Toward a General-Purpose Artificial Intelligence Test by Combining Diverse Tests

C. Pereyda, L. Holder
School of Electrical Engineering and Computer Science, Washington State University, Pullman, WA, USA

Abstract— *Deciding whether an AI system is intelligent has been a challenge since the creation of computing machines. The standard method for determining intelligence utilizes the Turing Test. This test has fallen out of favor due to the rapid development of AI systems and its subjectivity. With so many robust systems currently in use today, researchers need a more objective method for determining the intelligence and performance of an AI system. We attempt to begin solving this problem by creating a simple, yet generalizable test. This test is constructed using several well known tests. We examine the results of applying this test on AI systems and discuss how each sub-test can be validated for efficient and effective use in this test.*

Keywords: Evaluation, General intelligence, Combined test, Pattern recognition

1. Introduction

The purpose of this paper is to explore a novel method for evaluating AI systems and measuring the tests used in the evaluation process. We believe this work can be further extended to create a generalized AI system testing framework. There is not yet a standard framework or testing system that has been widely adopted by researchers in the field of AI. With so many new and varying AI systems being created, there is a strong need for a unified testing framework that is compatible with a multitude of approaches. This framework would need to be general enough to allow the wide spectrum of applications that AI can be applied to, such as problem-solving, inference, learning and perception.

Measuring the performance of an AI system using a well-defined problem is a simple task. These problems are often not generalizable and the implemented AI systems are often not usable for different problems. At the other extreme, we could use every feasible test to measure the performance of an AI system. This idea was examined by Hernández-Orallo [3], who proposed using the set of all possible tests and laid the theoretical foundation for an implementation. This however is not practical due to the set of possible tests being infinite. To solve this, we attempt to create a test that is formed from a combination of already defined tests.

Using this combination test we explore how an agent performs when given diverse challenges. We measure the consistency of a sub-test by applying handicaps to the AI systems. The verifiability of each sub-test is then evaluated by varying the amount of training an AI system uses to solve the test. These test properties can then be used to evaluate future sub-tests for inclusion to a more generalized combinational testing framework.

The rest of the paper is outlined as follows. We will explore the idea of intelligence in section 2. Section 3 formally defines our problem. Sections 4 and 5 explain the two individual tests that comprise our combined test: the Raven's Progressive Matrices test and the OpenAI Gym Cart-Pole test. Section 6 examines mixing of both RPM and OpenAI Gym. Section 7 explains how we trained and evaluated our AI systems on the proposed tests. In section 8 we explore novel methods for verifying our tests. Section 9 discusses the experimental results. Finally in Section 10 we conclude our work and give some ideas on further directions.

2. Intelligence

The standard method for determining human intelligence is an IQ test. The most notable IQ tests are the Stanford-Binet and the Wechsler Adult Intelligence Scale [7]. IQ tests allow for the evaluation of problem-solving skills. This is often considered to be an important aspect of intelligence. Other tests allow for the evaluation of some prior knowledge. A classic example of a prior knowledge test is the Scholastic Assessment Test (S.A.T.), where a large portion requires the knowledge of English vocabulary. These tests are very rigid and do not easily allow for adaptability. They can also rely very heavily on an understanding of a certain language. This is a problem for most AI systems as they do not generally possess the ability to understand a language, which is a challenge for many AI systems. This aspect eliminates many IQ tests from being usable for testing AI systems.

The previously held gold-standard for determining an AI system's intelligence has been the Turing Test [11]. An AI system can pass this test if it can trick human judges that it is human. This evaluation method has fallen out of favor due to the subjective nature of the test [3]. With the development of highly sophisticated AI systems, we need a more objective form of measurement. Ideally this test would measure an absolute intelligence of the AI system. For the measurement to be absolute, we need to remove the fundamental relativism that is currently used to describe intelligence. This measurement of intelligence would include its ability to learn information and apply the information it has already learned.

Within this field, there is no universal definition of intelligence. Legg and Hutter examined how intelligence is defined in the field of psychology and AI research, as well as various other fields [6]. They ultimately came to the informal definition, "Intelligence measures an agent's ability to achieve goals in a wide range of environments." This definition stemmed from the most common features attributed to intelligence. Intelligence is something that is found by examining the interaction of an agent and the environment it is in, how well the agent can achieve some goal in the environment, and how well the agent can adapt to new environments and goals.

Hernández-Orallo and Dowe expanded on this idea by laying the theoretical foundation for a general method for evaluating intelligence [4]. They propose that the particular environment used for evaluation does not matter, but rather the environment's complexity. The intelligence of an agent can be found by evaluating the average performance of the agent over many different environments where the performance is scaled by complexity. A well-defined implementation of one environment using this theory was developed by Javier [5]. The work done by Javier precisely follows from the work of Hernández-Orallo and Dowe. He evaluated a few different AI systems on a single problem and found how each performed compared to each other as the complexity was changed. His work has led us to further explore the idea of creating a methodology for implementing the underlying theory. The strength of this theory lies in how well it is defined. Determining the difficulty of an environment relies heavily on Algorithmic Information Theory, which was formally developed by Solomonoff [10]. The difficulty is mainly determined by how hard it is to create the environment. The complexity of an environment can generally be found through Kolmogorov-complexity approximations [12] or Minimum Message Length [4]. These ideas were developed from the field of information theory and are usable for most environments.

3. Problem Definition

Hernández-Orallo and Dowe [4] formally defined how to find the intelligence of an agent from a finite number of interactions over a finite set of environments. In their work they used Kolmogorov-complexity [12] to objectively determine the difficulty of an environment. They then used this complexity to weight the score of each environment. In our work we have empirically found the difficulty of each environment and we will denote this value d. This difficulty is then normalized against the others environments' difficulties to generate a weight w. We then find the agent's score in the environment by averaging the interactions between the agent and the environment (An interaction could be one test question or one game). This score will be denoted V. Following Hernández-Orallo and Dowe's notation, we will denote a given agent as π, an environment as μ, and an interaction between the agent and the environment as r. We will define the set of used tests as T. We can find the weight associated with an environment w_μ by using its corresponding difficulty d_μ.

$$w_\mu = \frac{d_\mu}{\sum_{\mu \in T}^{T} d_\mu} \tag{1}$$

Thus the average score an agent π receives over N interactions on a given environment μ is:

$$V_\mu^\pi = \frac{1}{N} \sum_{i=1}^{N} r_i^{\mu,\pi} \tag{2}$$

Where $r_i^{\mu,\pi}$ is the score the agent μ receives on an environment π for some interaction i. Using this value we can then find the general intelligence of the agent over the set of possible tests T:

$$\Upsilon(\pi) = \sum_{\mu \in T} V_\mu^\pi * w_\mu \tag{3}$$

We are attempting to determine the intelligence of an AI system through the use of testing. The specific test that we will use is a combination of two already existing tests: Raven's Progressive Matrices [8] and OpenAI Gym's CartPole environment [1]. The score from both of these environments will be used to determine the final result for the agent. This is not intended to show that we can create a general intelligence, but that this method of evaluation is valid and can be further scaled to evaluate a proposed general intelligence.

Our test consists of two sub-tests. During the evaluation, the agent will be given a random sub-test and asked to find the solution. The two tests were chosen because they are both deterministic and fully-observable. They are also tests that a human could easily solve given enough time. The two tests are also distinct enough that the agent should not be able to use the exact same logic to solve both of them, thus requiring a more general agent to achieve better results. We will evaluate several differently trained agents using this test and compare their performance. One agent will be trained only on the RPM task and another trained only on the OpenAI Gym environment. We will then introduce a new agent that has been trained on the combination test.

These two sub-tests will form the basis of a testing framework. In this work we attempt to justify the effectiveness of this framework as a tool for measuring the intelligence of an AI system. This justification will come from the performance measures of our baseline system across two validation methods. The first validation method examines how the agent performs when handicapped. The method will be valid if a linear decline in score is observed when an agent is incrementally and linearly handicapped. The second method examines how an agent performs when the amount

of training is varied. The method will be valid if we observe a steady linear increase in score as the amount of training increases linearly. If both of these methods show to be valid, then this framework has merit for measuring the intelligence of an AI system.

The test is designed to be a black box for the agent and ideally for AI designers. This is to prevent AI designers from gaming the system by creating idiot savants. One such example of this was done by Sanghi and Dowe [9], who showed it was easy to create a specialized AI system for IQ tests. This AI system was created in 960 lines of Perl code and was able to achieve an average IQ of 96 (across various tests). This result is very impressive and shows how a simple system can fool a test. Creating highly specialized AI systems is very effective for singular purposes, but will not serve to help the problem of generalizability.

4. Raven's Progressive Matrices

A well known method for measuring intelligence is through the use of Raven's Progressive Matrices (RPM) [8]. This test is ultimately a pattern recognition test that was designed to be simple to administer and easy to interpret. A RPM generally consists of 8 sample images setup in a 3x3 grid with one space from that grid missing. The goal is to find the most correct solution from a list of possible choices. Each image shows various characteristics that can be used to determine a pattern and find the answer. These characteristics can be anything that is discernible inside of the image, such as color, shape, orientation and so on. An example RPM can be seen in Figure 1; the correct solution for the missing image is a small green circle or [2,2,0].

For our test, we created a simple RPM generator to train and evaluate our agents. The generator creates a random RPM with N characteristics for each object. Each characteristic may be changed either across the rows or the columns, but not both. Each characteristic is limited to N possibilities for simplicity. Each characteristic starts at some random value and is then randomly changed to an unused value as the row or column changes. More difficult RPMs can have more characteristics with more unique progressions. The generator creates the whole $N \times N$ RPM and then separates out the $N^2 - 1$ sample data and the solution data for use with our agents.

The score of an agent on this environment is determined from the number of correct characteristics guessed. For a N-RPM problem the score is scaled to be the total number of correct guesses, maximum N, divided by N. This maintains a constant maximal score of 1 and a minimal score of 0 across all N-RPM problems. In general a correct guess in a RPM problem is only achieved when all characteristics are guessed correctly from the set of possible choices. To allow for easier training methods, we chose to have the AI system generate the solution rather than compare potential solutions.

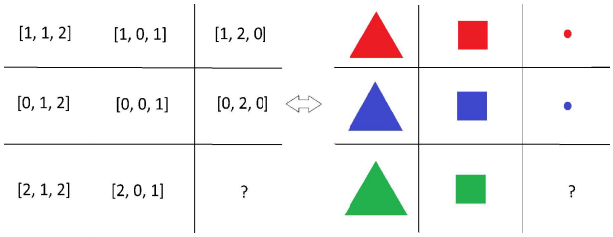

Fig. 1: An example of the Raven's Progressive Matrices test. The right side is the visual representation (color, size, shape) presented to a human. The left side is the machine-readable equivalent of the test, where each cell is presented as "[color,size,shape]".

5. OpenAI Gym

A popular method for developing AI systems is Reinforcement Learning. This method uses the standard model of an agent interacting with the environment to affect it and receive some positive or negative reward as a result. OpenAI Gym [1] is a testing framework that is designed for reinforcement learning techniques. This framework specializes in testing an AI system's ability to adapt to new situations. It consists of many individual environments with unique characteristics.

Reinforcement Learning environments provide a unique method for testing intelligence, because the environments make no assumptions of prior knowledge. The point of the environment is to train the agent based on whether it is doing good or bad with respect to the environment. This allows us to test for the adaptability aspect of intelligence. While it does provide a valid and reliable test for adaptability, it cannot test well for prior knowledge (across RL environments).

In our work we used the CartPole environment. This environment simulates a cart with a free-rotating pole on it. The goal of the agent is to keep the pole upright. The agent receives four input values, the cart's position and velocity and the pole's angle and angular velocity. Using these inputs the agent can output whether to move the cart left or right in an attempt to keep the pole balanced. CartPole is substantially different from the RPM environment, because CartPole runs over many consecutive iterations. This requires our agent to discover a form of cause and effect relation that would be completely missed in RPM. To score the CartPole environment, we use the duration of how long the agent is able to keep the pole balanced as its score. This time is not based in the physical world, but it is handled by the CartPole simulator. The maximum possible score for an agent in CartPole is 200; this value is a limit set by CartPole environment.

6. Mixed Environment

Using the 3-RPM environment and the CartPole environment, we then created a combination environment. This

environment runs several iterations of both 3-RPM and CartPole for training and testing. The score from this combined environment is determined by the weighted scores of both of its sub-environments. The weights were found by averaging the minimum number of environment interactions our agents took to achieve a certain score threshold. The environment interaction for the CartPole environment is one attempt in the OpenAI Gym simulator. This simulator runs for a variable amount of time, depending on how long the agent can balance the pole or until a maximum duration is reached. The duration is ignored and instead we only examine each action taken in the simulator. For the 3-RPM environment, the action taken is one 3-RPM question.

For 3-RPM this threshold was an accuracy of 0.95 and for CartPole, a minimum score of 190. These values were chosen because they are high values to achieve with respect to the maximum possible values for 3-RPM and CartPole (1.0 and 200 respectively). The high values force an agent to learn how to perform well in their respective environments. The choice of 0.95 of the maximum possible value, outside of it being a large value, was chosen arbitrarily. The average minimum values were found by using the same baseline agent, a neural network topology, for each environment. That is, we trained several random neural networks, with the same topology, until it reached the respective minimum threshold. Using this method we found that our baseline system takes 383 epochs to reach the threshold in CartPole and 118 in 3-RPM. Using Equation (1) and the empirically determined values, the weights for these tests can be found. For each environment μ we plug in the corresponding value and sum of the values over the set of all tests, which in this case is the sum of our two values. From this method, the weight on the CartPole environment is 0.76 and 0.24 for the 3-RPM environment. These values will then be used to scale the final score for the mixed environment.

7. Trained AI systems

We trained four unique agents, one for each of the environments (3-RPM, 6-RPM, CartPole, Mixed). The 3-RPM agent was trained for 1000 epochs over 90% of the 3-RPM problem set, approximately 1500 tests. We similarly trained the 6-RPM agent for 1000 epochs over 90% of the 6-RPM problem set, 9000 tests. The CartPole-agent was trained over 2000 interactions in the CartPole simulator. The mixed-agent was trained for 1000 epochs over one interaction in CartPole, 3-RPM, and 6-RPM, in that order. We can see from Figure 6 these are adequate choices for the number of epochs to use in training our agents.

Each agent was given unlimited training time to meet the epoch amounts. In the future, it may be worthwhile to measure the amount of training time, either physical or simulated, which can used as a metric for intelligence. A more intelligent system would require less time than another system but achieve similar performance. This characteristic would then be used in creating a more rigorously defined baseline system to evaluate the sub-tests used in this testing framework.

We decided to utilize artificial neural networks as the basis for all of our agents. This type of agent was chosen because neural networks are easy to implement and to use for solving a large class of diverse problems. The structure of the network is 48 input nodes, 4 fully connected hidden layers of 16 nodes each and an output layer of 6 nodes. For tests not including the 6-RPM environment, we changed the network structure to have 24 input nodes and 3 output nodes. Each layer is dense and implements the ReLU activation function. This topology was not tuned for any of our problems and was arbitrarily chosen. The agents were created using the Keras package [2]. This package runs on top of TensorFlow to allow for simple neural network creation.

We ran the trained agents on each of the environments. To allow for maximum compatibility between the environments, we set the input and output dimensions to the maximums found in the three environments. When the agents did not receive a full set of inputs, the rest of its inputs were padded with zeros. Similarly if the agent outputs too many results, the excess results were ignored. The raw scores for the 3-RPM and CartPole environments were scaled by the maximum possible value. The mixed environment score was scaled by weighting the scores of its sub-environments in terms of the difficulty of the environment and then adding that value to the final score. This was done to take the difficulty of the environment into consideration; a harder environment should be weighted more heavily.

8. Test Verification

We have proposed utilizing two diverse tests as a basis for our examination on combining tests. Here, we attempt to justify the claim that these tests evaluate the fundamentals of intelligence. Based on the performance of an agent in an environment. If we can objectively determine that these environments measure the key aspects of intelligence, we can assert that these environments are useful for creating a generalized intelligence test as well as having a verified method for choosing future environments.

Our first method involves handicapping our agents. That is, we compare our baseline agent to agents that have reduced neural network topologies. A reasonable assumption is that larger networks are able to achieve greater results when compared to smaller networks. If these varying agents demonstrate this characteristic in these fixed environments, we can assert that the environments are measuring the agents capacity to learn and extrapolate information from training.

We trained both our CartPole and 3-RPM agents using a fixed number of environment interactions (2000 for CartPole and 1000 for 3-RPM). This was done to prevent smaller networks from taking advantage of longer training periods which could lead to an increased score. We then handicapped

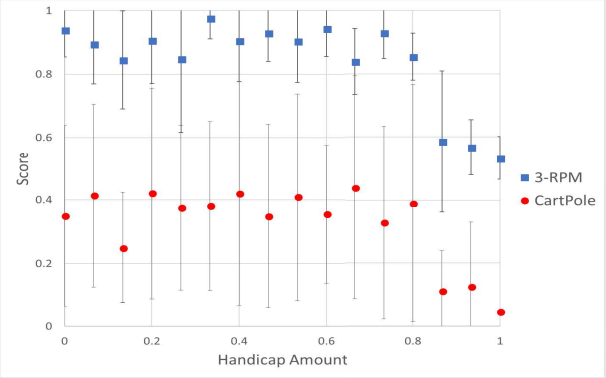

Fig. 2: The effects of handicapping agents. The score of an agent is plotted against the number of nodes removed in both the 3-RPM and CartPole environments. The blue squares show the score in the 3-RPM environment while the red circles show the score in CartPole. The agents were handicapped by removing nodes from their hidden layers until only one node remained in each layer. Error bars shown are for one standard deviation.

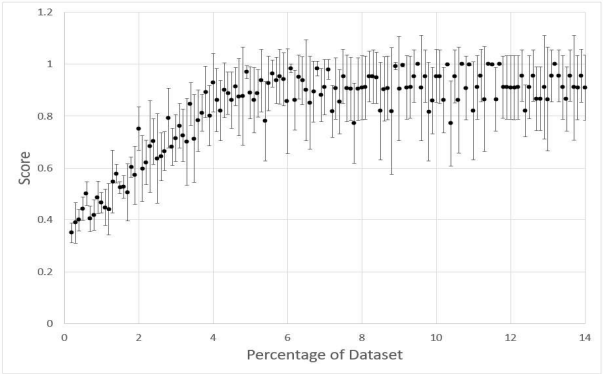

Fig. 3: The baseline agent's performance as the amount of training data from the 3-RPM environment was varied. The training data varies from 0 to 13% (200 tests) of the possible tests from the 3-RPM problem set. A logarithmic trend-line is used to visualize the agent's increase in performance as the number of training samples increased.

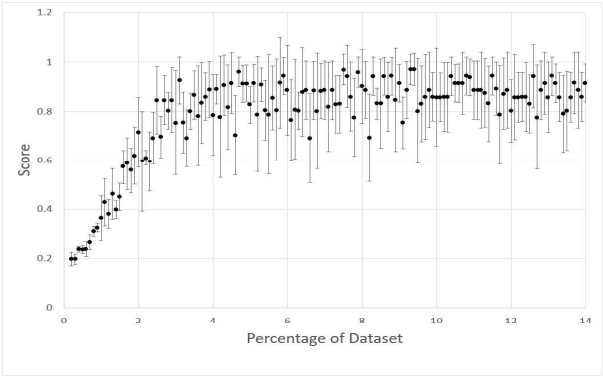

Fig. 4: The baseline agent's performance as the amount of training data was varied. The training data varies from 0 to 14% (1400 tests) of the possible tests from the 6-RPM problem set. From this we can see that the agent requires many more tests to be able to achieve the same score as in the simpler 3-RPM test.

our agents by removing a node from each hidden layer until the number of nodes in each layer reached just one. For each of these varying handicapped agents, we took the average of 20 agents to produce our final scores in the environments. The results can be seen in Figure 2. From the data, we can see there is no noticeable change from no handicap to a handicap of 12 (80%) nodes removed. However, there is a very sharp drop off when we increase the handicap from 12 to 13. This seems to be a critical point for both of the agents in which their ability to train and process information is significantly reduced.

Another similar method is to vary the amount of training data used. The amount of training data used was varied from 0 to 13% (200 tests out of 1500 possible) of the total set of our 3-RPM problems. The results for this can be seen in Figure 3. From these results we can confidently determine that this test is not very hard to generalize. The agent needs to train on only 10% of the total set in order to perform well on the entire set. We also tested the agent for all percentages up to 100%. This was done to see if the agent would ever become over fit and thus would not generalize to the rest of the set. The agent never became over fit because it achieved a near perfect score for all percentages greater than 10%.

We wanted to examine if the number of epochs could be potentially over fitting our agent. To test for this, we varied the number of epochs used to evaluate our agent. The data used in this experiment was fixed with a training percentage of 90% and a testing percentage of 10%. From these results we found there is a very sharp increase in performance over 50 epochs. Beyond 50 epochs there is no indication of over fitting the agent for to the environment. We tested up to 1000 epochs with no significant variance at any point.

In all of these methods for evaluating the 3-RPM problem, we have determined that the problem is too simple. To correct this we have created a harder version of the 3-RPM problem called 6-RPM. We repeated the same experiment using the 6-RPM environment. Still following the 10% holdout and 90% training data as before, we re-evaluated the environment using the same handicapping measure as before. The results can be seen in Figure 5. From the graph we can see a much more linear decrease in score as the handicapping amount was increased.

9. Results

The results of running this testing framework on the trained systems are shown in Table 1 and Table 2. The

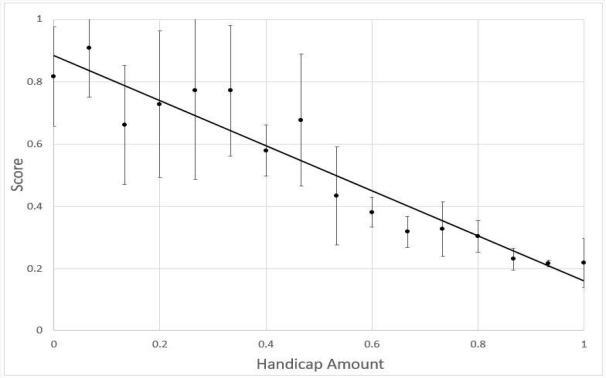

Fig. 5: The effect of handicapping an agent on the 6-RPM test. This is identical to the previous handicapping example, but with a new test. The handicap starts at 0 and progresses by removing a single node from the agent's hidden layers until only one node remains. When this value is reached the agent is maximally handicapped.

difference between these two tables is that Table 2 includes the 6-RPM as another sub-test and that a different baseline system was used to generate the results for the tables. The results from these two tables are similar in that each agent achieved the best score in each of its specific environment. The Mixed and Random agents also performed similarly with the addition of the 6-RPM environment. This shows that our testing framework is consistent even if another sub-test is added to the combined test.

Using our previously defined notation, we can get the intelligence of an agent Υ by examining the Mixed score in Table 2. This value represents the absolute intelligence of the agent with regards to these sub-tests. Ultimately, the combined test will include more sub-tests in order to increase the scope of intelligence that we are measuring. The intelligence being measured now is just the system's ability to perform on RPM and CartPole. But these initial results show that such a combined test is a valid framework for evaluating the intelligence of AI systems.

The CartPole agent performed the best in the mixed environment. This is due in part to the weights that are attached to each environment. Since the CartPole environment is weighted significantly more than the 3-RPM environment, we would expect good CartPole players to receive a greater score in the mixed environment. Another significant factor is that there were no other good CartPole players among our agents. While the 3-RPM agent and mixed agent performed significantly better than our random agent, they did not achieve high enough scores to beat the CartPole agent. Our mixed agent scored as well as the random agent for 3-RPM but did manage to achieve a significantly better CartPole score. This result is odd as our environments seem to be associated with each other. They are associated because training an agent on one environment will result in a higher score on the other environment. Yet, training an agent on both environments will lead to a decrease in both environment's scores. We expected the environments to be different enough that the specialized agents would achieve a minimum score on the other environment or that the environments were similar enough that the mixed agent could learn both well. This result shows that there is a need to measure the similarity between tests. If two tests are dissimilar, then the AI system should be given a higher intelligence score for performing well on both of them.

We examined the convergence of each agents' weights (the weights used in the neural network) to determine whether the mixed agent was reaching a solution. From Figure 6, we can see that both the 3-RPM agent and the CartPole agent are converging on an optimal solution, but this is not true for the mixed agent. We think that training the two different environments on one agent may be causing this lack of convergence. Since our mixed agent is not reaching an optimal solution, we should not expect it to be able to score nearly as well as the agents that reach a solution.

A significant difference between the two tables is the decrease in score in the CartPole environment. This uniform decrease is shown across all of our agents except for the random agent which maintained a similar score. We believe this was due to the changing of our baseline system to allow for the much larger 6-RPM problem to be used. This result shows that the baseline system chosen can largely affect the results of these environments. It is necessary to compare environments using a single baseline system and making comparisons based off of multiple systems can lead to skewed data.

Trained Agents	3-RPM Score	CartPole Score	Mixed Score
3-RPM Agent	0.820	0.448	0.536
CartPole Agent	0.437	0.703	0.633
Mixed Agent	0.332	0.314	0.321
Random Agent	0.333	0.048	0.114

Table 1: The score of each trained agent on each of the available environments.

Trained Agents	3-RPM	6-RPM	CartPole	Mixed
3-RPM Agent	0.976	0.213	0.078	0.245
6-RPM Agent	0.944	0.853	0.119	0.433
CartPole Agent	0.525	0.168	0.225	0.194
Mixed Agent	0.750	0.167	0.180	0.220
Random Agent	0.333	0.162	0.049	0.177

Table 2: The score of each trained agent on each of the available environments. The 6-RPM environment was included for the mixed test; as a result the baseline model used to evaluate each environment was changed.

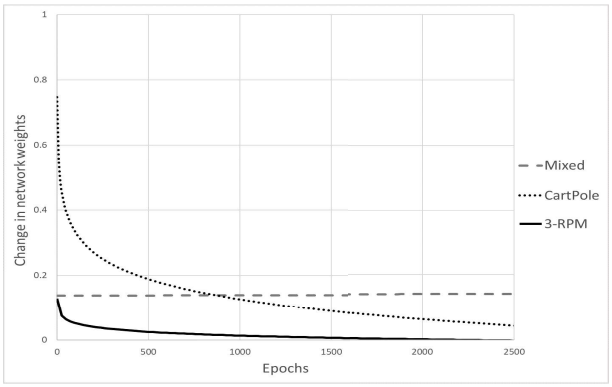

Fig. 6: The convergence of an agent's weights as the number of epochs increases. The lines shown are the trend lines of the change in weights. Each data set was scaled by the maximum value in the set to allow for a better comparison between the agents. The solid line is the 3-RPM agent, the dashed line is the mixed agent, and the dotted line is the CartPole agent.

10. Conclusion

A framework which incorporates all the best aspects of these tests is needed before we can evaluate an AI system for general intelligence. Not only would we be able to evaluate for general intelligence, but we would also be able to measure performance across a wide breadth of possible environments. This could lead to more sophisticated AI systems being created since there would be a standardized comparison measure. The framework would need to allow for a variety of AI implementations to be measured and compared.

From these results it can be seen that adding another sub-test to the combined test does not appear to make significant changes to the results. This shows that the framework can be used to generalize testing to include more sub-tests that can further measure the general intelligence of a system. Results also show that a single baseline system must be used to evaluate the environments to prevent skewed data. Once a single baseline system is determined the consistency of the sub-tests can be found.

Ideally, the speed of the AI system's solution would also be used as a performance measure, where faster results are given better scores compared to slower results. While this work did not explore the aspects of time-based performance measures, they are certainly needed to fully measure an AI system. These time dependent tests would be independent of the hardware the system is running. To find out the hardware speed, several speed tests would need to be administered before the performance could be measured. If we then multiply the time it took for the AI system to complete the test by the inverse of the hardware speed, we should obtain a hardware-independent time measurement. This idea is addressed in depth by [4]. Alternatively, we could constrain the test environment to a specific hardware configuration so that all competing systems are on a level playing field in terms of computational resources.

We have shown a few novel methods for verifying tests used in the evaluation of AI systems. An ideal test would have a linear increase in difficulty for an agent that is handicapped linearly. Additionally a reliable test would require an agent to learn from many examples. To verify future tests using these same metrics, we need to expand our definition of a standard agent as one that can respond to a more diverse set of tests while still being a system we can handicap and vary the amount of training data.

References

[1] Greg Brockman et al. "OpenAI Gym." In: *CoRR* abs/1606.01540 (2016).

[2] François Chollet et al. *Keras*. `https://github.com/keras-team/keras`. 2015.

[3] José Hernández-Orallo. "Beyond the Turing Test." In: *Journal of Logic, Language and Information* 9.4 (Oct. 2000), pp. 447–466. ISSN: 1572-9583.

[4] José Hernández-Orallo and David L. Dowe. "Measuring universal intelligence: Towards an anytime intelligence test." In: *Artificial Intelligence* 174.18 (2010), pp. 1508–1539. ISSN: 0004-3702.

[5] Javier Insa-Cabrera, David L. Dowe, and José Hernández-Orallo. "Evaluating a Reinforcement Learning Algorithm with a General Intelligence Test." In: Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 1–11. ISBN: 978-3-642-25274-7.

[6] Shane Legg and Marcus Hutter. "A Collection of Definitions of Intelligence." In: Amsterdam, The Netherlands, The Netherlands: IOS Press, 2007, pp. 17–24.

[7] Shane Legg and Marcus Hutter. "Universal Intelligence: A Definition of Machine Intelligence." In: *Minds and Machines* 17.4 (Dec. 2007), pp. 391–444. ISSN: 1572-8641.

[8] John Raven. "The Raven's Progressive Matrices: Change and Stability over Culture and Time." In: *Cognitive Psychology* 41.1 (2000), pp. 1–48.

[9] Pritika Sanghi and David L Dowe. "A computer program capable of passing I.Q. tests." In: *4th International Conference of Cognitive Science* (2003).

[10] R.J. Solomonoff. "A formal theory of inductive inference. Part I." In: *Information and Control* 7.1 (1964), pp. 1–22. ISSN: 0019-9958.

[11] Alan Turing. "Computing Machinery and Intelligence." In: *Mind* LIX.236 (1950), pp. 433–460.

[12] C. S. Wallace and D. L. Dowe. "Minimum Message Length and Kolmogorov Complexity." In: *The Computer Journal* 42.4 (1999), pp. 270–283.