JOURNAL OF
EMERGING INVESTIGATORS

Article

A machine learning approach for abstraction and reasoning problems without large amounts of data

Suleyman Emre Isik[1*], Ali Eren Aytekin[2*], Halil Vurus[2]
[1] Bishop of Llandaff High School, Cardiff, United Kingdom
[2] Adana Anatolian High School, Adana, Turkey
*Authors contributed equally

SUMMARY

Data-hungry machine learning techniques can sometimes be more successful than human intelligence for the ability they have acquired for the specific task they were trained for. However, task-based machine learning techniques with very little data are rather inconsistent compared to human cognitive abilities due to the lack of generalization. This makes it difficult for algorithms to handle volatile and hard-to-predict real life problems. Alternative approaches that have the potential to offer human-like abstraction capability are needed. This research is aimed to create an algorithm that emulates the performance-like, reasoning tasks that people apply in Intelligence Quotient (IQ) tests without the need for large amounts of data. The created algorithm solves reasoning problems in the created data set. Generalization is expected to be able to solve arbitrary complex tasks rather than a skill acquisition for a task. We obtained an accuracy score of 0.834 for the solutions created by the developed algorithm. Significance tests on the variations of accuracy have shown that consistency is achieved through unknown tasks and over-fitting problems are avoided which was not the case for task-based developed Convolutional Neural Network (CNN) methods using Cellular Automaton (CA) during this research. The algorithm on abstraction-reasoning and testing provides a benchmark for measuring Artificial Intelligence (AI) skill acquisition in unknown tasks with very small amount of data to learn.

INTRODUCTION

Since its emergence, computer science has been producing solutions to complete tasks that people can do but have difficulty doing. Learning algorithms that emerged in the 1990s have become able to do many specific tasks better than humans by closely mimicking the way people think and learn. For instance, on March 15, 2016, for the first time, a computer beat a world champion in the game of Go, which is an abstract strategy board game with too many possibilities, using a powerfully trained deep learning model on this task (1). The major successes of the field have been in building special-purpose systems capable of handling narrow, well-described tasks, sometimes at above human-level performance. However, it is the narrow and specific task-based nature that differentiates AI from human level abstraction-reasoning ability which makes it difficult for algorithms to handle volatile and hard-to-predict real-life problems. The problems caused by this task-based nature necessitated flexibility and robustness for certain broader subfields of AI, such as L5 self-driving, domestic robotics, or personal assistants; there is even increasing interest in generality itself (e.g., developmental robotics, artificial general intelligence) (2, 3).

The first and most important step to take in order to offer an approach that is closer to human intelligence is to examine the concept of intelligence and to define it in the most useful way. Various definitions have been made for intelligence in the past. Legg and Hutter summarized the definitions made in the context of artificial intelligence research as follows: "Intelligence measures a person's ability to achieve goals in a wide and varied environment (4)." Two main characteristics are emphasized here: a task-goal focus and generalizability to a wide range of environments. Accordingly, while human intelligence can perform tasks with its high ability, these abilities can also be generalized for new tasks in new environments (skill acquisition). This feature is a mechanism that human nature has developed in line with evolutionary psychology to solve new unknown tasks and problems (5, 6).

In the direction of the development of AI, many approaches have emerged to develop and evaluate AI models. One of them is the human observational approach that examines, judges, and scores the system's inputs and outputs. This is a highly subjective, difficult, and expensive method to automate. White-box analysis, on the other hand, is inspecting the implementation of the system to determine its input-output response and score it (e.g., an algorithm that plays "Connect Four") (7). Peer confrontation, for example, is having the system compete against either other AIs or humans. This is the preferred mode of evaluation for player-versus-player games, such as chess. The benchmarking approach, which is based on enabling the system through algorithms to produce outputs for a "test set" of inputs (or environments) for which the desired outcome is known (solvable by humans), is another of the most valuable approaches for the evaluation of artificial intelligence. In particular, it is reproducible (test set fixed), scalable (cheap to run the evaluation multiple times), easy to set up, and flexible enough to be applied to a wide variety of possible tasks (8). For this reason, benchmarking has been an important part of progress in artificial intelligence

and is a suitable approach for this research.

In contrast to the task-based development of artificial intelligence from the past to the present, psychometry-abstraction and reasoning assessments methods- tests intelligence over broad cognitive abilities as opposed to task-based. What is important at this point is that the tasks were previously unknown to the person taking the test. Thus, the person who takes the test does not have a practice before. This approach in psychometrics is similar to the evaluation of artificial intelligence models. Recently, there has been a significant increase in the approach of testing broad cognitive abilities in parallel with psychometrics for systems targeting flexibility in the field of artificial intelligence. Examples of these are "Arcade Learning Environment for Reinforcement Learning agents" (9), "Project MalmO" (10), "The Behavior Suite" (11), "GLUE" (12), and "SuperGLUE" (13). The rationale behind these projects is to measure a more general skill than a skill in a particular task by expanding the range of target tasks. However, a major problem with these multitasking models when it comes to assessing flexibility is that the tasks are still known in advance by their developers and models are developed to pass tests. The task-based nature appears in forms such as a use of task-specific prior knowledge inherited from developers and external knowledge obtained through pre-training. Therefore, simply extending task-specific skills assessment to more tasks does not produce a qualitatively different type of assessment. Such benchmarks, in contrast to the psychometric approach, still look at skills rather than broad cognitive abilities (this does not mean that such measures are not useful, only that such static multitasking measures do not directly assess flexibility or generalizability).

Unlike these tasks and skill-based approaches, the idea of using cognitive tests for artificial intelligence, which is used to measure human intelligence in psychometrics, was suggested in 1964 (14). When we evaluate the concept of artificial intelligence word for word, this approach makes sense. One of the reasons is that if an AI is created by a developer to solve a question in a task-based IQ test, it won't work on the other question. Here the IQ test will really test the cognitive abilities of the artificial intelligence rather than the intelligence of the developer. If two people with the same prior knowledge and experience are asked to solve an unprecedented problem, the person with the higher intelligence will perform better. Here, intelligence is independent of skill; skill is only an output of intelligence. The intelligence referred to here is fluid intelligence that can reason in unique situations, rather than crystallized intelligence based on past training (15).

We created an algorithm that solves the performance-like reasoning tasks that people apply in IQ tests without a need for large amounts of data. The created algorithm solves the reasoning problems that have not been seen before. The examples regarding to the problems are as in **Figure 1** which includes a sample task demonstration with input(i) on the left and output(o) on the right for each case (a and b) and **Figure 2** which is the input about the task that the test taker

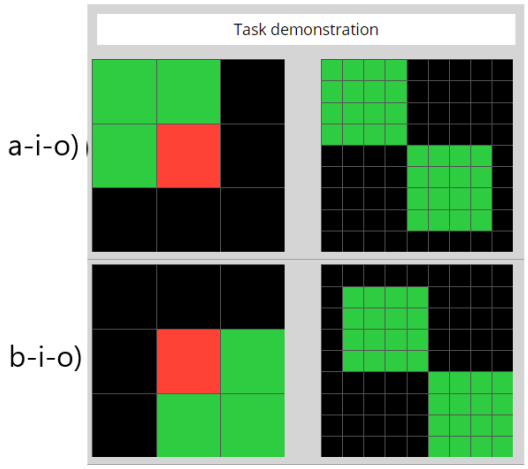

Figure 1. A task demonstration used to develop a reasoning. Case a part o is a solution of the case a part i and case b part o is a solution to case b part i.

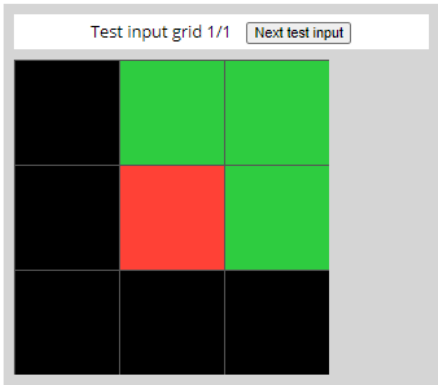

Figure 2. Test input that test taker sees to produce the corresponding output. The input is supposed to be solved by the reasoning developed in the task demonstration.

is expected to solve. If task demonstrations are examined to create the output shape for this sample task input, it can be deduced that a 9×9 output should be created for the 3x3 input. Then, it can be deduced that the green cells in the input form an output connection from the corners of the 4×4 green cells from the corner to the red end. The expected solution for this is shown in **Figure 3**.

Another random task solution is shown in full screen in **Figure 4**. When the task demonstrations are examined, it can be deduced that the bounded area among the yellow cells determine the dimensions of the output grid. Each colored object inside the yellow area has various dimensions and for each colored box in the figure outside the yellow area, a shape was created according to the dimensions of the objects inside the yellow areas in the output. As a result, the shapes outside the yellow area were placed inside the yellow area without changing the position of the shapes in the yellow area. While humans can solve these and similar never-before-seen problems in a short amount of time, even a powerful deep

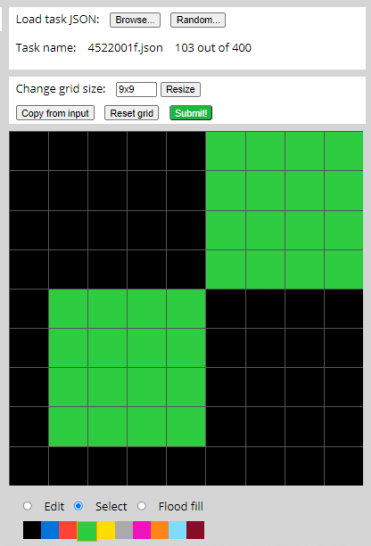

Figure 3. The solution of the test input based on some simple reasonings. The solution must be 9x9 and 4x4 green squares must be connected by their corners in the same direction where the red cell points out.

learning algorithm needs to be trained with a load of data to solve only one simple task. For a task, the test taker will have access to the inputs and outputs of the task representations (training set) as well as the test set's input. The goal is to generate a test set output for each test set input. While doing this, it is expected that the dimensions of the output grid are selected, and each cell is assigned an integer between 0 and 9 representing the colors.

Generalization is expected to be able to solve different tasks and problems rather than a skill for a task. Therefore, the development of the algorithm through sample tasks and the evaluation of the algorithm over confidential data were carried out by two developers who were unaware of each other's work.

In this research, we built the algorithm on abstraction-reasoning aims to provide a benchmark for measuring non-stochastic AI skill acquisition in unknown complex tasks, which opens a door to Artificial General Intelligence through the consistency achieved by very little data.

RESULTS

An algorithm has been developed to solve a set of problems, which involves coloring the grids according to its corresponding input, that requires similar cognitive abilities to those examined in IQ tests. The results of the research are obtained through the developed genetic algorithm involving Domain Specific Language (DSL) and various functions, some of which involve machine learning, to figure out the solutions.

For each task in the test set, up to 3 outputs can be produced as a prediction for each test input grid. For a given task output, if the ground truth is contained in any of the 3 predicted outputs, then the error for that task is 0, otherwise it is 1. The final score is averaged across all tasks. Mathematically, for each task i, the algorithm can make up to 3 predictions o_{ij}, where $1 \le j \le 3$. The error, e, for the task *i* with ground truth g_i is:

$$e_i = \min_j d(o_{ij}, g_i)$$

Where d(x, y) is 0 if x = y, otherwise 1. The overall error score is the average over all N task outputs:

$$score = \frac{1}{N}\sum_i e_i$$

Figure 4. A task demonstration, test input and solution. The simple goal is to scale the shape outside the unconnected yellow cells by using the separate shapes inside the yellow cells with a grid size equal to the size of quadrilateral bounded by yellow cells.

The developed algorithm obtained an accuracy score of 0.882 when evaluated with the data available by the first person who developed the algorithm. Augmenting the samples with diagonally flipped tasks was a simple trick that gave significantly better score during the development. Preprocessing all samples by remapping colors according to some heuristics also worked well. Functions which take more than one parameter were growing the search space super-exponentially, and that they usually took either the input or output image size as the second argument. Finally, when evaluated by the second developer, who is responsible for datasets and tests, with the three other data sets unknown by the first developer, an accuracy score of 0.834 was achieved on average.

Binomial hypothesis testing has been executed to understand the meaning of this difference in accuracy scores. Sufficient evidence has been obtained to reject the null hypothesis of H_0, which specifies that the accuracy scores are not significantly different, at 5% significance level with a p-value of 0.00328. Therefore, the one-tailed alternative hypothesis that claims the accuracy scores are significantly lower is considered to be valid. This was an unexpected result as the characteristics of all datasets are similar. However, despite this, the scores have shown consistency and still can be considered as high. This reveals that generalization, which is the ultimate purpose of this research, has been achieved and overfitting problems are avoided. If the accuracy scores in the evaluation phase were consistently and significantly higher than those in training, overfitting concerns could arise that would disrupt generalization.

Everything was implemented efficiently in C++ (with no dependencies) and run in parallel. A simple scheduler tried to use the 9 hour / 16 GB memory efficiently. The implemented algorithm was run on Windows and Mac OS X, no GPU required. A natural way to make this approach run faster is to reduce the search depth. When a full 9 hours is used, about half the problems can be run at depth 4, while running at depth 3 is about 20x faster (and takes 20x less memory). During development, the algorithm had been running at depth 2, which is again about 15x faster than depth 3, while solving about 70% as many tasks on the evaluation set. All in all, this algorithm has a low computational hardware requirement, and it can even be lowered by changing parameters and giving up some accuracy.

DISCUSSION

In this paper, we considered the relationship between intelligent system and skill. **Figure 5** shows the intelligent system's process of solving a task by creating a skill. For example, while the system that creates neural networks for solving tasks is an intelligent system, the model that works for that task is a skill. The interaction between the task, the intelligent system, and the skill program consists of two phases: the training phase and the evaluation phase. The purpose of the training phase is to build a skill that will generalize to future assessment situations. The purpose of the assessment phase is to assess the capacity of this ability to cope with new situations.

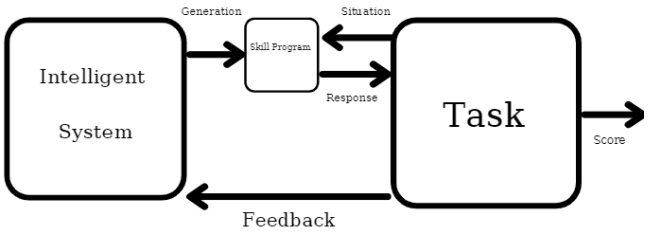

Figure 5. The relationship of intelligent system and task through skill. A program synthesis engine that can look at a task and output a solution program will be an "intelligent system" and the resulting solution program that can handle future input for that task will be a "skill program".

In this research, we considered pairs of source-target training images, and the task was to infer the rule which will allow to transition from source image to the target and later apply such rule to an arbitrary number of images. The rules which should be inferred were extremely diverse, and therefore the first question to ask was how should such rules be represented? The knowledge representation is a critical problem, therefore a long literature review has been done on it (16). We found that CA method, which is a collection of transition rules that specify how to update a set of numbers situated on a grid with a recurrent CNN, may be useful because its quite complex behaviors emerge from rather simple rules (17).

Our first approaches involve simple data augmentation techniques and a supervised 2D CNN model to make predictions. The model takes a 2D matrix as input and outputs, the SoftMax probabilities of different values occurring in the output matrix. However, only a few training examples were available to the first developer for each task; new input-output pairs are created, for example, by randomly switching colors. The extra augmented data helps the model capture patterns more easily. It can be seen from in **Figure 6** that the same training pairs are augmented (hundreds of times) to produce a large dataset. This dataset is used to train the recurrent CNN for each task. The CNN predicts a probability distribution over the "pixels" or values in the matrix. This probability distribution is used to generate the final output matrix. This method achieved up to 0.95 accuracy score in the training set with a low consistency. However, this method failed in the never-seen tasks by the accuracy score falling under 0.30 since generalization, the purpose of this research, couldn't be achieved as each task required different reasoning.

Where all tasks could be solved by a correct use of CA, some tasks were much more arbitrarily complex than others. These tasks largely rely on programmers' skills, hence, there still was a probability of failing. This sparked the need for a system with some 'general skills' that automatically detect shapes, symmetries, background color, or features that remain unchanged. A DSL had to be developed to express

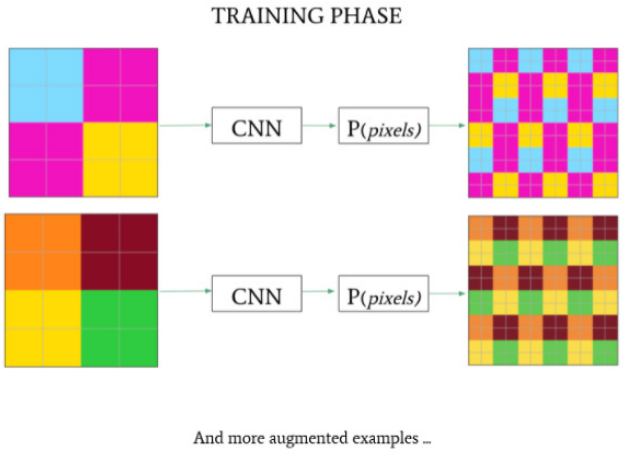

Figure 6. Producing larger data sets through augmentation of the same training pairs. We used the augmented dataset on a recurrent CNN for each task that predicts a probability distribution.

these most basic concepts that people naturally accept and easily identify. A framework was developed, and various functions have been written to analyze the sample tasks at the matrix level and to apply basic transformations.

Combining and organizing the developed DSL with various functions was essential. After a period of research, we concluded that the smartest way to do this organization was to develop a genetic algorithm that will combine basic operations until the desired result is achieved. Once a simple and powerful DSL has been developed to express various transformations on images, it must now be applied to an efficient genetic algorithm that can find the solution to a task by itself. The implemented algorithm achieved consistency in completing the tasks without any of overfitting symptoms, unlike the deep learning-related approaches.

The studies carried out in line with the purpose of this project play a very important role on the way to Artificial General Intelligence. It has shown that there are so many "simple" things that deep learning algorithms cannot do that simply; this step towards solving this problem will make a significant contribution to the progress through the identification of abstractions which has always been an extremely powerful tool in math, and the right abstraction methods have the potential to be game-changers in AI. While the main objective of this research, generalization, has been achieved as shown by the consistency in our results section, the common limitations in AI such as poor ability to understand correlations, causations, or ontological relationships separately, are not ultimately solved and further studies are needed in this direction. Also, we suggest working on completely different types of datasets with a similar objective to obtain further insights for the main goal.

MATERIALS AND METHODS

For this paper, we created abstraction and reasoning tests which are expected to measure fluid intelligence instead of crystallized intelligence. Construction of the grids for a task involved picking the height and width of the output grid, then filling each cell in the grid with a symbol (integer between 0 and 9, which are visualized as colors). The data sets were constructed using Notepad++ in JSON format. Each task requires some simple cognitive abilities, but unlike most IQ tests, it doesn't include any mathematical or verbal tasks. These tasks have no real-world counterpart and are completely based on simple reasoning and abstraction abilities. We developed a simple interface to make these tasks easier to understand. Every task must be solvable by humans. The tasks contain a task demonstration with output as a solution to inputs of different colors and shapes on a scalable(n×m) grids. Each task is solved with different cognitive methods. After a few task demonstrations for a randomly selected task, the test taker is asked to give an input to solve and draw the output over the interface. The taker has 3 tries and if one of them is correct, the solution is considered successful. Only exact solutions (all cells match the expected response) are considered correct. Accordingly, training and evaluation datasets contain both training and test input and output pairs. Each task includes task demonstrations and tests input-output pairs. This set is used to prototype the algorithm. The evaluation set contains 400 evaluation tasks. It allows the developer who created the algorithm to complete and evaluate the algorithm. The test set includes 3 separate sets of 100 tasks that were not seen by the developer who set up the algorithm and were for final scoring.

We developed a DSL to express these most basic concepts that people naturally accept and easily identify. A framework has been developed on this premise. Various functions have been written to analyze the sample tasks at the matrix level and to apply basic transformations. All the nodes in the DSL are either images or lists of images. An image has a position, a size, and a 2D array of colors (0-9). Black (0) is treated as transparent in most of the functions. First, up to 3 or 4 unary transformations (modifications on the basic features described to handle the data) to the input image are applied sequentially with a search depth 3, depth 3 augmented with diagonal flips (times two diagonal flips), and finally run depth 4 until running out of time or memory. Then, a subset of the following is applied (in order): stack lists of images→ move image to origin→ color all non-black pixels some color (for each color in every training output) → resize image (crop / pad) to fit output size. Finally, stack the results of the transformations above to produce a final output. However, an example such as in **Figure 7** is still simple and needs better enhancements to complete highly complex arbitrary tasks. For this, we developed a genetic algorithm to combine and organize the developed DSL with various functions. Once a simple and powerful DSL was developed to express various transformations on images, genetic algorithm organizes it until the desired result is achieved.

This strategy works as follows: A random program is created and run with a node. The best solution is kept. Based on these best solutions, a new program is created, reevaluated

Figure 7. A task demonstration, test input, and solution. The reasoning is as simple as changing the color of the orange cells to grey.

and updated by mutation. This process continues over and over until a solution is found. Because some multi-fitness functions are used, our approach can be multi-purposed: it can try to optimize more than one goal at the same time. Since up to three predictions can be made for each task, the genetic algorithm follows the three best candidates. Candidate solutions will be a set of operations with the resulting task (the original input matrices and the output matrices created when applying the various set of functions to them). A score, which expresses the number of pixels that differs between actual outputs and candidate-generated outputs, will then determine which candidate is better.

In the direction of this strategy, we created and implemented an algorithm as represented in **Figure 8**. This algorithm follows these steps: concepts such as task properties, matrix shape, colors, and symmetries pass through the preprocessing stage and are stored in 6 basic classes. These classes are Task, Sample, Matrix, Shape, Grid, and Frontier. An object of the Task class is created and located in the Task.py file. Based on the information from the preprocessing, the task can be turned into something easier to handle. For example, it might make sense to change some colors, rotate some matrices, crop the background, or ignore some grids. After the transformations are done, three dummy candidates are looped to get the three best candidates. At each iteration of the loop, the associated functions are executed for each of the three best candidates available, considering the properties stored in the object of the Task class. Many different functions have been tried, such as "moveShapes", "replicateShapes", "extendColor" or "pixelwiseXorInGridSubmatrices", to name a few examples. If any function produces a candidate that scores better than any of the top three available candidates, that new candidate is included in the top three candidates list and the worst one is removed. The score is calculated by executing the functions that lead to generating that candidate in the training samples and checking how many pixels are wrong. Therefore, 0 is the best possible score. The final candidates are obtained by

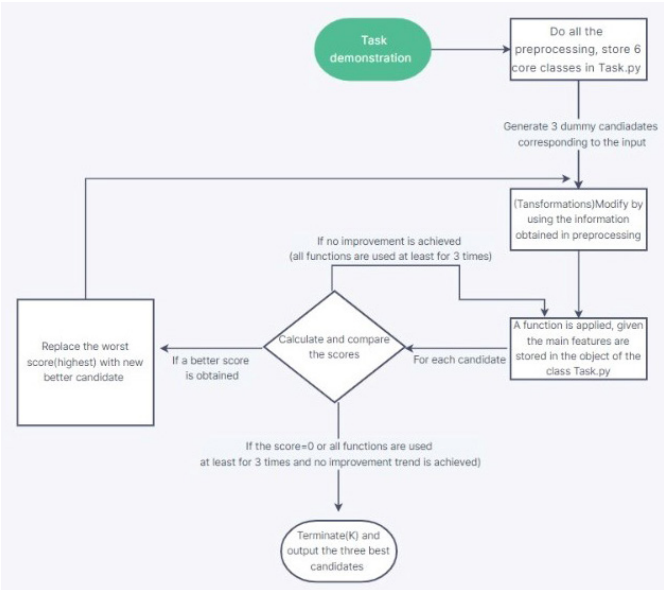

Figure 8. Flow charts representing the simplified algorithm. The algorithm involves the preprocessing, transformations, loops, and selection mechanism of the best three candidates.

undoing the transformations carried out from the last three candidates obtained.

The list of functions that can be executed for each candidate is in "getPossibleOperations". Many are executed several times with different parameters. All of them contribute to solving at least one of the tasks given in the training or evaluation set.

Received: June 25, 2021
Accepted: February 01, 2022
Published: June 25, 2022

REFERENCES

1. "Artificial Intelligence: Go Master Lee Se-Dol Wins against AlphaGo Program." *BBC News*, BBC, www.bbc.com/news/technology-35797102.amp
2. Asada, Minoru, *et al*. "Cognitive Developmental Robotics: A Survey." *IEEE Transactions on Autonomous Mental Development,* vol. 1, no. 1, pp. 12-34, May 2009. doi: 10.1109/TAMD.2009.2021702
3. Goertzel, Ben and Cassio Pennachin, editors. *Artificial General Intelligence*. Springer Berlin, 2007. link.library.missouri.edu/portal/Artificial-general-intelligence-Ben-Goertzel/hZxPPtxxmSY/.
4. Legg, Shane, and Marcus Hutter. "A Collection of Definitions of Intelligence." *Frontiers in Artifical Intelligence and Applications,* vol. 157, pp. 17-24, 2007. doi: 10.48550/arXiv.0706.3639
5. Cosmides, Leda, and John Tooby. "Origins of Domain Specificity: The Evolution of Functional Organization." *Mapping the Mind: Domain Specificity in Cognition and Culture.* Edited by Lawrence A. Hirschfeld and Susan A. Gelman, Cambridge: Cambridge UP, 1994. pp. 85-116.
6. Weidman, N. (2003), Steven Pinker. *The Blank Slate:*

The Modern Denial of Human Nature. New York: Viking, 2002. doi: 10.1002/jhbs.10173
7. Allis, L. Victor. “A Knowledge-based Approach of Connect-Four”, Journal of International Computer Games, vol. 11, no. 4, pp.165, Oct. 1988. doi: 10.3233/ICG-1988-11410
8. Schmid, Martin, *et al.* “Player of Games.” *arXiv preprint,* 6 December 2021. doi: 10.48550/arXiv.2112.03178
9. Bellemare, Marc, *et al.* “The Arcade Learning Environment: An Evaluation Platform for General Agents.” Journal of Artificial Intelligence Research, vol. 47, pp. 253-279, June 2013. doi: 10.1613/jair.3912
10. Perez-Liebana, Diego, *et al.* “The Multi-Agent Reinforcement Learning in MalmÖ (MARLÖ) Competition.” *Challenges in Machiine Learning (NIPS Workshop), 2018,* 23 Jan. 2019. doi: 10.48550/arXiv.1901.08129
11. Osband, Ian, *et al.* “Behaviour Suite for Reinforcement Learning.” *arXiv preprint*, 14 Feb. 2020. doi: 10.48550/arXiv.1908.03568
12. Wang, Alex, *et al.* “GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding.” *arXiv preprint*, 22 Feb. 2019. doi: https://doi.org/10.48550/arXiv.1804.07461
13. Wang, Alex, *et al.* “SuperGLUE: A Stickier Benchmark for General-Purpose Language Understanding Systems.” *arXiv preprint*, 2 May 2019. doi: https://doi.org/10.48550/arXiv.1905.00537
14. Green, Bert F. Jr. “Intelligence and Computer Simulation”. *The New York Academy of Sciences,* 1964. doi: 10.1111/j.2164-0947.1964.tb03486.x
15. Cattell, Raymond B. “Abilities: Their Structure, Growth, and Action.” *Houghton Mifflin*, 1971.
16. Davis, R., Shrobe, H., and Peter Szolovits. “What is a Knowledge Representation?” *AI Magazine*, vol. 14, no. 1, pp. 17-33, 1993.
17. Gilpin, W. “ Cellular automata as convolutional neural networks.” *Physical Review E*, vol. 100, no. 3, 2019. doi: 10.1103/PhysRevE.100.032402