No AGI Without Inference-time Search

Simon Ouellette[1]

[1]Université du Québec à Montréal

1 Theory

In this position paper, we make the case that human-like AI (AGI) is not possible by only using a learning-based algorithm. We argue that a search component is necessary, in contrast to most modern AI systems that are typically end-to-end deep learning models (e.g. LLMs). To be more specific, we will argue that inference-time search is necessary to achieve out-of-distribution generalization in certain types of reasoning, such as abductive reasoning or planning.

The fundamental reason for the need for a search component is uncertainty. When a model is uncertain about its prediction, we mean that the probability of predicting the correct answer is less than one. When the model makes a prediction, it collapses this one-to-many mapping (one input has several possible outputs under the model's uncertainty) to a one-to-one mapping (the model must pick one answer - typically the most probable). When a *verifier* is available, as is often the case in abductive reasoning and planning problems, there is the opportunity to resolve that uncertainty.

A search component can iterate through the potential solutions as per the model's probability distribution (preferably in decreasing order of probability) to test which is correct. Purely learning-based systems do not intrinsically possess such a mechanism. Mathematically, a neural network is a deterministic mapping between a domain and a range. Therefore, what has to change, in order to resolve the uncertainty, is the way we process the predicted probability distributions. Instead of the traditional *argmax* that always picks the most probable answer, we must move towards a sophisticated search over the probability space.

The reason why resolving this uncertainty is fundamental for AGI, is that AGI is expected to work in open-world domains (such as reality itself), not just closed-world domains. In such domains, out-of-distribution generalization is key, because at inference time the model will regularly experience novel states (and thus experience uncertainty). We envision a system in which the solution of a given problem is, in a first step, approximated by the learned model. Given some problem domain, this process, akin to "Fast Thinking" or "System 1" in humans, quickly provides a starting point in the *manifold of possible solutions*, but it might be inexact due to sources of uncertainty, such as novelty or randomness. A second process, akin to "Slow Thinking" or "System 2", then refines the proposal by searching for an exact solution in the nearby region of the starting point provided by the learned model.

2 Evidence

Lake & Baroni (2023) studied out-of-distribution generalization with a purely meta-learning based transformer, on the SCAN and COGS dataset. The SCAN dataset involves mapping simple English commands such as "jump twice and walk" into sequences of atomic actions such as "JUMP, JUMP, WALK". The COGS dataset involves mapping a simple English sentence

MAIS 2024, October 10th 2024, Montréal

into a logical form that parses the semantics of that sentence. They find that transformers do generalize well when the task distribution consists of surface-level re-mappings between the English words and the instruction atoms. However, transformers failed to generalize to structurally novel tasks, such as longer output sequences on SCAN and more complex sentence structures in COGS, with error rates at 100%.

They concluded that transformers probably cannot learn the more difficult forms of generalization because they do not possess the intrinsic ability for what is known as productivity (Fodor & Pylyshyn (1988)). Productivity refers to the ability of a cognitive system to derive a potential infinity of combinations from an axiomatic set of concepts and rules. In computer science terms, this is arguably equivalent to search (see appendix C). In order to truly solve novel SCAN and COGS problem instances, an AI system would have to attempt compositionally novel solutions from a set of learned axioms, and then verify each of them against the expected output. Since it can only interpolate over compositional structures it has seen during training, however, it failed at this form of generalization.

Ouellette et al. (2024) studied the *sample efficiency* of various learning algorithms in planning-intensive environments such as Sokoban and Minigrid. First, they find that the learning systems equipped with a search component (model-based RL) outperform the purely learning-based ones when the agent's partial observation window gives little information about what to do next. That is, in levels where almost every single observation contained landmarks that made it clear what to do, such as a closed door that needs to be opened, purely learning-based systems performed well by learning "reactive" heuristics. In contrast, in large information-poor grids where a lot of observation frames were mostly empty cells, search-based systems performed much better. This is coherent with the uncertainty argument we proposed.

Furthermore, their proposed algorithm that mixes learning and search outperformed the state-of-the-art Transformer-only model by a factor of about 100 times in terms of *sample efficiency*. This is relevant to our argument because data efficiency is instrinsically linked to model uncertainty: the less data is used, the more uncertain a model will be. This is especially important in open-world domains where sampling indefinite amounts of data from the total space of possible observations is unrealistic.

Finally, possibly the strongest evidence is the Abstraction & Reasoning Corpus (Chollet (2019)), also known as ARC-AGI. It consists of a set of visual reasoning tasks in which a few input-output grid examples are provided, and the solver must discover the underlying procedure (or algorithm) in order to apply it to a test grid. It was designed specifically to be an open-world domain in which mere interpolation over a training space is not sufficient. This is achieved by having a hidden test set composed of distinct tasks from any of the publicly visible ones. Any visual reasoning task over grids that a human can solve is permitted for inclusion in ARC, making it essentially an intelligence test for AI systems.

The best solutions so far on ARC-AGI (ARCPrize (2024)) use inference-time search. Icecuber (2020), winner of a previous competition at 21% accuracy, doesn't contain any learning. It is a hard-coded, problem-specific search algorithm. MindsAI (Cole (2023)), at 47% accuracy, is a Transformer that does test-time fine-tuning on the task. For each task, it runs gradient descent (continuous search) on the examples. The authors indicated that this test-time fine-tuning is a key component of their solution. Finally, another solution (Greenblatt (2024)), at 43% accuracy, leverages a commercial LLM to generate a variety of Python programs and refine them iteratively until the correct program is found. In comparison, humans have an average success rate of 84% on ARC-AGI. LLMs on their own perform relatively poorly, with o1-preview at 18%, Claude 3.5 at 14%, and o1-mini at 9.5%.

References

ARCPrize. Arc prize leaderboard: https://arcprize.org/leaderboard, 2024.

Chollet, F. On the measure of intelligence. *CoRR*, abs/1911.01547, 2019. URL `http://arxiv.org/abs/1911.01547`.

Cole, J. Test-time augmentation to solve arc, 2023. https://lab42.global/community-interview-jack-cole/.

Fodor, J. A. and Pylyshyn, Z. W. Connectionism and cognitive architecture: A critical analysis. *Cognition*, 28(1-2):3–71, 1988.

Greenblatt, R. Getting 50% (sota) on arc-agi with gpt-4o, 2024. https://redwoodresearch.substack.com/p/getting-50-sota-on-arc-agi-with-gpt.

Icecuber. Summary of icecuber solution, 2020. https://www.kaggle.com/competitions/abstraction-and-reasoning-challenge/discussion/154597.

Lake, B. M. and Baroni, M. Human-like systematic generalization through a meta-learning neural network. *Nature*, 623(7985):115–121, 2023.

Ouellette, S., Beaudry, E., and Bouguessa, M. Conviction-based planning for sparse reward reinforcement learning problems. In *ICAPS Workshop on Bridging the Gap Between AI Planning and Reinforcement Learning*, 2024.

A ARC-AGI

See figure 1 for three examples of ARC-AGI tasks. Once the solver has identified the underlying procedure that converts the left grid into the right grid, it must apply it to a separate test grid and submit its solution. The ARC evaluation framework then compares this submitted grid to the hidden solution. If it is exactly correct pixelwise, it is considered a successful solution – otherwise it is considered a failure.

B Definitions

Abductive reasoning. A type of reasoning in which the reasoner is presented with a limited set of observations. The goal is to find the most plausible explanation for the observations. ARC-AGI is an abductive reasoning problem domain because the solver is presented with a few examples of input-output grid pairs, and must identify the underlying procedure that generated the outputs from the inputs. More generally, few-shot program induction is also a form of abductive reasoning.

Verifier. Refers to a mechanism that allows checking the correctness of a solution or prediction proposed by the system. In a planning context, for example, we typically know the exact state we are trying to achieve, so the verifier consists of comparing the current state to the target state. In abductive reasoning, for example, the solver must propose a hypothesis (in some language), that explains the observations. The observations can be, as in ARC-AGI, as set of inputs and corresponding outputs. A verifier, in this case, can take the inputs, apply the hypothesis, and verify if the outputs correspond to the ground truth. This is typical in program induction problems. In theory this verifier could be a learned model. It could also be imperfect, which would mark an upper bound to the system's ability to solve problems in a given domain.

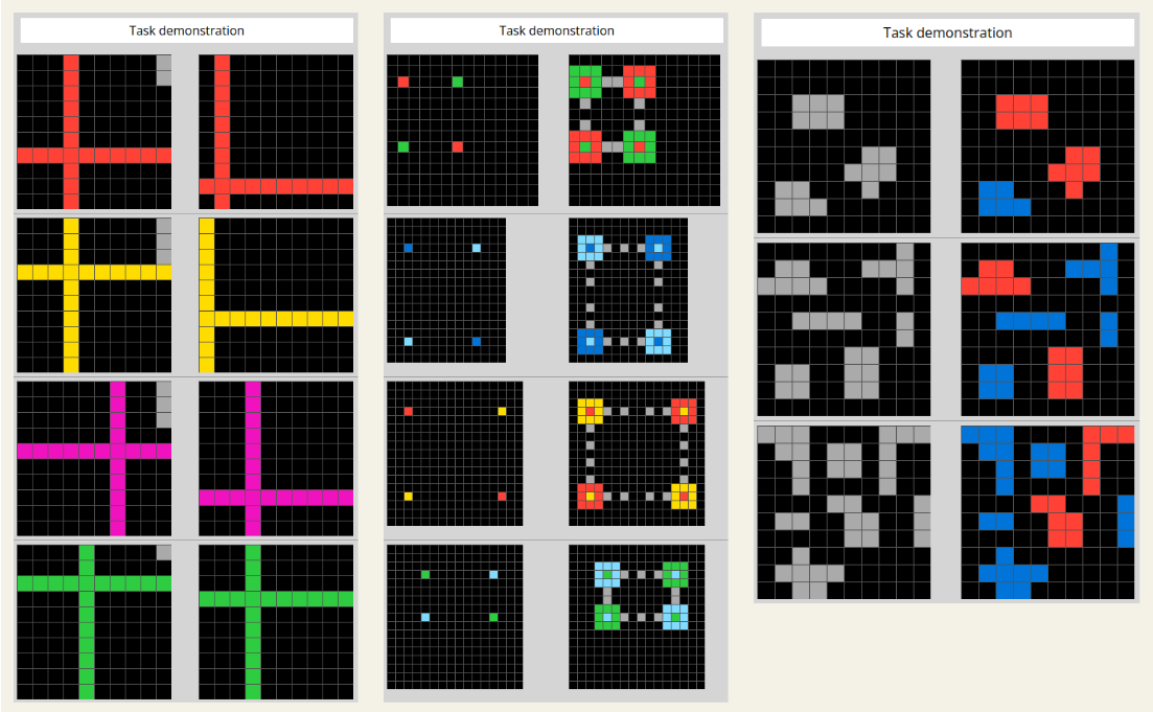

Figure 1: Three task examples from ARC-AGI. Left columns are the input examples, right columns are the expected outputs. **Left:** Must move the colored lines downward and leftward by the amount indicated by the number of grey pixels in the top-right of the grid. **Middle:** Must surround the pixels by a square of the "opposite" color, then connect them with grey dotted lines. **Right:** Must color red the shapes that have 6 pixels and color blue the other shapes.

Manifold of possible solutions. This refers to the set of multi-dimensional vectors that represents the list of valid solution proposals to a given problem. Typically only a small subset, or even only exactly one vector, corresponds to the correct solution. Spatially this manifold represents an N-dimensional coordinate system in which each point is one such solution vector.

Sample efficiency. It is an evaluation metric of a learning algorithm that quantifies the number of training examples to reach a pre-determined level of performance on some problem domain. The less examples are required, the more sample efficient the algorithm.

C Elaboration on productivity-search equivalence

Conceptually, productivity in cognitive science refers to having a set of axiomatic concepts and rules, and to being able to compositionally generate all possible permutations thereof. For example, an agent capable of productivity can be provided with the fundamentals of mathematics and can derive valid propositions from these. This agent is only limited by time and computational power – instead of being limited by what examples it has learned (indeed, it might not have been shown any example at all). In contrast, an agent that lacks productivity has an output bounded by the training domain. Even with infinite computational power and time, it would never reach a certain subset of the possible compositions.

The analogy between productivity and search here is as follows: in search, we are given a grammar from which to produce valid propositions, and then verify against the desired goal state. Using some heuristics, the search then seeks to find increasingly more accurate propositions. We argue that this process of generating valid proposals from a provided grammar can be thought of as precisely this concept known as productivity. While the analogy might be incomplete insofar as search is perhaps more than productivity, e.g. productivity plus verification, the context in which Lake & Baroni (2023) refer to productivity makes it more relevant.

Indeed, we interpret the authors as meaning that the Transformer could solve the structurally out-of-distribution tasks if it were able to generate unbounded compositions of some grammar and verify them against the in-context support set (i.e. the provided few-shot task examples). Furthermore, the authors argue that the issue is not that the Transformers do not successfully learn the fundamental concepts and rules. Instead, they suggest that it is the inability to compose these concepts and rules beyond the structures seen during training that is at fault: hence their insistence on productivity itself. Even if we misunderstood the authors in this regard, this is nonetheless how we interpret their empirical results.

D Discussion on Bayesian modelling

In Bayesian modeling, the uncertainty is made explicit because the Bayesian model outputs a distribution, rather than a single prediction. However, the problem requires an exact solution, not a probability distribution over solutions. As in non-Bayesian models, some inference-time mechanism must be applied to collapse this one-to-many mapping to a one-to-one mapping.

Bayesian modelling then, makes it more explicit that some additional step needs to be taken to find the exact solution among the predicted distribution of solutions. It could be a simple "max probability" heuristic, or a more sophisticated inference-time search process.

Hence, Bayesian modelling does not fix the fundamental issue we are discussing in this paper. The entirety of the argument applies to Bayesian models as well, though Bayesian models would be arguably better "System 1" components to use to guide the search, since the probability distribution is more explicit.