid object' remains an object, and hence is contained in the same equivalence class. Therefore, natural language is a quotient operator that removes abstraction.

From the following example, we demonstrate that we lack an efficient lexicon to describe tasks, which makes their resolution remarkably complex. The complexity comes from the fact that we do not have a simple representation in natural language, which forces us to use context-specific vocabulary and the composition of numerous ideas via words.

Example 4.6. *Consider a 10x10 grid containing black pixels and a 2x3 red rectangle positioned four pixels from the right border and five pixels from the top border. If the logic is to output the number of objects in the grid, then in the description we must specify the presence of an object in the grid. If the task is to output the color, we must specify the color of the object. Similarly, if the task is to determine the width or height of the objects, we must specify their dimensions. If the task requires the distance relative to the left, top, bottom, or right borders, we must also include the spatial position.*

Therefore, a naive, exhaustive description is highly suboptimal. As the task becomes more complex, its length grows exponentially longer. To manage this, we unconsciously abstract certain characteristics and decide to focus on a subset of the information. This abstraction is achieved by comparing the set of puzzles in a task to see what can be abstracted to reduce some degrees of freedom, for instance, by omitting the explicit mention of spatial position when it is not relevant to the task.

This approach results in a quotient representation of the information. For example, describing the object as a "red rectangle" implicitly assumes position invariance and indicates that rotating the object does not alter its description. Through this process, we interact with the task more effectively by prioritizing relevant features and abstracting unnecessary details.

For humans, this process is unconscious and trainable. With enough training, we get a feel for what the logic function of the task should be, even before we think deeply about it. This allows us to translate the puzzles (visual) into a convenient natural language description favorable to human reasoning. It also allows us to put together ideas to formulate the logic function.
As we remain uncertain about how LLMs reason with ARC-AGI and whether they possess both low- and high-level search abilities, we propose a framework in the next section to assist LLMs in performing high-level searches. Therefore, limiting their focus to low-level searches.

[19] rotations, flips, homotopy,...

5 Experiments

A recurrent problem that arises with deep neural models is the incertitude that the model has learned or understood the concept in a generalizable way and not some "shortcuts" (Lapuschkin et al. 2019; Geirhos et al. 2020). The usual method to evaluate model comprehension is OOD testing (Section 3.2). Implementing such a method for ARC-AGI seems challenging as the learning goal is obscure and the data are by construction OOD. Given that LLMs could be the gateway to AGI and we still do not understand exactly how they proceed information, any attempt to improve clarity is worth pursuing.
For ARC-AGI, we affirm that to generalize without shortcuts, the model has to learn 16 distinct human perceptions arising from the four-core knowledge system of Spelke and Kinzler 2007. This list simultaneously serves as the first step-stone to evaluate ARC-AGIs's models' comprehension and to help deep learning guide program search.

5.1 Motivations

This framework's motivations are twofold: to improve benchmark performance and to improve our understanding of LLM reasoning. By improving search efficiency, granularizing tasks and providing deeper insights into LLM reasoning processes, this framework aims to address these goals.

One of the few papers using concepts to make tasks more granular is Li et al. 2024. They initially hand-solved 160 tasks in Python, and at the top of each file they added a list of 1 to 6 concepts. The initial list was of length 147 and grew to 8430 when ChatGPT o1 mutated them, only 7 appeared more than 10,000 times, and 83 appeared more than 1000 times.

Task	**Count**
Color Transformation	31,052
Object Detection	22,492
Rotation	14,447
Symmetry	14,274
Filling	13,888
Color Change	12,092
Color Mapping	10,970

Table 5: Seven most present concepts in BARC 200k data Li et al. 2024.

From the results in Table 5, it seems logical to conclude that we lack an efficient vocabulary to describe tasks. Terms such as 'color transformation,' 'color change,' and 'color mapping' seem to refer to the same concept and provide no additional insight beyond indicating that the logic function is related to color.

We think that it will improve the benchmark based on the observation that scratchpads reduce the chances of getting lost while reasoning toward a direction but do not eliminate the risk of pursuing the wrong one! We hope that the labels will later improve benchmark performance in the following ways:

1. After observing enough examples and solutions, we often acquire an intuition that guides us toward a specific path to the solution without deep deliberation, i.e., acquire type I intelligence. This neural classification model aims to act similarly, that is, to decompose the two types of intelligence. Therefore, it will serve as a high-level search, ultimately reducing search time.

2. Improves the granularity of reasoning, which is crucial for transformer models as discussed in Section 3.2, thus improving accuracy.

We think that it will help us better understand the reasoning of LLMs as the beauty of perceptions lies in their ability to guide attention rather than dictate the form of the logic function as explained in Section 4.3. They inform us where to focus our attention rather than form of the output. The perceptions are not located in x or in y, but in the gap between x and y. This makes it difficult for neural networks to grasp as they are not directly correlated with the puzzle distribution.

The perceptions are not as broad as the priors; otherwise, they will be present in all the tasks and will not infer any information. They are not as narrow as the logic function; otherwise, they will only classify one example due to the Zero-Shot nature of the problem. We chose them over concepts (gravity, maze,...) so that they could be invariant to the transformations described in Theorem 4.2. The concepts are not invariant to the transformations as shown in Figure 12. After applying a 90-degree rotation to the task, is the label 'gravity' still considered 'gravity', or does it become 'gravity to the left'?

5.2 Classification via Human perceptions

The list of perceptions was inspired by the work of Moskvichev, Odouard, and Mitchell 2023. In their paper, they proposed a list of 16 concepts and 10 examples per concept. We started with their list, renamed them with human perceptions, combined some together, and added new ones. Human perceptions are related to Gestalt principles Wertheimer 1923, which aim to explain how we group and discern objects from chaotic stimuli.

Theorem 5.1. *To solve ARC-AGI, a model must acquire an understanding of the four core knowledge present in the list of the 16 concepts.*

1. ***Containment**: Ability to recognize whether an object is inside, enclosed by, or surrounded by another object.*
2. ***Depth**: Ability to determine the relative position of objects, such as whether one object is on top of or behind another.*
3. ***Symmetry**: Ability to recognize balanced and mirrored patterns within objects or scenes and generate symmetrical forms based on axes of reflection, rotation, or translation. The capability to cache objects is included.*
4. ***Categorical**: Ability to identify groups based on shared features, dichotomize space into distinct categories, compare the resulting divisions, and apply group-specific rules or distinctions accordingly.*
5. ***Spatial-Orientation**: Ability to recognize and differentiate directions, particularly horizontal and vertical alignment.*
6. ***Spatial-Ordinal**: Ability to distinguish and relate positions (e.g., higher, lower, left, right) relative to a reference point or directional context.*
7. ***Similarity**: Ability to group or differentiate objects based on shared characteristics such as shape, pattern, or color, and to apply rules or actions accordingly.*
8. ***Quantitative**: Ability to estimate, count, and assess the number of discrete objects or elements in a scene.*
9. ***Replication**: Ability to duplicate, reproduce, or augment patterns and objects, facilitating the creation and extension of similar elements (similar to continuity).*
10. ***Figure-Ground**: Ability to distinguish meaningful forms from background noise by leveraging cues like similarity, enabling the extraction and recognition of salient objects in any environment.*
11. ***Continuity**: Ability to extend or continue shapes, lines, or patterns to the edge of a defined boundary.*
12. ***Size**: Ability to estimate and compare dimensions such as length, breadth, and overall scale of objects or spaces.*
13. ***Closure**: Ability to perceive incomplete shapes, lines, or patterns as enclosed and coherent figures.*
14. ***Centroid**: Ability to identify the central point of an object, including its center of mass and whether its structure is solid or hollow.*
15. ***Topological**: Ability to understand shapes, connectedness, and inherent symmetries.*
16. ***Motion**: Ability to perceive, interpret, and plan movements.*

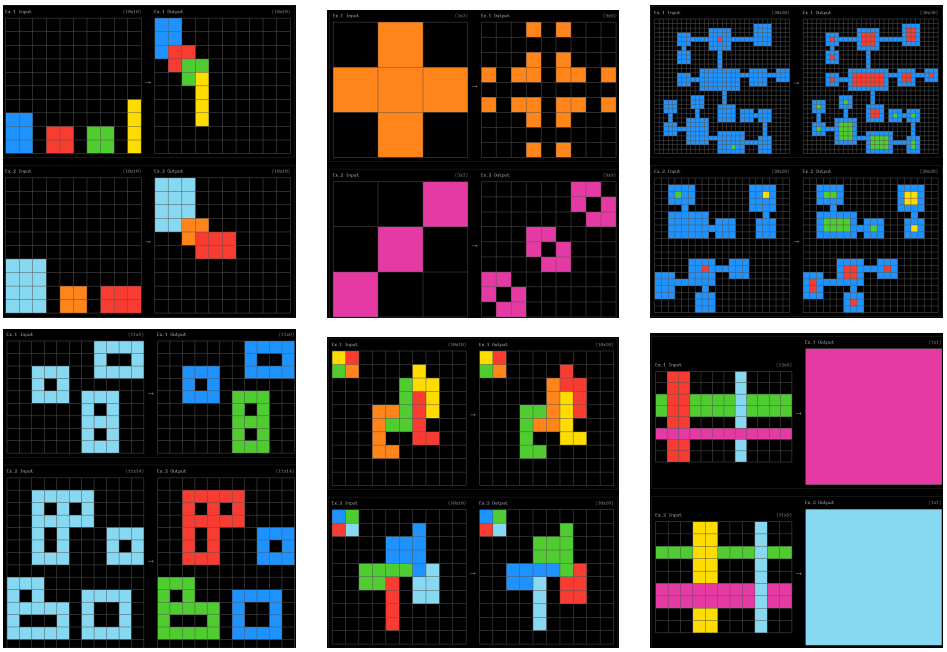


Figure 17: Two puzzles from six different ARC-AGI evaluation tasks.

Example 5.1. *In each of the examples in Figure 17, we illustrate the human perceptions utilized. Top left: Spatial Ordinal, the position of the objects matters in the sequence of motion; Motion, the ability to visualize the movement of objects through space; and Topology, as we glue the bottom-right corner of the objects together. Top middle: Spatial Ordinal and Replication. Top right: Centroid and Topology, as we color the center of the connected objects. Bottom left: Centroid and Quantitative, as we count the number of holes in the object. Bottom middle: Categorical, the top-left cube indicates that we must exchange the colors that are horizontally adjacent. Bottom right: Depth, as we output the object positioned on top of the others.*

5.2.1 Procedure

How did we come up with this list of perceptions? We considered many ARC-AGI tasks and ConceptARC tasksConceptARC tasks in the hope of finding an efficient vocabulary after realizing the lack of it in Table 5. In ConceptARC, we identified better names for the concepts, added descriptions, removed certain concepts, and introduced new ones. We can see two examples of sets of different concepts in Figure 18 with their relabeling. We manually labeled 150 training tasks, which we expanded to 150,000 distinct tasks using the generators from Hodel 2024, along with 50 evaluation tasks. As mentioned above, these tasks are transformation-invariant, allowing us to easily triple the data size by applying three random compositions of functions from Table 2.

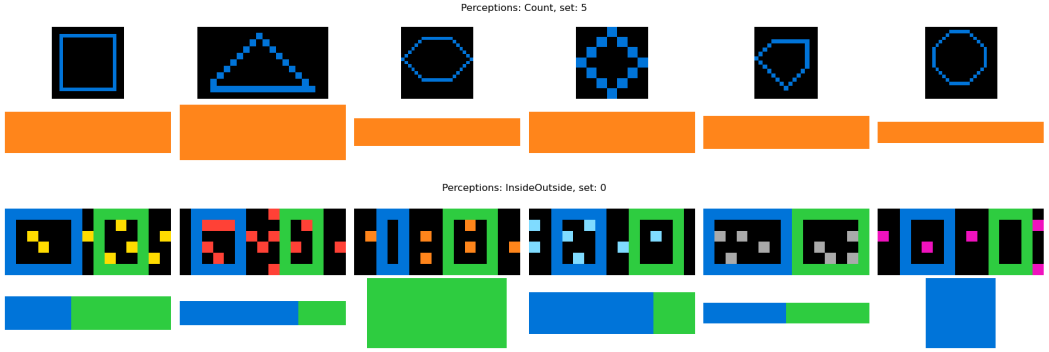


Figure 18: Two different sets of concepts are presented. In the top example, we would relabel it as 'quantitative' and 'topology' since it involves counting the number of edges of the object. In the bottom example, we would relabel it as 'quantitative', 'containment', and 'categorical' as it counts the number of pixels in each square and outputs the number .

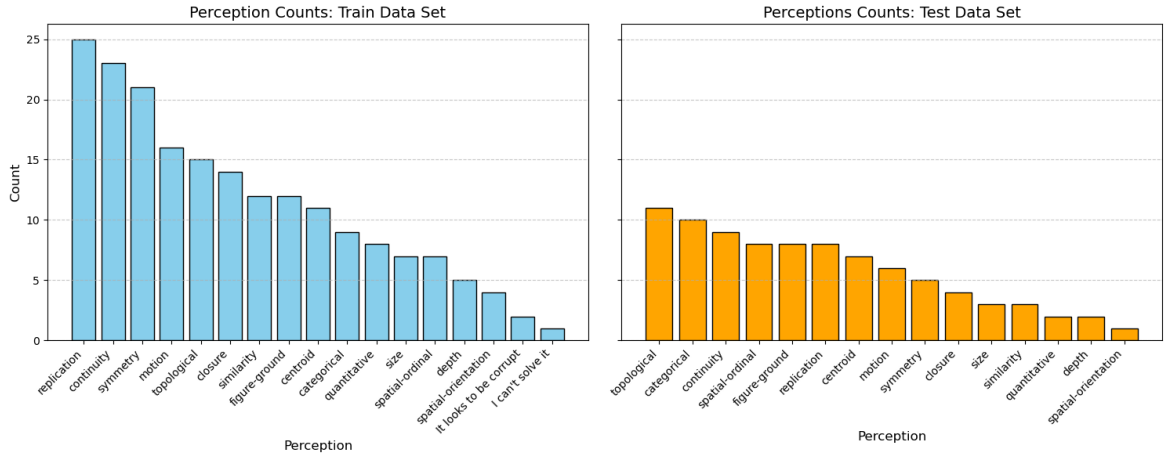


Figure 19: Histogram of Perceptions for the First 150 ARC-AGI Training Tasks and the First 50 ARC-AGI Evaluation Tasks.

5.2.2 Limitations

Each task can have up to three perceptions, resulting in a total of $\binom{17}{3} - 1 = 679$ labels[20]. So far, we have not encountered a task that requires more than three labels. However, if this were the case, we would maintain it at three, as having four is a rare event. If in the private evaluation set it is an event that occurs, it might impact performance.

As shown in Figure 19, some labels may be overrepresented, leading to prediction bias. This issue can be mitigated by generating additional tasks to counterbalance the over-representation using data augmentation techniques.

5.3 Framework

The inductive and transductive BARC model from Li et al. 2024, fine-tuned on the ARC-Potpourri data set (400K training examples), achieved accuracies of 29.125% and 38%, respectively, on the ARC-AGI test dataset (without test time tuning (TTT) and a voting scheme). The inductive model is interpretable as it generates Python code to solve the tasks, which allows us to understand its reasoning process. However, this is not the case for the transductive model, and it is still possible that the inductive model develops an alternative understanding and representation of the task and its underlying logic than ours.
To investigate this, we will compare the base and fine-tuned versions of the Llama 3.2 3B model with the base and fine-tune versions of their inductive and transductive Llama 3.1 8B model from our data set.[21]

5.3.1 LoRA-Shrunk Classification Model (LSCM)

To ensure a fair comparison with the BARC fine-tuned models, we took inspiration from Franzen et al. 2024 in the representation and tokenization of the data as illustrated in Table 6 and Figure 21. Following their approach, we reduce vocabulary size and embedding dimension to adopt a token-efficient paradigm that optimizes input processing[22]. The initial vocabulary size was 128,256, which we reduced to 21 tokens. Using LoRa, we fine-tuned the attention mechanism, feedforward network, embedding layer, and output layer.

Additionally, we opted for a binary representation of the input to prevent it from being influenced by the perception name, ensuring reliance purely on the semantic representation of the tokens. We used Binary Cross-Entropy with Logit Loss for training, predicting the label if the probability exceeded 0.1. The 95% percentile of the tokenized sequence length was 7000. To have more freedom with the architecture we used a Causal Model to which we added a classification head as illustrated in Figure 20.

[20] As the tasks are not always labeled with three perceptions, we have to include the null one.
[21] Due to GPU-Memory constraint we could only use a 3B base model for the base architecture.
[22] In the results, we used the original model, as the shrunk model encountered code-related issues.

Token	Category	Purpose
<|begin_of_text|>	Beginning of Sequence (BOS)	Marks the beginning of a text sequence.
<|end_of_text|>	End of Sequence (EOS)	Marks the end of a text sequence.
[PAD]	Padding Token	Filler for consistent input or output size.
<I>	Input Token	Marks the beginning of the input pair.
<O>	Output Token	Marks the beginning of the output pair.
\n	Newline Token	Separator to preserve the arrays shape.
---	Token Puzzle Separator	Separates puzzles in the dataset.
0-9	Token Color	Encoding 10 colors.

Table 6: List Special Tokens LSCM.

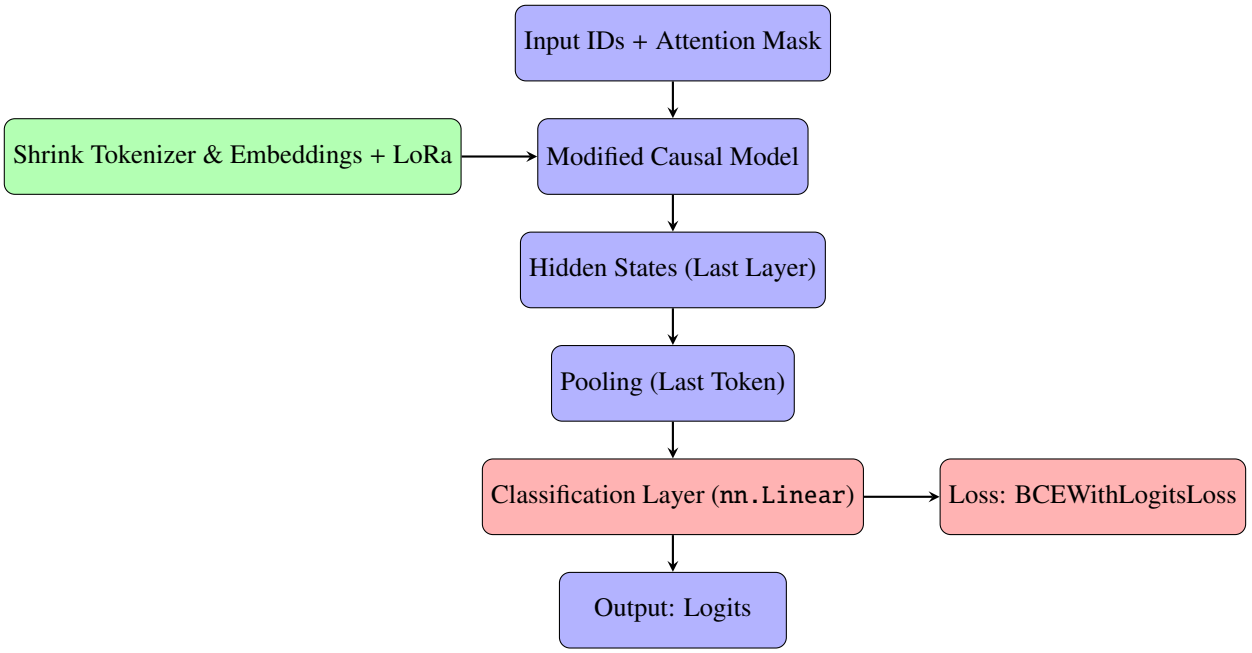


Figure 20: Architecure LSCM Models.

```
<|begin_of_text|><I>0 7 7
7 7 7
0 7 7
<O>0 0 0 0 7
0 0 0 7 7
0 0 0 0 7
0 7 7 0 7
7 7 7 7 7
---
<I>...
0 0<|end_of_text|>
```

Figure 21: Decoded example showing tokenized representation of a task.

5.3.2 Prompt-Guided Causal Classification LLaMA (PGCC)

We fine-tuned their 8B Llama model's attention mechanism, feedforward network, embedding layer, and output layer with LoRa. This was achieved with the explicit prompt shown in Figure 22. To have a fair comparison with the LSCM model, we only used data composed of 150 tasks. Due to the similarity of the prompt on which it was trained, the model could not output the perceptions in the desired format.

In hindsight, we should have added a classification head to the model and only fine-tuned the head. This would have ensured the desired output format and allowed for a fair comparison between the base Llama model and its BARC fine-tuned version.

```
<|begin_of_text|><|SYSTEM|>
You are an exceptional puzzle classifier with world-class pattern
    recognition skills. Your task is to analyze puzzles and
    accurately classify them using human perception categories.
You have been provided with a set of perceptions, and your
    response must include at least one and at most three of these
    categories. Here is the list of the perceptions with a
    descripition attached:
Containment: Recognize whether an object is inside, enclosed by,
    or surrounded by another object.
Depth:...
<|USER|>
Given input-output grid pairs, carefully observe the recccurent
    patterns to classify the pairs.
Each pair uses the same human perceptions and the order of the
    perceptions does not matter.
You SHOULD NOT include any other text in the response than the
    perception.
labelsGrids are 2D arrays represented as strings, with cells (
    colors) separated by spaces and rows by newlines.
Here are the input and output grids:

example 1
Input:
Red Black Red
Red Black Red
Red Red Black

Output:
Red Black Red Black Black Black Red Black Red
Red Black Red Black Black Black Red Black Red
Red Red Black Black Black Black Red Red Black
Red Black Red Black Black Black Red Black Red
Red Black Red Black Black Black Red Black Red
Red Red Black Black Black Black Red Red Black
Red Black Red Red Black Red Black Black Black
Red Black Red Red Black Red Black Black Black
Red Red Black Red Red Black Black Black Black

example 2
Input:
...
<|ASSISTANT|>
The labels are: spatial-ordinal, replication
```

Figure 22: Format Prompt PGCC Inductive Model.

5.4 Results

We trained the models on the data set composed of 150 examples and tested them on the 50 evaluation tasks due to GPU memory constraints. The results are displayed in Table 7.

Metric	**LSCM**	**LSCM FT**
Exact Match Ratio (Accuracy)	0.0	0.04
Hamming Loss	0.24	0.159
Precision (Samples Average)	0.082	0.28
Recall (Samples Average)	0.16	0.55
F1 Measure (Samples Average)	0.103	0.35

Table 7: Evaluation Metrics Prediction: classification threshold = 0.1

Due to time constraints, the results obtained lack sufficient information to determine whether the model effectively classifies the tasks. However, the results indicate that the model can learn, although further investigation is required to understand its learning plateau.

5.5 Future Work

It would be interesting to investigate whether there is a correlation between perceptions and the DSL functions used in the task solver. Such an analysis could help determine whether perceptions facilitate low-level DSL search. The current classification labels are not optimal. For instance, the ”containment” label is never used in the annotated tasks. This label could be replaced with three more specific labels: ”centroid,” ”closure,” and ”spatial ordinal.”[23] Additionally, we still need to classify the remaining training and test tasks and those in ConceptARC and 1D-ARC. More experiments addressing our limitations are required to compare a base model with an ARC-AGI fine-tuned model. Finally, perceptions must be incorporated into the models to evaluate whether they improve benchmark performance.

6 Conclusion

In this work, we revisited the challenges faced by autoregressive Transformer-based LLMs, particularly their ability to autonomously generate abstractions. To address this, we explored ARC-AGI, one of the most challenging benchmarks for LLMs to date. We derived the first mathematical and intuitive framework to explain its structure and challenges. Additionally, we reviewed state-of-the-art (SoTA) approaches and demonstrated that the current methods closely resemble Kociemba’s two-stage algorithm. Specifically, the problem is first transformed into a simplified state, which is then solved using a hashtable (in this case, the LLM).

We argued that solving ARC-AGI efficiently requires a two-level search, for which we proposed a framework to facilitate high-level searches. To this end, we demonstrated that solving ARC-AGI necessitates the use of 16 distinct and non-overlapping human perceptions, which serve as the foundation for high-level searches. We hand-labeled 200 tasks, which can easily be expanded into a dataset of 1.2 million.

Subsequently, we investigated how LLMs perceive ARC-AGI, i.e., whether their perceptions align with human perceptions. To test this, we compared two neural classification models fine-tuned on our classification dataset: one ARC-AGI pre-trained LLM, which retains an understanding of human knowledge, and another LLM that lost this understanding due to token and embedding transformations combined with binary classification. Due to time constraints, we were unable to complete the experiments and therefore cannot confirm whether ARC-AGI fine-tuned models understand tasks in a way that closely aligns with human perceptions.

Answering this question could also shed light on two significant philosophical questions, both tied to linguistic determinism. First, is language merely a tool for expressing thoughts, or is it a framework that shapes reasoning[24]? Second, is language a byproduct of intelligence, or is it intelligence itself, and thus, is the pursuit of AGI through LLMs merely chasing a chimera?

[23] Although ”containment” is present in tasks created for ConceptARC.
[24] Sapir-Whorf Hypothesis

7 References

Abbe, Emmanuel, Samy Bengio, Elisabetta Cornacchia, et al. (2022). *Learning to Reason with Neural Networks: Generalization, Unseen Data and Boolean Measures*. arXiv: 2205.13647 [cs.LG]. URL: https://arxiv.org/abs/2205.13647.

Abbe, Emmanuel, Samy Bengio, Aryo Lotfi, et al. (2024). *How Far Can Transformers Reason? The Globality Barrier and Inductive Scratchpad*. arXiv: 2406.06467 [cs.LG]. URL: https://arxiv.org/abs/2406.06467.

Akyürek, Ekin et al. (2024). *The Surprising Effectiveness of Test-Time Training for Abstract Reasoning*. arXiv: 2411.07279 [cs.AI]. URL: https://arxiv.org/abs/2411.07279.

Alemohammad, Sina et al. (2023). *Self-Consuming Generative Models Go MAD*. arXiv: 2307.01850 [cs.LG]. URL: https://arxiv.org/abs/2307.01850.

Alford, Simon et al. (2021). *Neural-guided, Bidirectional Program Search for Abstraction and Reasoning*. arXiv: 2110.11536 [cs.AI]. URL: https://arxiv.org/abs/2110.11536.

Artificial General Intelligence Conference 2024 (2024). https://agi-conf.org/2024/. Accessed: 2024-12-23.

Atzeni, Mattia, Mrinmaya Sachan, and Andreas Loukas (2023). *Infusing Lattice Symmetry Priors in Attention Mechanisms for Sample-Efficient Abstract Geometric Reasoning*. arXiv: 2306.03175 [cs.AI]. URL: https://arxiv.org/abs/2306.03175.

Barbadillo (2024). *ARC-24 Solution: Omni-ARC*. Accessed: 2024-12-23. URL: https://ironbar.github.io/arc24/05_Solution_Summary/.

Berglund, Lukas et al. (2024). *The Reversal Curse: LLMs trained on "A is B" fail to learn "B is A"*. arXiv: 2309.12288 [cs.CL]. URL: https://arxiv.org/abs/2309.12288.

Bober-Irizar, Mikel and Soumya Banerjee (2024). *Neural networks for abstraction and reasoning: Towards broad generalization in machines*. arXiv: 2402.03507 [cs.AI]. URL: https://arxiv.org/abs/2402.03507.

Boix-Adsera, Enric et al. (2024). *When can transformers reason with abstract symbols?* arXiv: 2310.09753 [cs.CL]. URL: https://arxiv.org/abs/2310.09753.

Bubeck, Sébastien et al. (2023). *Sparks of Artificial General Intelligence: Early experiments with GPT-4*. arXiv: 2303.12712 [cs.CL]. URL: https://arxiv.org/abs/2303.12712.

Butt, Natasha et al. (2024). *CodeIt: Self-Improving Language Models with Prioritized Hindsight Replay*. arXiv: 2402.04858 [cs.AI]. URL: https://arxiv.org/abs/2402.04858.

Chollet, François (2019). *On the Measure of Intelligence*. Available at: https://arxiv.org/abs/1911.01547. arXiv: 1911.01547 [cs.AI].

Chollet, Francois et al. (2024). *ARC Prize 2024: Technical Report*. arXiv: 2412.04604 [cs.AI]. URL: https://arxiv.org/abs/2412.04604.

Dziri, Nouha et al. (2023). *Faith and Fate: Limits of Transformers on Compositionality*. arXiv: 2305.18654 [cs.CL]. URL: https://arxiv.org/abs/2305.18654.

Ellis, Kevin et al. (2020). *DreamCoder: Growing generalizable, interpretable knowledge with wake-sleep Bayesian program learning*. arXiv: 2006.08381 [cs.AI]. URL: https://arxiv.org/abs/2006.08381.

Franzen, Daniel et al. (2024). *The ARChitects: Solving the ARC Challenge 2024*. Accessed: 2024-12-22. URL: https://github.com/da-fr/arc-prize-2024/blob/main/the_architects.pdf.

Geirhos, Robert et al. (Nov. 2020). "Shortcut learning in deep neural networks". In: *Nature Machine Intelligence* 2.11, pp. 665–673. ISSN: 2522-5839. DOI: 10.1038/s42256-020-00257-z. URL: http://dx.doi.org/10.1038/s42256-020-00257-z.

Greenblatt, Ryan (July 2024). *Solving ARC with GPT-4o*. Machine Learning Street Talk (MLST) Podcast Episode. URL: https://podcasts.apple.com/gb/podcast/ryan-greenblatt-solving-arc-with-gpt4o/id1510472996?i=1000661332631.

Hocquette, Céline and Andrew Cropper (2024). *Relational decomposition for program synthesis*. arXiv: 2408.12212 [cs.AI]. URL: https://arxiv.org/abs/2408.12212.

Hodel, Michael (2022). *Domain Specific Language for the Abstraction and Reasoning Corpus*. https://github.com/michaelhodel/arc-dsl. Accessed: 2024-12-22.

— (2024). *Addressing the Abstraction and Reasoning Corpus via Procedural Example Generation*. arXiv: 2404.07353 [cs.LG]. URL: https://arxiv.org/abs/2404.07353.

Hoffmann, Jordan, Sebastian Borgeaud, Arthur Mensch, et al. (2022). "Training Compute-Optimal Large Language Models". In: *arXiv preprint arXiv:2203.15556*. URL: https://arxiv.org/abs/2203.15556.

Jacot, Arthur, Franck Gabriel, and Clément Hongler (2020). *Neural Tangent Kernel: Convergence and Generalization in Neural Networks*. arXiv: 1806.07572 [cs.LG]. URL: https://arxiv.org/abs/1806.07572.

Kaplan, Jared et al. (2020). *Scaling Laws for Neural Language Models*. arXiv: 2001.08361 [cs.LG]. URL: https://arxiv.org/abs/2001.08361.

Kim, Sejin and Sundong Kim (2024). *System 2 Reasoning via Generality and Adaptation*. arXiv: 2410.07866 [cs.AI]. URL: https://arxiv.org/abs/2410.07866.

Kociemba, Herbert (1992). *Two-Phase Algorithm Details*. URL: https://kociemba.org/math/imptwophase.htm.

Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E Hinton (2012). "ImageNet classification with deep convolutional neural networks". In: *Advances in Neural Information Processing Systems*. Vol. 25. Curran Associates, Inc., pp. 1097–1105. URL: https://proceedings.neurips.cc/paper_files/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf.

Lapuschkin, Sebastian et al. (Mar. 2019). "Unmasking Clever Hans predictors and assessing what machines really learn". In: *Nature Communications* 10.1. ISSN: 2041-1723. DOI: 10.1038/s41467-019-08987-4. URL: http://dx.doi.org/10.1038/s41467-019-08987-4.

LeGris, Solim et al. (2024). *H-ARC: A Robust Estimate of Human Performance on the Abstraction and Reasoning Corpus Benchmark*. arXiv: 2409.01374 [cs.AI]. URL: https://arxiv.org/abs/2409.01374.

Li, Wen-Ding et al. (2024). *Combining Induction and Transduction for Abstract Reasoning*. arXiv: 2411.02272 [cs.LG]. URL: https://arxiv.org/abs/2411.02272.

Maslej, Nestor et al. (Apr. 2024). *The AI Index 2024 Annual Report*. Available at: https://aiindex.stanford.edu/wp-content/uploads/2024/05/HAI_AI-Index-Report-2024.pdf. Stanford, CA: AI Index Steering Committee, Institute for Human-Centered AI, Stanford University.

McCarthy, John et al. (1955). *A Proposal for the Dartmouth Summer Research Project on Artificial Intelligence*. http://jmc.stanford.edu/articles/dartmouth/dartmouth.pdf.

Moskvichev, Arseny, Victor Vikram Odouard, and Melanie Mitchell (2023). *The ConceptARC Benchmark: Evaluating Understanding and Generalization in the ARC Domain*. Available at: https://arxiv.org/abs/2305.07141. arXiv: 2305.07141 [cs.LG].

Nakkiran, Preetum et al. (2019). *Deep Double Descent: Where Bigger Models and More Data Hurt*. arXiv: 1912.02292 [cs.LG]. URL: https://arxiv.org/abs/1912.02292.

Nezhurina, Marianna et al. (2024). *Alice in Wonderland: Simple Tasks Showing Complete Reasoning Breakdown in State-Of-the-Art Large Language Models*. arXiv: 2406.02061 [cs.LG]. URL: https://arxiv.org/abs/2406.02061.

Nye, Maxwell et al. (2021). *Show Your Work: Scratchpads for Intermediate Computation with Language Models*. arXiv: 2112.00114 [cs.LG]. URL: https://arxiv.org/abs/2112.00114.

Ouellette, Simon (2024). *Towards Efficient Neurally-Guided Program Induction for ARC-AGI*. arXiv: 2411.17708 [cs.AI]. URL: https://arxiv.org/abs/2411.17708.

Ouellette, Simon, Rolf Pfister, and Hansueli Jud (2024). *Counting and Algorithmic Generalization with Transformers*. arXiv: 2310.08661 [cs.LG]. URL: https://arxiv.org/abs/2310.08661.

Park, Jaehyun et al. (2023). *Unraveling the ARC Puzzle: Mimicking Human Solutions with Object-Centric Decision Transformer*. arXiv: 2306.08204 [cs.AI]. URL: https://arxiv.org/abs/2306.08204.

Peng, Binghui, Srini Narayanan, and Christos Papadimitriou (2024). *On Limitations of the Transformer Architecture*. arXiv: 2402.08164 [stat.ML]. URL: https://arxiv.org/abs/2402.08164.

Radford, Alec et al. (2018). "Improving Language Understanding by Generative Pre-Training". In: *OpenAI*. URL: https://cdn.openai.com/research-covers/language-unsupervised/language_understanding_paper.pdf.

Rahaman, Nasim et al. (2019). *On the Spectral Bias of Neural Networks*. arXiv: 1806.08734 [stat.ML]. URL: https://arxiv.org/abs/1806.08734.

Shin, Donghyeon et al. (Nov. 2024). "From Generation to Selection: Findings of Converting Analogical Problem-Solving into Multiple-Choice Questions". In: *Findings of the Association for Computational Linguistics: EMNLP 2024*. Ed. by Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen. Miami, Florida, USA: Association for Computational Linguistics, pp. 6696–6708. DOI: 10.18653/v1/2024.findings-emnlp.392. URL: https://aclanthology.org/2024.findings-emnlp.392.

Spelke, Elizabeth S. and Katherine D. Kinzler (2007). "Core knowledge". In: *Developmental Science* 10.1, pp. 89–96. DOI: 10.1111/j.1467-7687.2007.00569.x. URL: https://doi.org/10.1111/j.1467-7687.2007.00569.x.

Vaswani, Ashish et al. (2023). *AttentionF Is All You Need*. arXiv: 1706.03762 [cs.CL]. URL: https://arxiv.org/abs/1706.03762.

Villalobos, Pablo et al. (2024). *Will we run out of data? Limits of LLM scaling based on human-generated data*. arXiv: 2211.04325 [cs.LG]. URL: https://arxiv.org/abs/2211.04325.

Volotat (2024). *The ARC Game*. Interactive interface for engaging with ARC tasks. URL: https://volotat.github.io/ARC-Game/.

Wei, Jason, Yi Tay, et al. (2022). *Emergent Abilities of Large Language Models*. arXiv: 2206.07682 [cs.CL]. URL: https://arxiv.org/abs/2206.07682.

Wei, Jason, Xuezhi Wang, et al. (2023). *Chain-of-Thought Prompting Elicits Reasoning in Large Language Models*. arXiv: 2201.11903 [cs.CL]. URL: https://arxiv.org/abs/2201.11903.

Weiler, Maurice et al. (2023). *Equivariant and Coordinate Independent Convolutional Networks. A Gauge Field Theory of Neural Networks*. Available at: https://maurice-weiler.gitlab.io/cnn_book/EquivariantAndCoordinateIndependentCNNs.pdf.

Wertheimer, Max (1923). "Untersuchungen zur Lehre von der Gestalt. II". In: *Psychologische Forschung* 4.1, pp. 301–350. DOI: 10.1007/BF00410640. URL: https://doi.org/10.1007/BF00410640.

Xu, Yudong et al. (2024). *LLMs and the Abstraction and Reasoning Corpus: Successes, Failures, and the Importance of Object-based Representations*. arXiv: 2305.18354 [cs.CL]. URL: https://arxiv.org/abs/2305.18354.

Zhang, Chiyuan, Samy Bengio, et al. (2017). *Understanding deep learning requires rethinking generalization*. arXiv: 1611.03530 [cs.LG]. URL: https://arxiv.org/abs/1611.03530.

Zhang, Chiyuan, Maithra Raghu, et al. (2022). *Pointer Value Retrieval: A new benchmark for understanding the limits of neural network generalization*. arXiv: 2107.12580 [cs.LG]. URL: https://arxiv.org/abs/2107.12580.