Measuring the Relative Similarity and Difficulty Between AI Benchmark Problems

Christopher Pereyda, Lawrence Holder
School of Electrical Engineering and Computer Science
Washington State Universeity
Pullman, WA
{christopher.pereyda, holder}@wsu.edu

Abstract

There has been an explosion of challenge problems, algorithmic tests and datasets for evaluating AI systems. Yet no methodology exists to objectively measure either the collective difficulty of these problems or their similarity. This is an obstacle to creating more general AI systems. We propose a theory for measuring the similarity between pair-wise problems. We evaluate this theory by utilizing a methodology based on a deep neural network to objectively measure these properties between test problems using foundational datasets. An implementation of these methods is then used to measure the similarity between well known datasets. Results show that the proposed measure successfully identifies the difficulty and similarity among problems. This can be used to ensure diversity in test suites used to evaluate AI systems.

Introduction

With the rise of fundamental datasets, AI competitions, and big data, we are faced with ever more diverse and challenging problems. Many of these tasks can be solved through the use of an AI system. These systems can achieve high performance with little fine-tuning or domain expertise. Yet the best solutions are most often highly specialized for the specific problem. These specialized solutions to the numerous problems present two significant challenges for the AI community to address.

First, there is no measure of distinctiveness. With all of these varying challenges, little thought is given to the distinctiveness of each. Are image classifications problem all effectively the same? Are reinforcement learning problems significantly different from theorem proving? Some AI competitions are being developed with the goal of creating more general AI systems to solve them (Perez-Liebana et al. 2016). Yet even these competitions are generally bound to a niche area and topic. As the number of AI challenge problems increases, we need a measure of their distinctiveness in order to better assess the generality of the AI systems applied to these problems.

Second, there is the lack of generality among AI systems. This lack of generality exists both on the system side and the test side (Langley 2006). Many currently offered tests do not aim to combine different and varying aspects of intelligence, but simply challenge one particular aspect. This limits the types of AI systems that can be effectively evaluated and studied and encourages the exact opposite of generalized systems.

One proposed method for measuring the generality of AI systems is the AIQ test (Pereyda and Holder 2018). The Artificial Intelligence Quotient (AIQ) test utilized a small set of active environments to measure the intelligence of a given AI system. While the preliminary results were promising, more work was left to be done. One key idea that was not expanded upon was the notion of similarity. The theory the authors drew from, created by Legg and Hutter (Legg and Hutter 2007), utilized a method of sampling from the set of all possible tests. Sampling in this way is not feasible for a practical intelligence measure. If similarity can be well defined, we can utilize the theory while maintaining feasibility.

With these challenges in mind, we hope to push towards more generalized AI systems and encourage more development and research into this developing field. We begin by examining related work that provides concepts of similarity measurement. We then propose a theoretical basis for creating a similarity measure. We show the results of several experiments evaluating the theory and exploring the parameters of the various problems. We conclude with a final similarity metric and utilize this measure on several well known problems.

Related Work

The developing field of general AI testing has shown interest in achieving more rigorous metrics and applicability (Hernández-Orallo 2017). Much work has been done to construct better theories (Hernández-Orallo 2010) and to implement these ideas (Insa-Cabrera, Dowe, and Hernández-Orallo 2011). One method to avoid the limitation of utilizing a similarity metric was constructed by (Legg and Veness 2013). Instead of examining pair-wise similarity, they examine an even distribution of test problems drawn from the set of all possible tests. The difficulty can then be measured as a function of their AI system's score, and from this difficulty they can measure the distinctiveness of their sample of tests from the whole. These test problems were generated by creating random programs and converting these to reinforcement learning problems. This is infeasible for practically measuring AI systems for two reasons. First, the method involves creating a set of tests with hundreds of thousands of random samples. Training and testing state of the art AI systems on this many tests requires too much time to be practical. Second, the tests themselves generally do not have any practical value. These tests are generated at random, so the tests have little relation to the real-world. This is significant because most AI systems are targeted toward real-world problems.

These two weaknesses can be avoided with the use of a similarity measure. The test sets can be generated from the set of currently available AI challenge problems. If we can measure the similarity between the tests, we can effectively weight each test according to how different it is from the rest. This weight was constant in Legg and Veness's (2013) work due to the random nature by which tests were selected. Utilizing real world tests will introduce meaning into the set, which will give an effective measurement for practical AI systems.

The AIQ framework (Pereyda and Holder 2018) relies on the theory of Universal Intelligence created by Hernández-Orallo and Dowe (Hernández-Orallo and Dowe 2010). In the AIQ framework the similarity of tests were measured by training an AI system on one test and then evaluating how well it performs on another test. While simplistic, this offers another method to measure the difference between two tests. This method utilizes the idea that similar tests both have similar input-output schemes and have similar concepts that an AI system can extract. For example, an AI system trained to play a certain genre of video game should do better on video games from that genre than AI systems that were trained on different genres. However, when test problems are diverse (e.g., cart-pole reinforcement learner attempting image classification), the AI system can have significantly poor performance, which can result in unreliable measurements of problem similarity. In this work, we improve the reliability of measuring test similarity in such contexts.

Some related work has been done in terms of measuring similarity between tests in the area of transfer learning (Gorski and Laird 2006). The ideas utilized in transfer learning are very similar to what we propose as a similarity measure. They aim to take an AI system trained on a test and apply it to a similar test such that information gained from its initial training transfers to the subsequent test. This is another possible similarity metric between tests. Some tests have been created to measure this similarity (Cook, Holder, and Youngblood 2007). Instead of examining the tests one after the other, the approach proposed here will allow us to examine a mixture of the tests.

Theoretical Framework for Test Similarity

As part of the AIQ framework in (Pereyda and Holder 2018) the IQ of an AI system was defined in terms of its performance on a set of tests. However, the IQ score was not weighted based on the diversity of the set of tests, where diversity can be measured as the average pair-wise dissimilarity between tests in the set. An AI system designed for image classification will do well on a set of image classification tests, but this set is not as diverse as, say, a set of tests consisting of image classification and reinforcement learning problems. Therefore, we seek a measure of similarity (or dissimilarity) between tests that can inform a diversity measure of a set of tests.

In this section we derive a measure of similarity between two tests. Suppose we have two tests, A and B. Let us assume that test A is easier than test B. That is:

$$V_A^\pi > V_B^\pi \tag{1}$$

where V_t^π is the expected performance from a baseline agent π on test t. Based on this scenario a simplistic measure of the dissimilarity is the difference between the two values $|V_A^\pi - V_B^\pi|$. As mentioned earlier, this simplistic difference may suffer when the tests are too hard or too easy, in which case the difference is near zero, even for tests taken from very diverse domains. For a better measure, we make the assumption that π is sufficient to perform A and B independently. We discuss a *sufficient* agent in a later section, but essentially it means that the agent's performance on a merger of tests A and B is in between the agent's performance on A and B individually (i.e., $V_A^\pi \geq V_{AB}^\pi \geq V_B^\pi$). Two tests can be merged in many different ways, and we discuss proper techniques for merging two tests in a later section.

Given Equation 1 and our sufficiency assumption on agent π, in the worst case π perform at the level of the harder test B. If we have a merged test consisting of only the harder test B, then π will learn B achieving performance V_B^π. If the merged test contains some problems from the easier test A, then π will achieve a level of performance that lies between V_B^π and V_A^π. So π will never perform worse that it did on test B given any split of data.

If A and B have some degree of similarity, then we can plot the performance achieved on the merged test AB, as a function of the proportions of A to B used to generate AB. We can plot the horizontal line V_A^π to represent the maximal performance and plot the horizontal line V_B^π to show minimal performance (see Figure 3). We can then define similarity $S(A, B, p, \pi)$ between two tests A and B as a function of the distance between the performance achieved by baseline agent π at a certain proportion p of problems from A and B in the merged test AB.

$$S(A, B, p, \pi) = |V_A^\pi - V_{AB}^\pi(p)| - |V_B^\pi - V_{AB}^\pi(p)| \tag{2}$$

Figure 3 shows an example of these components of the similarity between two tests. In our experiments we chose an equal proportion of problems from the two tests being compared (i.e., $p = 0.5$). Trying other values for p, or optimizing p to maximize or minimize similarity, is a direction for further research.

Equation 2 does not take into consideration the relative scale of the performance. It would disproportionately weight larger absolute gaps between the performance results rather than examining the relative performance. So we normalize Equation 2 to arrive at our final definition of similarity:

$$S(A, B, p, \pi) = \frac{|V_A^\pi - V_{AB}^\pi(p)| - |V_B^\pi - V_{AB}^\pi(p)|}{|V_A^\pi - V_B^\pi|} \quad (3)$$

Merging Tests

The sufficiency of an agent, and thus the overall similarity measure in Equation 3, are dependent on the method $m(A, B)$ for merging the two tests A and B being compared. We consider two different methods for merging tests: *stitching* and *sampling* (refer to Figure 1). First, we can merge the tests element-wise, by creating larger elements from elements in A and B. We call this the "stitched" or "stitching" merge method. For example, merging elements from two tests, where each element is a 100x100 image, would yield stitched elements of 100x200 images (see Figure 1). The inputs to the agent are increased to accommodate the larger problem, and the outputs would also increase to include outputs for both problems (e.g., two different classifications of the two different images). Second, we can combine the tests set-wise, by taking a sample of elements from A and B. We call this the "sampled" or "sampling" merge method. For example, merging elements from the two image-based tests yields a set of images consisting of samples from each of the two tests (see Figure 1). In this case, the inputs to the agent increase to the larger of the two individual problem's inputs and unused inputs are padded with zeros in the test data, similary for outputs.

As can be seen in Figure 2, the choice of how to merge the two tests has an impact on the sufficiency of the baseline agent. In the figure we see that the sampling method (A,B) leads to a sufficient agent, while the stitching AB method does not. There are other methods for merging two tests in terms of how the inputs and outputs are represented, especially for tests requiring more than a single class or action response (e.g., blocks-world planning). But note that our goal here is a sufficient agent to support a consistent, relative measure of test similarity, not the best possible performance on any one test.

Experiments

We conduct three experiments to validate the proposed similarity measure. First, we compare different image classification problems that are targeted toward deep neural network learning systems. Second, we compare several different problems from the OpenAI Gym (Brockman et al. 2016) that are targeted toward reinforcement learning systems. And third, we mix problems from the first two sets to create a more diverse test suite. For each set of tests, we determine a viable method for merging problems, construct a sufficient learner, and evaluate the similarity and difficult of the tests.

Image Classification Tests

For this experiment we utilize four well known datasets: MNIST (LeCun et al. 1998), FashionMNIST (Xiao, Rasul, and Vollgraf 2017), CIFAR10, and CIFAR100 (Krizhevsky 2012). These tests were chosen due to their widespread use

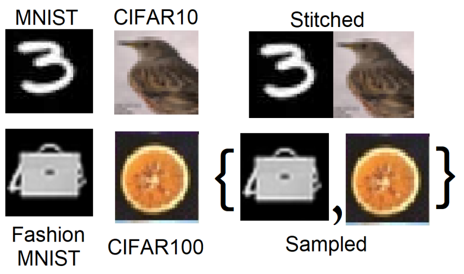


Figure 1: Example images from the four datasets used in the experiments. The right side shows two methods for merging the individual tests into a combined test.

Table 1: Validation accuracy of deep neural network on image classification tests.

Test	Description	Accuracy
MNIST	Handwritten Numbers	0.9932
Fashion-MNIST	Clothing	0.9251
CIFAR10	Real Images	0.8099
CIFAR100	Extended CIFAR	0.4892

in the AI community and because they have similar properties. Examples of the datasets can be seen in Figure 1.

We chose a deep neural network as the baseline agent π for these tests. The reason was two-fold: we expect the neural network to be the most generalizable of agents, and we also expect the neural network to have the ability to extract key concepts from the tests and thus better handle a combined test.

We implemented the deep neural network using Keras (Chollet and others 2015). This network utilized several repeating layers of increasing CNNs into a final flattened layer of fully connected nodes for the output. This model was largely influenced by Machine Learning In Action (Kumar 2018). The full model can be seen in Figure **??**. The model was trained over 15 epochs for each problem. We used Adam optimization and categorical cross entropy as the loss function. The results of training the neural network on the standard datasets can be seen in Table 1. The results show that the network performs well on the MNIST and Fashion MNIST datasets, but worse on CIFAR10 and CIFAR100. We could improve these results by further optimizing the network or specializing it to each test, but our goal is to provide a sufficient system, not necessarily the best.

Merging Methods Here we discuss the details for merging two image classification tests. Supposing test A has elements of size 32x32, and test B has elements of size 32x32, the "stitched" test AB will have elements of size 64x32. If the elements in the datasets vary in size from each other, we evenly pad the smaller elements with 0 values to reach the size of the larger elements. To detect a possible bias in this approach we also create combined tests by stitching each individual test to itself: tests AA and BB. These tests are MNIST data stitched to itself (AA) and CIFAR10 data stitched to itself (BB). Comparing the test AB to AA and to BB instead of the simple A and B tests will help identify additional components of complexity introduced by the merging process that could bias the results. For example, the modified input and output spaces may affect the results. Or changes in the image representations could also affect the results. This could be due to MNIST data consisting of mainly a black background and CIFAR10 data consisting of a noisy background.

The second method of merging tests is the "sampling" method. For example, if test A has 50,000 elements and test B has 50,000 elements, then test (A,B) will have 50,000 (25,000 from A and 25,000 from B). In the case where the sizes of the datasets do not match, we reduce the size of the larger dataset to match the size of the smaller dataset. This was done to decrease any class imbalance that might influence the final results. In this method we employ the use of the proportion variable p, which represents the proportion of samples drawn from A to samples drawn from B. We will examine how different values of p affect the results of the sampling method. In the case of combining the output classes, we set the extra outputs to zero. That is, the system must be able to individually classify elements from both tests used to make up (A,B). That is, if test A has 10 classes and test B has 100 classes, then test (A,B) will have 110 classes.

Baseline Agent Validation First, we seek to validate the baseline neural network agent and evaluate the approaches to merging two tests. We want to verify that the neural network is sufficient for our datasets. If the neural network is either too good or too poor at these classification problems, we would have to adjust the neural network until it performed adequately. Second, we want to examine how the two methods of test merging perform. Since sufficiency depends on the performance on the merged tests, we want a merging method that does not significantly modify the inherent difficulty of the individual tests. Lastly, we want to see how varying the complexity of the neural network affects the expected performance. From this data we can conclude how the complexity of the baseline agent will affect the final similarity metric.

We begin by training the network on MNIST (test A) and CIFAR10 (test B), while varying the total number of parameters in the network (trainable and un-trainable). This is done by reducing the depth of the filter in each convolution layer by half. We begin with 32x32 dual convolution layers into a maxpool. We repeat this for three times, then feed the results into a dense layer with ReLU activation. The network architecture remains the same throughout the process. The results of this experiment can be seen in Figure 2. From these results we can see that the neural network is sufficient to learn MNIST and CIFAR10. That is, the neural network does not learn both tests A and B with 100% accuracy and does not fail to learn either. We can see that for the value $p = 0.0$, the performance on (A,B) drops slightly below the lowest performance B. This is largely due to the volatility of the network. The results also show that utilizing the stitching method (AA, BB, AB) violates the sufficiency constraint that the merged test should lie between the subtests use to make it. So the stitching method is not adequate

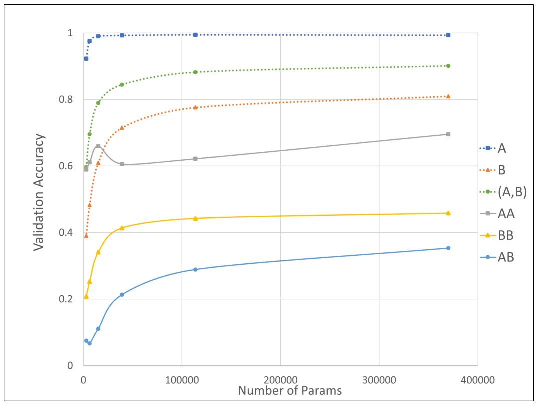


Figure 2: Validation accuracy vs. the complexity of the neural network. We varied the number of parameters in the network by halving the depth of the filters in each convolution layer. Test A is the MNIST dataset. Test B is the CIFAR10 dataset. Test AA, BB, and AB were created by stitching together the sub-tests A and B element-wise. Test (A,B) was created by sampling from both test A and test B.

for measuring the similarity between these problems. However, the sampling merge method does satisfy the sufficiency constraint, and so it can be used in this case.

The experimental results also show that the complexity of the baseline agent is a factor in determining the similarity, but it does not largely affect the results so long as the performance (validation accuracy in this experiment) decreases or increases at the same rate as a function of the complexity of the baseline agent.

Next we examine how the proportionality affects the similarity measure. We start with MNIST and CIFAR10 as test A and B respectively. We vary the proportionality of A and B used to make up the combined test (A,B). At $p = 0.0$ we have entirely test B used to construct the combined test (A,B). At $p = 1.0$ we have only test A in (A,B). At $p = 0.5$ there is an even split of A and B in (A,B). The total number of training samples was kept constant at 50,000. Similarly, the number of test samples was 10,000. The results of this experiment can be seen in Figure 3.

Similarity Measurements Finally, we evaluate our methodology for measuring the relative similarity between the four datasets. This is done by training the sufficient neural network on the four individual tests, the four tests representing a test merged with itself, and the six pair-wise merged tests. The resulting validation accuracy for each test after training is input to Equation 3 to give a final similarity measure shown in Table 2. We can see that for these tests the similarity metric performs as expected when tests are compared to themselves. CIFAR100 and CIFAR10 are the next most similar pair. This is expected due to their image types being relatively similar.

A dissimilarity graph can be seen in Figure 4. The data shown is the dissimilarity metric $(1 - S)$, the further the points are away from each other, the more dissimilar they are. We then attempted to create the graph utilizing to-scale

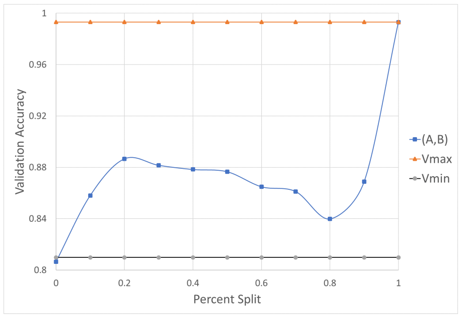


Figure 3: Validation Accuracy vs. Percent split. We vary the proportion of the sub-tests used to generate the merged test. MNIST is test A and CIFAR10 is test B. The merged test is generated by the sampling merge method. Vmax is the performance achieved by the neural network on test A. Vmin is the performance achieved by the neural network on test B.

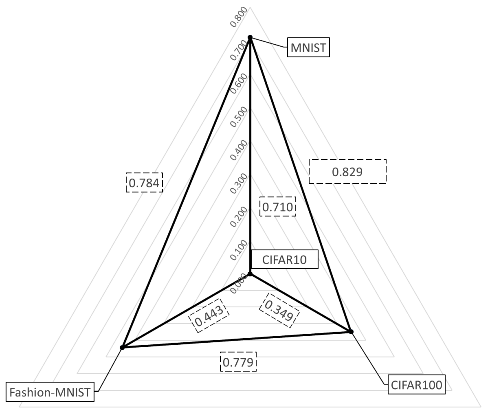


Figure 4: A graphical representation of the dissimilarity between the four tests.

line segments. While not entirely to scale, this graph still shows dissimilarity using relative distances. The grey triangles show the distance from any of the tests to the center test (CIFAR10). These line segments are drawn to scale and the other lines were drawn to fit these points.

From this graph we can conclude a few different aspects of the tests. First, the similarity between the tests follows our intuition. That is, CIFAR100 would be the most similar to CIFAR10 and would be dissimilar to the others. Our intuition that MNIST would be most similar to Fashion-MNIST was incorrect, in that, MNIST is most similar to CIFAR10, but relatively dissimilar to all the other tests. This may be due to Fashion-MNIST and CIFAR10 containing more complicated information. It could also be that the data in Fashion-MNIST emulates more real-world images which is a key component of CIFAR10.

OpenAI Gym Tests

The tests utilized in this experiment were gathered from the OpenAIGym Framework (Brockman et al. 2016). The

Table 2: Similarity measurements between the four tests.

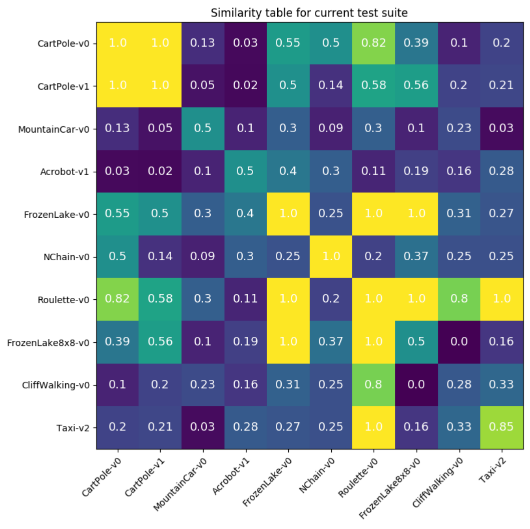
	MNIST	F-MNIST	C10	C100
MNIST	1.0	0.216	0.290	0.171
F-MNIST	0.216	1.0	0.557	0.221
C10	0.290	0.557	1.0	0.651
C100	0.171	0.221	0.651	1.0

tests include the classic control problems (CartPole-v0, CartPole-v1, MountainCar, Acrobot) and toy text problems(FrozenLake, FrozenLake8x8, Roulette, NChain, Taxi, Cliffwalking). The classic control problems are environments setup in a simplified physics engine. These require the agent to be able to learn how to interact with the specified object within this real-world scenario. For example, in the MountainCar environment the agent must learn to push and pull the car over the mountain. The toy text problems are simple text-based games. The agent is given text observations that correspond to the simulated grid world.

Merging Tests The merged tests were created by drawing an evenly-distributed sample of problems from the two base tests. When the test input and output spaces were not equivalent, the smaller spaces would be padded with zeros so they were the same size. When too many outputs were given as actions to test environment, the extra spots would be truncated. So a minor additional challenge to the combined test is that the neural network has to figure out which outputs matter for which test.

Baseline Agent The neural network baseline agent utilized for this experiment is different from that used for image classification. The input to the network is fed into a 16 node dense layer, then three layers of convolutions with 64 filters with a decreasing number of kernels (8,4,2) with ReLU activation. The network finished with a 64 dense layer into the desired output size. The only differences the architecture had between tests was the number of connections from the input and output layers, due to input and output sizes varying. The network was trained for 10,000 steps. Each test instance has between 10 and 500 steps. Training examples for the network consist of a sequence of steps (actions taken) and the resulting reward (as determined by the test environment). When the merged test is given to the neural network, random samples from both base tests are drawn evenly. The sample environment is given as a singular instance to the network to train over. The network does this until 10,000 steps have been reached.

Similarity Measurements A heatmap of the similarity between the OpenAI Gym tests is shown in Figure 5. The values show how similar each test is compared to every other test. Ideally, tests would be perfectly similar to each other along the diagonal. This is true for only half of the tests. We believe this is due to some of the environments being too complex for our baseline agent to handle. We did not develop the baseline agent necessarily perform well over all the chosen tests. Instead this agent was chosen due to its simplicity and ability to handle most of the simpler tests. If the



	CartPole-v0	CartPole-v1	MountainCar-v0	Acrobot-v1	FrozenLake-v0	NChain-v0	Roulette-v0	FrozenLake8x8-v0	CliffWalking-v0	Taxi-v2
CartPole-v0	1.0	1.0	0.13	0.03	0.55	0.5	0.82	0.39	0.1	0.2
CartPole-v1	1.0	1.0	0.05	0.02	0.5	0.14	0.58	0.56	0.2	0.21
MountainCar-v0	0.13	0.05	0.5	0.1	0.3	0.09	0.3	0.1	0.23	0.03
Acrobot-v1	0.03	0.02	0.1	0.5	0.4	0.3	0.11	0.19	0.16	0.28
FrozenLake-v0	0.55	0.5	0.3	0.4	1.0	0.25	1.0	1.0	0.31	0.27
NChain-v0	0.5	0.14	0.09	0.3	0.25	1.0	0.2	0.37	0.25	0.25
Roulette-v0	0.82	0.58	0.3	0.11	1.0	0.2	1.0	1.0	0.8	1.0
FrozenLake8x8-v0	0.39	0.56	0.1	0.19	1.0	0.37	1.0	0.5	0.0	0.16
CliffWalking-v0	0.1	0.2	0.23	0.16	0.31	0.25	0.8	0.0	0.28	0.33
Taxi-v2	0.2	0.21	0.03	0.28	0.27	0.25	1.0	0.16	0.33	0.85

Figure 5: Heatmap showing similarity between OpenAI Gym tests. Lower triangle left empty.

environment is too hard then the similarity metric will have too much error within it. We can see that similar tests do score a high similarity, CartPole-v0 compared to CartPole-v1 and FrozenLake compared to FrozenLake8x8. We can also see that toy text environments are more similar as a group compared to the classic control environments. This might be due to the agent having to learn one specific aspect of the physics engine and not being able to adapt that well to the other environments.

Mixture of Tests

In this experiment we measure the similarity between tests drawn from active and passive environments. We want to compare a set of relatively similar environments to a very dissimilar environment. To accomplish this, we measured the similarity between a set of passive environments (image classification tests) to an active environment (CartPole test).

First, we converted the CartPole test into a passive environment. This was done by training an agent on CartPole until the maximal score was reached. The agent was then used to classify the ideal action for a set of images of the CartPole environment. The training process is the same now as previously, with the addition of the new classification problem of CartPole.

The evaluation is slightly different now due to the varying rewards between the CartPole and the other classifications tests. First, CartPole is scaled in the same manner for the evalution process. The CartPole images are scaled from 400x600 to 32x32, to match the size used by the image classification tests. Second, the agent is tested on a random sample of problems from the first image classification test and then given an instance of the CartPole test. The performance is determined based on the image classification accuracy plus the normalized length of time the agent is able to keep the pole balanced. The final performance (sum of accuracy and balance length) is also normalized between 0 and 1.

Table 3: Similarity measurements between the four image tests and CartPole (CP).

	MNIST	FMNIST	C10	C100	CP
MNIST	1.0	0.216	0.290	0.171	0.083
FMNIST	0.216	1.0	0.557	0.221	0.080
C10	0.290	0.557	1.0	0.651	0.079
C100	0.171	0.221	0.651	1.0	0.652
CP	0.083	0.080	0.079	0.652	1.0

The results of this experiment can be seen in Table 3, which is the same as Table 2, but an extra row and column for the CartPole similarity measurements. The baseline agent was able to learn CartPole with an effective accuracy of 0.10. This means on average the cart was up for 20 steps. This baseline agent performed better than a random agent, which achieves a score of around 0.063, but the baseline agent was not able to achieve high performance (optimal being 1.0). From the additional test of CartPole, we can see it is very dissimilar to the majority of the tests. However, CartPole is surprising similar to the CIFAR100 test. This is most likely due to the complexity of both CIFAR100 and CartPole. A more capable baseline agent would likely rectify this inconsistency. While the image tests are not tightly grouped by similarity, they are closer to each other than to the CartPole test. This meets are intuition of different types of problems being dissimilar.

Conclusion

In this work we have put forward a framework for measuring similarity between different tests. We conducted several experiments to determine what factors may influence the theory in practical ways. We discuss the results of these factors and how they could potentially affect the final similarity metric. We finally construct a practical and effective methodology for measuring the similarity between different tests. We perform this methodology on several well known datasets and the results follow intuition.

One problem with this metric is that it only exists on a relative scale. We still do not have an absolute similarity measurement, even with our methods. One possible solution to this problem is to take many similarity measurements over a maximally diverse set of tests. A way to achieve a maximally diverse set of tests is to create a large, but finite, representation of the set of all possible tests. This has been effectively done by Legg and Veness (2013). They demonstrated a method for generating a set of random tests drawn from the set of all possible tests and then applied an AI system to attempt to solve the test. Utilizing a similar method we could extend this relative similarity measure to be an absolute similarity measure.

We believe that this framework provides a preliminary step in the direction for measuring similarity between AI benchmark problems. This similarity can provide the basis on which to evaluate the diversity of a set of tests and the general intelligence of AI systems based on their performance on these tests.

References

Brockman, G.; Cheung, V.; Pettersson, L.; Schneider, J.; Schulman, J.; Tang, J.; and Zaremba, W. 2016. OpenAI Gym.

Chollet, F., et al. 2015. Keras. https://keras.io.

Cook, D. J.; Holder, L. B.; and Youngblood, G. M. 2007. Graph-based analysis of human transfer learning using a game testbed. *IEEE Transactions on Knowledge and Data Engineering* 19(11):1465–1478.

Gorski, N., and Laird, J. 2006. Experiments in transfer across multiple learning mechanisms. In *Proceedings of the ICML Workshop on Structural Knowledge Transfer for Machine Learning*.

Hernández-Orallo, J., and Dowe, D. L. 2010. Measuring universal intelligence: Towards an anytime intelligence test. *Artificial Intelligence* 174(18):1508–1539.

Hernández-Orallo, J. 2010. A (hopefully) unbiased universal environment class for measuring intelligence of biological and artificial systems. In *Third Conference on Artificial General Intelligence (AGI)*. Atlantis Press.

Hernández-Orallo, J. 2017. Evaluation in artificial intelligence: from task-oriented to ability-oriented measurement. *Artificial Intelligence Review* 48(3):397–447.

Insa-Cabrera, J.; Dowe, D. L.; and Hernández-Orallo, J. 2011. Evaluating a reinforcement learning algorithm with a general intelligence test. In *Conference of the Spanish Association for Artificial Intelligence*, 1–11. Springer.

Krizhevsky, A. 2012. Learning multiple layers of features from tiny images. *University of Toronto*.

Kumar, A. 2018. Machine learning in action. https://github.com/abhijeet3922/Object-recognition-CIFAR-10. Accessed: 2019-04-30.

Langley, P. 2006. Cognitive architectures and general intelligent systems. *AI magazine* 27(2):33–33.

LeCun, Y.; Bottou, L.; Bengio, Y.; Haffner, P.; et al. 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE* 86(11):2278–2324.

Legg, S., and Hutter, M. 2007. Universal intelligence: A definition of machine intelligence. *Minds and machines* 17(4):391–444.

Legg, S., and Veness, J. 2013. An approximation of the universal intelligence measure. In *Algorithmic Probability and Friends. Bayesian Prediction and Artificial Intelligence*. Springer. 236–249.

Pereyda, C., and Holder, L. 2018. Toward a general-purpose artificial intelligence test by combining diverse tests. In *Proceedings on the International Conference on Artificial Intelligence (ICAI)*, 237–243.

Perez-Liebana, D.; Samothrakis, S.; Togelius, J.; Lucas, S.; and Schaul, T. 2016. General video game AI: Competition, challenges and opportunities. In *Thirtieth AAAI Conference on Artificial Intelligence*.

Xiao, H.; Rasul, K.; and Vollgraf, R. 2017. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*.