Opening the Neural Network Black Box: Explorations in Polyomino World

Aishi Huang (aishih2@illinois.edu)
Department of Computer Science, University of Illinois
Urbana Champaign, Champaign, IL 61820 USA

Philip A. Huebner (huebner3@illinois.edu)
Department of Psychology, University of Illinois
Urbana Champaign, Champaign, IL 61820 USA

Jon A. Willits (jwillits@illinois.edu)
Department of Psychology, University of Illinois
Urbana Champaign, Champaign, IL 61820 USA

Abstract

There has been a long debate about whether artificial neural networks represent information in the ways in which it seems that humans do. In this paper, we train simple neural networks to classify objects according to their shape, size, and color. In five experiments, we show that while basic multilayer neural networks easily learn to correctly classify the objects on which they are trained, they show serious difficulties with transfer to novel items. However, our experiments also show that when networks are given further training to respond correctly to the novel items, they learn the novel items much faster than untrained networks. This shows that these networks are learning abstract representations that go beyond the simple items on which they were trained. We argue that this weakens arguments that these kinds of networks are fundamentally incapable of supporting the kinds of abstract representations that it is argued humans must have.

Keywords: neural networks, transfer learning, object recognition, knowledge representation

1. Introduction

Excitement about artificial neural networks, both as a theory of human learning and knowledge representation, and as a form of artificial intelligence, has waxed and waned over time. The initial excitement about Perceptrons (Rosenblatt, 1958) was deflated by Minsky and Papert's (1967) demonstration of the limitations of single layer networks. The excitement over the popularization of a way to train multilayer neural networks (Rumelhart et al., 1985) diminished due to how long these algorithms took to train, and arguments about the kinds of representations that "eliminative" connectionist networks are capable of learning (Marcus, 2001). In recent years, new architectures, training algorithms, and increased computational power have led to neural networks that are increasingly successful at a wide range of tasks, including image classification (Krizhevsky et al., 2012), natural language processing (Devlin et al., 2018), and speech recognition (Graves et al., 2013). These successes have increased interest in, and estimates of plausibility of, neural-network-based explanations of human cognition. However, questions still exist about the fundamental representational capabilities of these models (Doumas & Martin, 2018; Lake et al. 2017; Marcus, 2018), and behavioral research has shown that human behavior often seems to be qualitatively different than the performance of state-of-the-art deep learning models in both vision and language (Geirhos et al., 2018, Linzen et al., 2016).

One major problem with neural network research is that neural networks are viewed as a "black box", with little understanding of precisely how and why they work (or do not work). This limits our ability to evaluate neural networks as models of human cognition, and also limits our ability to design better neural networks for practical applications.

In this paper, our goal was to explore the fundamental capabilities and limitations of simple multilayer artificial neural networks trained to classify the shape, size, and color of simple objects. We designed a set of experiments to really test what they can and cannot learn, and a set of analyses to really understand what they do and do not represent.

The work had three specific sub goals. First, we used a set of experiments designed to see what kinds of transfer of which the networks are and are not capable. Second, we analyzed the hidden states of the networks in order to see whether and how the networks were representing shape, size, and color, in order to better understand representation in the networks. Third, we employed transfer learning (providing additional training for new items to a previously trained network), investigating the ways in which pre-trained networks showed facilitated learning for novel items on which the network had not been trained. As we will describe, facilitated transfer learning can be used to assess the extent to which a network has learned abstract representations that go beyond the items on which it was trained.

1.1. Abstraction and Generalization

There are many criticisms of the representational capacities of neural networks, outlined in the papers mentioned above. In this paper, we will focus primarily on the argument popularized by Marcus (2001, 2018), that neural networks are learning simple input-output mappings, but not learning deeper, abstract knowledge. In Marcus's words, neural networks are incapable of making generalizations "outside of their training set", in a way that seems so effortless for humans. Marcus cites many examples to make this point, but perhaps the most demonstrative example is a model based on a 7-month-old infant rule-learning experiment (Marcus et al., 1999). In this experiment, infants heard sequences of syllables that followed one of three rule-based structures: ABA sequences (where the first and third syllable were the same), ABB sequences (where the second and third were the same), and AAB sequences (where the first and second were the same). Critical for this experiment, infants trained on one rule showed discrimination during a test phase between novel sequences that followed the rule, compared to sequences that violated the rule. This was the case even though the syllables during training and testing phases of the experiment were an entirely different set of syllables. Marcus et al. concluded from this experiment that even 7-month-olds can learn generalizable rules that are independent of the items that they use as the basis of induction for those rules.

Marcus (2001) also used a recurrent network (Elman, 1990, shown in Figure 1) to simulate this experiment, demonstrating that the network did not show generalization like the infants. In Marcus's model, each syllable was represented as both an input and an output unit, and the model was trained to predict the sequences that the infants heard during training. At each time step, the unit in the input layer corresponding to the current syllable was activated, activation was propagated through the network's weights, leading to a prediction about the next syllable in the sequence. The model's weights were trained by comparing each prediction to the actual next syllable in the sequence, and adjusting the weights backpropagation to increase the network's likelihood of making correct predictions in the future. Marcus showed that this network showed absolutely zero generalization to the test sequences on which it was not trained. According to Marcus, the only thing the neural network could do was learn associations between specific syllables. He argued that abstract rules (like ABA) exist at the level of relations amongst variables, and neural networks do not (and cannot) represent variables in this manner.

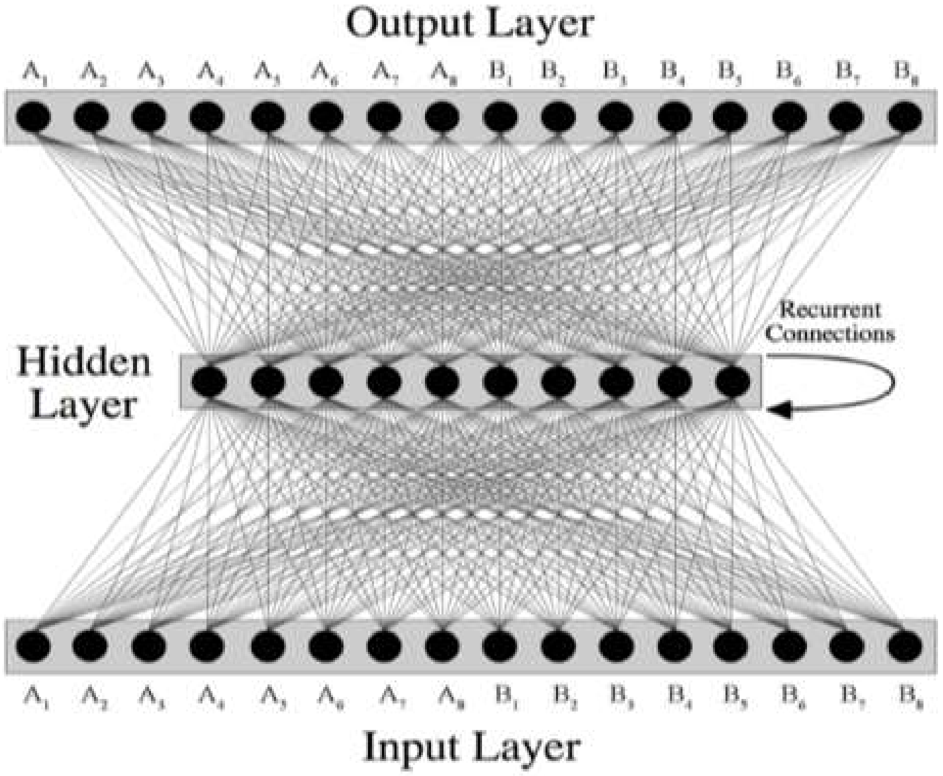

Output Layer
A1 A2 A3 A4 A5 A6 A7 A8 B1 B2 B3 B4 B5 B6 B7 B8
Recurrent Connections
Hidden Layer
A1 A2 A3 A4 A5 A6 A7 A8 B1 B2 B3 B4 B5 B6 B7 B8
Input Layer

Figure 1. Recurrent network used by Marcus to demonstrate the network's failure to learn ABB/ABA/AAB rules.

Why this network *must* fail to show generalization to novel sequences is simple to understand. Refer back to Figure 1. During training it was trained to predict sequences containing the input and output units A1-A4 and B1-B4. During the transfer test phase of the experiment, it was required to make predictions about sequences containing the input and output units A5-A8 and B5-B8. Of course the network failed at this task. It never experienced A5-A8 or B5-B8 during training. What it learned about these weights was that those units never occur and so the weights to those units at the output layer should be zero, so that those units were never predicted.

But this raises a question. Is it really true that the network didn't learn something equivalent to an ABA-style rule? Perhaps this kind of transition rule was encoded in the network's recurrent weights, but could not be demonstrated during testing because the network was extremely inhibited against ever predicting A5-A8 and B5-B8. Willits (2013) showed exactly this. Analyses of the internal weights demonstrated that alternation and/or repetition rules were in fact encoded in the recurrent weights. And if the network was put through a second round of training on the test items, it learned rule-consistent test items much more quickly than rule-inconsistent items. The network had learned that there were items belonging to an A category, and items belonging to a B category, and what the order relationships of A and B items were to each other. It just was inhibited from applying this to new items. So in a new transfer learning scenario where the network was needing to learn how to respond to a new sequence, if the new sequence was rule consistent, all the network needed to do was learn appropriate input and output weights to categorize the novel items as either an A or a B, and then the sequences would be predicted correctly. In contrast, to learn rule-inconsistent items the network needed to learn input weights, output weights, and new recurrent weights reflecting the new sequential rule.

1.2. Transfer Learning

Willits's demonstration is actually an example of a popular neural network training technique, transfer learning (Pan & Yang, 2009). In recent years, large, industry-oriented deep learning systems are rarely trained "from scratch". It is often the case that an existing network that already performs well at some task is used as a base model, which is then trained to do a second task. If the source and target task share underlying structure, this speeds up the model's learning of the target task. Willits (2013) shows that this principle applies to abstract relational rules (like ABA sequence rules) applying to entirely novel items participating in those sequences, and not just to cases where the same items exist in the original training and in the transfer training.

1.3. Polyomino World

Our goal was to create a very simple artificial dataset that would allow us to investigate the basic capabilities of neural networks, and their ability to learn to classify labels and features of simple visual objects. Rather than using a dataset of real images, we wanted images that could be perfectly controlled with regard to their properties, and very rudimentary in order to simplify analysis and understanding of the neural network's behavior. Towards this end, we have created the artificial world "Polyomino World". Polyomino World is a world consisting of scenes of objects. We created a program for generating a Polyomino World (the set of scenes), allowing us to define objects in terms of their shape which pixels a shape occupies), the color of those pixels, and the position of each object in the scene. The goal of the Polyomino World program is to allow easy customized creation of datasets for testing neural networks.

In this paper, we used the Polyomino World program to create a very simple set of scenes consisting of an 8x8 grid of grey pixels (the background), with each scene containing a single object. In the following experiments, we defined all of our objects to be simple Polyominoes (plane geometric figures formed by joining one or more equal squares edge to edge), with examples shown in Figure 2.

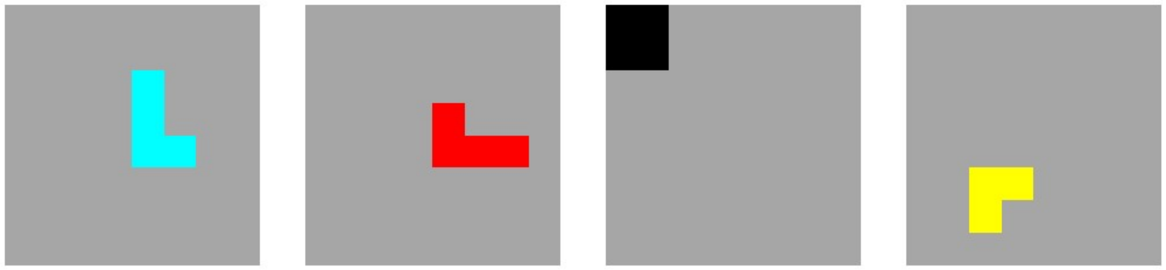


Figure 2. Example images from Polyomino World, with one colored shape on an 8x8 grid with a grey background. This image is transformed into a 192-element vector (8x8x3) containing 0's where each cell is empty/grey, and the color-appropriate RGB-values of +1 and -1 where a colored shape was present

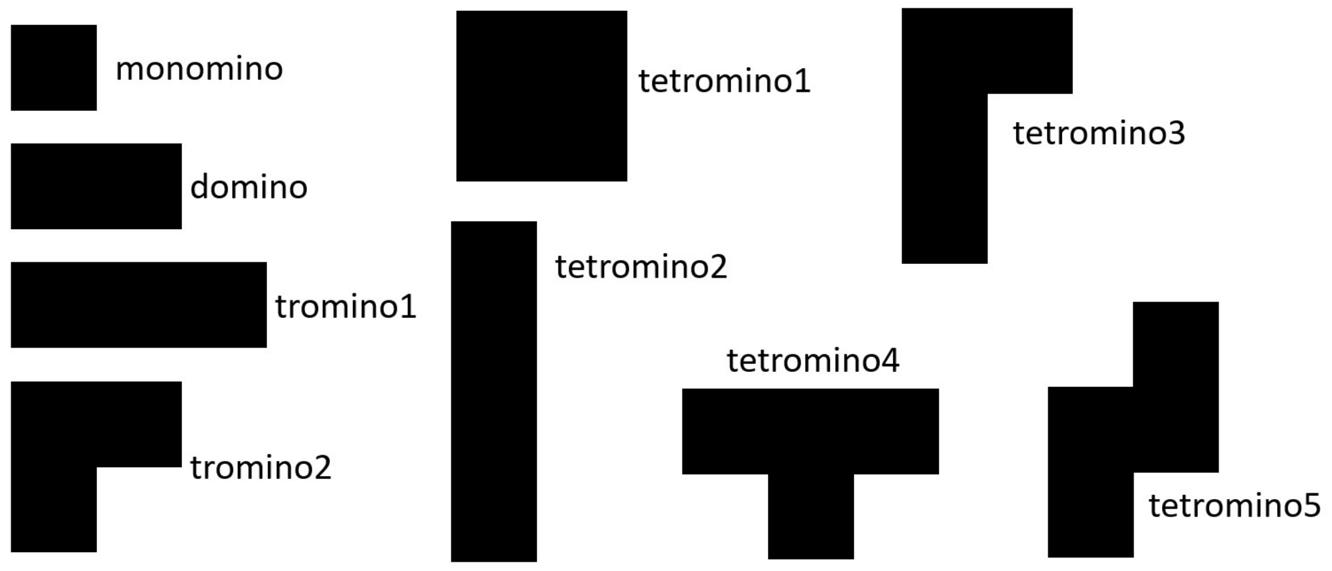


Figure 3. The full set of nine polyomino shapes.

The most well-known polyomino is the domino, consisting of two squares. In the following experiments, we used Polyominoes of size one (monominoes), two (dominoes), three (trominoes) and four (tetrominoes, the shapes used in the game *Tetris*). The full set of polyominoes we used is shown in Figure 3. As shown in Figure 3, there is only one kind of monomino (a square) and one kind of domino (a rectangle made of two connected squares). There are two distinct kinds of trominoes, and five different kinds of tetrominoes. Considering all shapes of size 1-4, there are 9 distinct shape types (monomino, domino, tromino1, tromino2, tetromino1, tetromino2, tetromino3, tetromino4, and tetromino5). Of course, some of the shapes can also be placed in different orientations. Dominoes and tetromino2's can be either horizontal or vertical. The tromino2, tetromino3, and tetromino4 can be placed in four different orientations, and the tetromino5 can be placed in eight different orientations. Thus, these polyominoes allow us to test the neural network's capabilities at learning translation invariant representations of the objects.

The Polyomino World program allows us to specify the color of each pixel of each object. In this paper, we limited it so that all of the pixels of each object had to be the same color, and would come from the set of eight colors that make up the eight corners of the RGB cube: red (RBG +,-,-), blue (-,+,-), green (-,+,-), cyan (-,+,+), magenta (+,-,+), yellow (+,+,-), white (+,+,+), or black (-,-,-).

The specific rules governing which shape and color combinations were present varied in each of the experiments, and will be described below.

1.4. Network Architecture and Training

The network architecture used in this paper was the same across all Experiments, and is shown in Figure 4. The network was a simple, fully-interconnected feedforward network with a single hidden layer. Why use such a simple network architecture, when state of the art object recognition networks are much more complex? The main goal of this paper is not to achieve state of the art performance on this relatively simple dataset. The purpose is to break open the neural network black box and obtain a better understanding of how they work. As such, we wanted to start with the simplest network that is capable of learning to correctly categorize the objects, and then seek to understand its inner workings. From this vantage point, future work can investigate other architectures, like deep networks with more hidden layers, convolutional and pooling layers, and recurrent networks, to investigate how those kinds of architectures change the nature of the representations in the model.

The networks we used is shown in Figure 4. The network had a 192-unit input layer (one unit for each 8x8 red pixel, blue pixel, and green pixel), and a 22-unit output layer (with a sigmoid activation function applied to each unit, scaling its output to range from zero to one). The output layer had one unit for each of the eight colors, one unit for each of the four sizes, and one unit for each of the nine distinct shape type labels. The network was trained to take a particular input image, and to simultaneously correctly classify the shape, size, and color of the object. The network's hidden layer size was manipulated in Experiment 1, and had 32 units in each subsequent experiment.

The network's weights (from the input layer to the hidden layer, and from the hidden layer to the output layer) were fully connected (with each unit in each layer connected to each unit in the subsequent layer).

Each network was trained in the following manner. First, it was presented with a set of input vectors representing each image in the training set. For each input vector, activation was propagated along the weighted connections to the hidden layer and then to the output layer. These output activations were compared to the correct output activation, which constituted three 1's (one each for the correct color, shape, and size labels), and a zero for all other units. The mean squared error of the difference between the predicted and actual output was used as an error function that was used to train the weights using stochastic gradient descent.

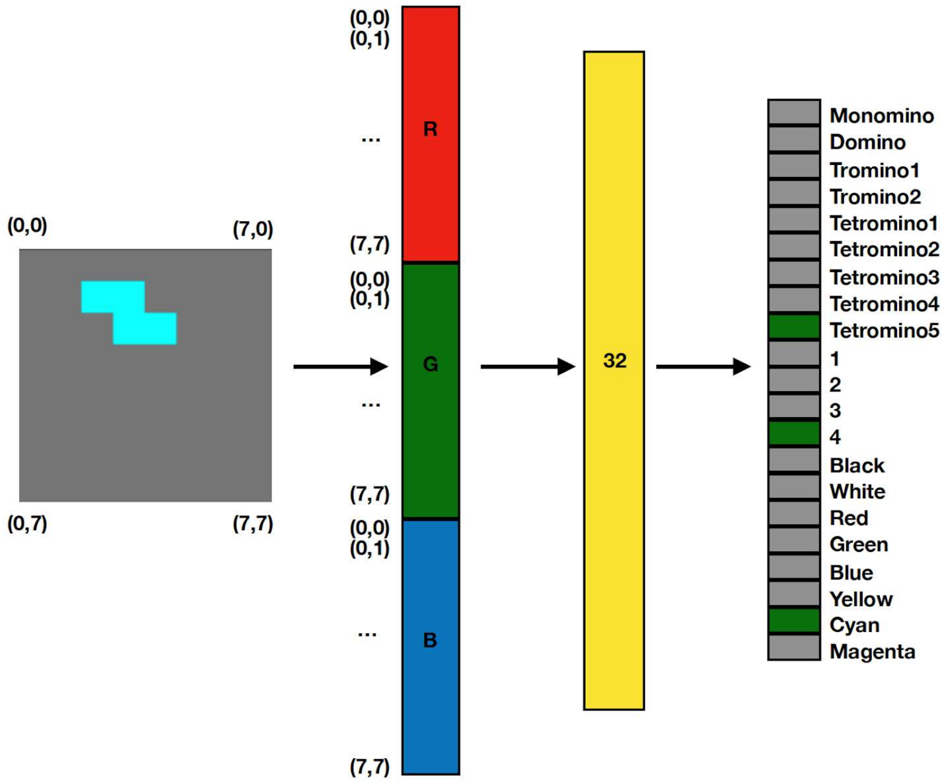


Figure 4. Network architecture for all experiments. The input layer was 192 units (8x8 units for each red, blue, and green color channel in the input image). The size of the hidden layer was manipulated in Experiment 1, and had 32 units in Experiments 2-5. The output layer had 22 units, specifying the 9 different shapes, 4 different sizes, and 8 different colors.

2. Experiment 1

Experiment 1 was designed to test the network's basic capabilities learning to classify an object's shape, size, and color. In this experiment, we manipulated four model parameters to find the values that maximized the network's performance: 1) hidden layer size, 2) batch size, and 3) learning rate. We had two primary questions for this experiment. The first was to discover which set of parameters were capable of learning to correctly classify the objects' shape, size, and color, and to learn to do so in the smallest number of training trials. This question was of practical interest (because we would use the best performing parameters in subsequent experiments) but also of theoretical interest, for the reasons described below. The second major question was, to the extent the network had difficulty classifying shapes, sizes, or colors, what were the particular items with which it had difficulty. This would be useful to know for the follow-up transfer experiments.

The first parameter we manipulated was the network's **hidden layer size**. The manipulation of hidden layer size was of theoretical interest, as one major question in this investigation is the extent of the neural network's capabilities of representing information in terms of the abstract features that defined the objects. The smallest hidden size we tested was of size 8, chosen because it was approximately the minimum number of hidden units that are theoretically necessary to solve the problem. To correctly classify each image, the neural network would need 3 hidden units for specifying the presence of color - one each for red, blue, and green, which in combination can produce any of the eight colors. In addition, the network would need at least four hidden units to classify all 9 shapes, as 4 hidden units be used to represent $2^4=16$ distinct categories. The network would need no additional hidden units to represent size, as the hidden units used for shape could also be used to represent size (with different weights from those hidden units to the output units for shape and size). Notably, a hidden size of 8 would require the shapes to be represented in a binary format, with at least 3 of the hidden units being necessary for determining which shape was present. This may not actually be the ideal internal representation for transfer. Alternatively, using 9 hidden units and representing each shape independently in a symbol-like fashion would allow better transfer, as the representations for the different shapes would not be entangled. Thus, we also tested a model with 16 hidden units. Finally, to see if the model learned faster with more hidden units, we also tested models with 32 and 64 hidden units. These hidden sizes would still result in a compressed representation of the mapping from the 192 inputs to the 22 outputs, but would result in distributed representations with more redundancy, with uncertain effects on the network's ability to transfer its knowledge to new situations.

The second parameter we manipulated was the **batch size**, or the number of scenes the network saw before weights were updated. We tested a batch size of one, meaning that after each scene, a classification prediction as made, error was computed, and weights were adjusted so as to minimize the error for that trial. We compared this to a batch size of 8192, meaning that after each scene, a classification prediction was made, error was computed, and this error was used to figure out how the weights should change to minimize error on that item, and suggested weight changes were added together for 8192 different items. After all 8192 items, the cumulative weight change was made to the network. Using a large batch size has advantages and disadvantages. Batch updating (updating weights only after many items) means the weights need to be changed less often, resulting in fewer computing resources being used. This is useful in the case where weight changes from different items would just cancel out, being changed one way for one item, only to be changed back for another item. In such a case the weight changes are unnecessary. Batch updating can have a downside, however. It may sometimes be the case that the weight changes from two items would cancel out during batch updating, but that some third item would benefit from the network adjusting for the first item before being adjusted back because of the second item. In practice, incremental learning (also often called online learning, or updating weights after each item) often outperforms batch learning in total error rate, but at the cost of taking much longer to do the learning.

The third parameter we manipulated was the **learning rate**. The learning rate is a constant value, usually between 0 and 1, and is a multiplier that is applied to each weight update. When a neural network model learns through error driven learning, the model's error in its predictions are used to calculate how the weights should change to have made a more correct predict for that item (or set of items, in the case of batch learning). The learning rate is multiplied by that suggested weight change, usually resulting in the weights only being adjusted a small amount in the desired direction.

2.1. Method

2.1.1. Stimuli. For training, we created a training set of images containing only one object. An image was created for all possible orientations of each shape, in all possible positions in the 8x8 grid. The total number of scenes for each shape is 8 (the number of colors), times 9 minus the shape's height, times 9 minus the shape's width, times the number of rotational and flipping variants of that shape. This resulted in a total of 11496 scenes in the dataset, with the breakdown by shape type shown in Table 1

Shape Name	Colors	Variants	Height	Width	Scenes
monomino	8	1	1	1	512
domino	8	2	1	2	896
tromino1	8	2	1	3	768
tromino2	8	4	2	2	1568
tetromino1	8	1	1	4	392
tetromino2	8	2	2	2	640
tetromino3	8	8	2	3	2688
tetromino4	8	4	2	3	1344
tetromino5	8	8	2	3	2688
Total					11496

2.1.2. Design. In this experiment we ran 5 models starting from random weight initializations for each combination of parameters, resulting in a 4 (the number of hidden sizes) x 2 (batch size 1 vs. batch size 8192) x 4 (the number of learning rates), for a total of 160 models.

2.1.3. Procedure. All models were initialized with random weights chosen from a uniform distribution ranging from X to Y. Next, each network was trained on the 11496 item training set for 2,000,000 trials, thus seeing the entire dataset approximate 174 times. The weights were adjusted either after each object (for batch size 1) or after the entire dataset (for batch size 8192) using stochastic gradient descent.

2.3. Results

For all 120 models, we evaluated accuracy of classifying shape, size, and color in a simple balanced accuracy framework. Balanced accuracy is a popular metric for dealing with the fact when you are evaluating accuracy for each type, and you want to consider both the hits (true positives) and correct rejections (no false positives), there are many more opportunities to reject a false positive than to affirm a true positive. Balanced accuracy calculates the hit rate and correct rejection rate separately, and averages them together. All numbers described and plotted below are balanced accuracy rates.

For learning to correctly classify color and size, models with a hidden size of 16, 32, and 64 all quickly achieved 100% balanced accuracy, within 10,000 trials, regardless of the other parameters. In contrast, the models with only 8 hidden unit models never reliably achieved 100% for color or size classification accuracy. Some random runs of each model with different learning rate and batch size values achieved 100% for at least some period of time, with averages for each parameter setting shown in Figure 5.

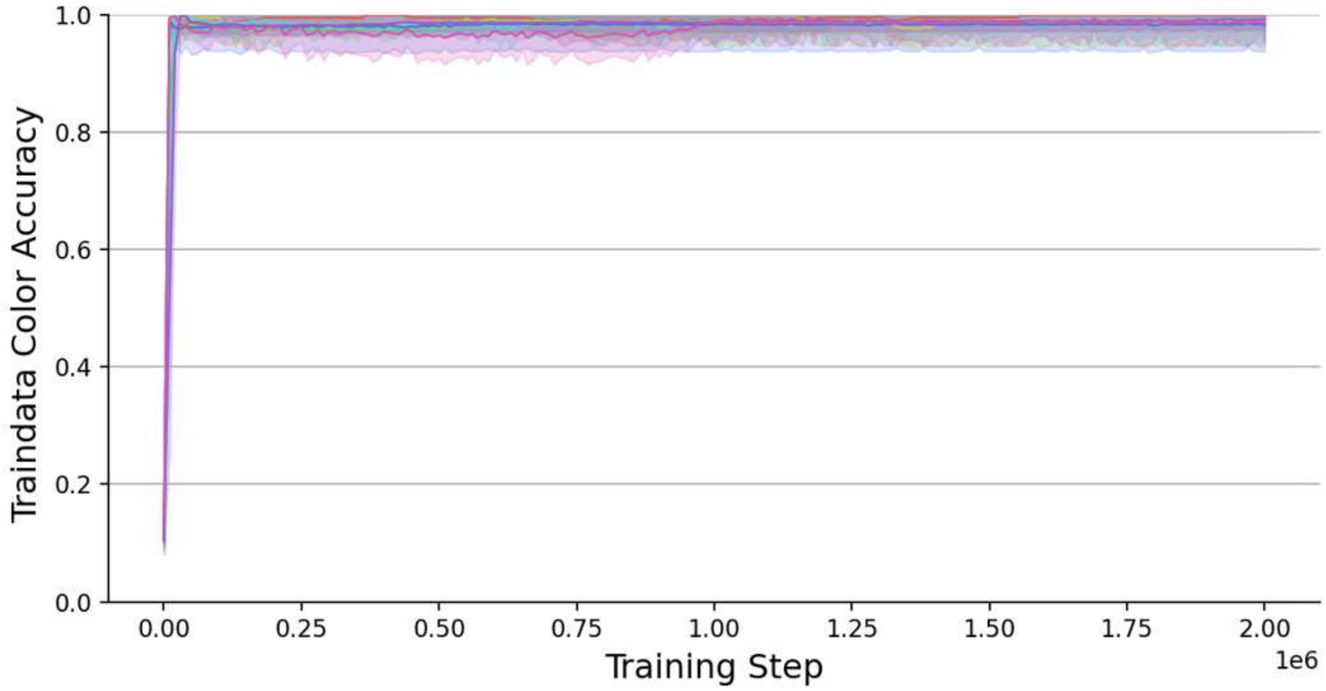
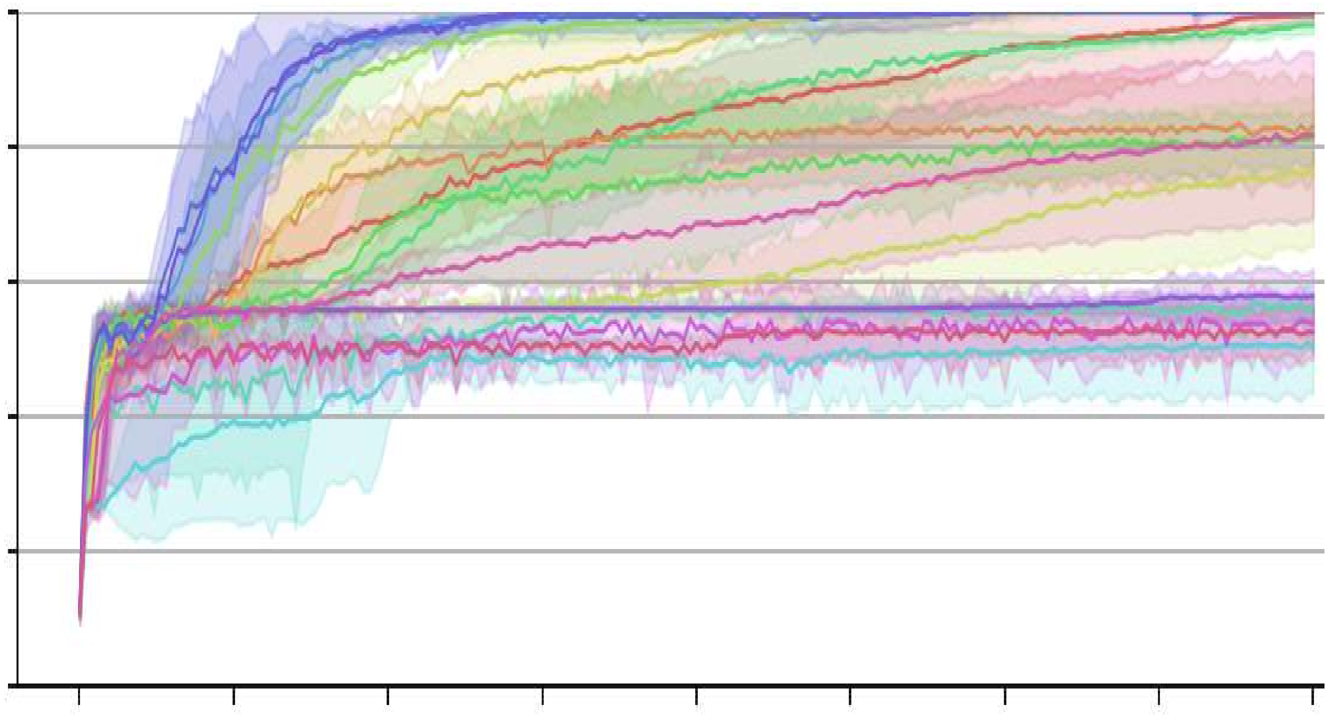


Figure 5. Performance of models with 8 hidden units on color and size classification, as a function of learning rate and batch size parameters.

For shape, balanced accuracy performance was much more variable for all sets of parameters. Models with a hidden size of 8 were still the worst performers, almost always topping out at a performance between 50%-55% balanced accuracy, with the only other parametric variation being the speed with which they got to that point as a function of learning rate. At the other end of the hidden size continuum, size 64 models always achieved (or were clearly on the path to achieve) 100% balanced accuracy after 2,000,000 training trials. The 64 hidden unit models achieved this more quickly as a function of the other two parameters, with the larger batch size and faster learning combining multiplicatively to result in faster achievement of 100% balanced accuracy. The batch size 8192, learning rate 0.8 model achieved this in approximately 350,000 trials, with the lower learning rate models on pace to achieve it after about 2,225,000 trials. The hidden size 32 models with faster learning rates also consistently reached 100% balanced accuracy after about 1,000,000 training trials, with the slower learning rate models on pace to achieve 100% balanced accuracy after about 3,000,000 training trials. The hidden size 16 models showed the largest interaction between the other two parameters. The hidden size 16, batch size 1 models struggled to achieve 100% accuracy, with the fastest learning rate models plateauing at about 82% accuracy. In contrast, the hidden size 16, batch size 8192 models with the fastest learning rate had achieved 95% accuracy by 2,000,000 trials, with the medium learning rate models converging on that performance but at a much slower rate. Results on shape are shown in Figure 6.



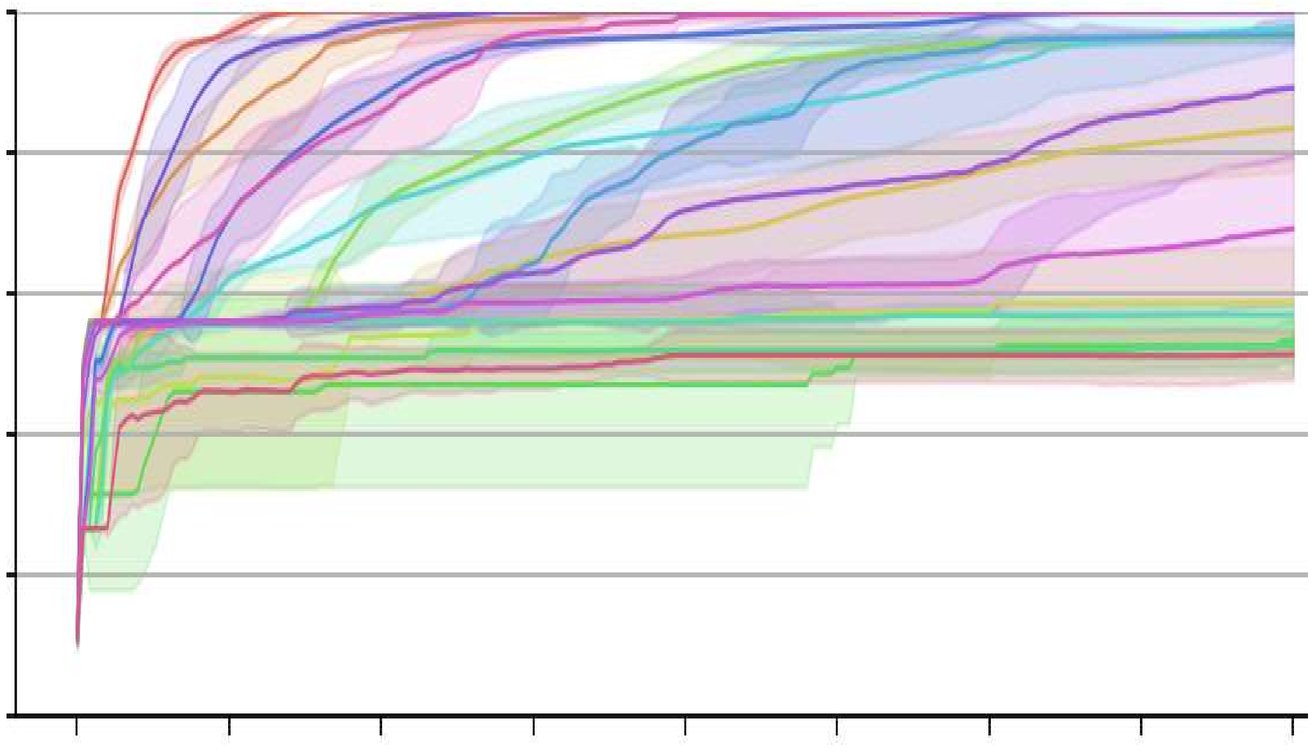


Figure 6. Top: models trained with a batch size of 1, with different lines plotted for the combinations of learning rate and hidden layer size parameters. Bottom: models trained with a batch size of 8192, with different lines plotted for the combinations of learning rate and hidden layer size parameters.

2.4. Discussion

The results from Experiment 1 demonstrate a number of important baseline facts about how neural network classifiers perform. The most notable fact is just how much data and experience the models need to achieve perfect accuracy on the task. Color and size, relatively easy tasks, take thousands of trials to achieve perfect accuracy, and shape, while admittedly a harder task, takes millions of trials. The implications of this fact for neural network models as a model of human learning are, however, not straightforward, for two reasons. First is that the number of trials a neural network model sees an item, and the number of times a human sees an image, may not be directly comparable. One might hypothesize a slightly more complex neural network model that is able to store memories of associations between images and their classifications, and then replay those memories (re-presentations, if you will) in order to give the network more training.

The second reason the number of trials required for learning may not be as bad as it seems, and the one more germane to this paper, is the consideration of prior knowledge and transfer learning as a part of the learning process. These models are effectively blank slates with no pre-existing knowledge of any kind (other than the pixelated and RGB structure of the input). How many trials does it take a human to learn their first shape labels? This is an answer we don't know the answer to. While the answer is surely not hundreds of thousands for each shape (as was required by our model), the number could still be very high. Whatever the answer, a direct comparison is difficult to make without better developmental data on the visual experiences of children. But the future experiments raise the bar of the model. In those experiments, the networks are tested in various ways to see how quickly they learn new objects, either of a novel shape, color, position, or orientation. Those cases are more comparable to a human who is learning a new object but on a basis of prior knowledge, and so their speed and ability to do so is a more fair and straightforward comparison of human learning.

3. Experiment 2

Experiment 1 established the baseline performance of the neural networks when shown all of the objects, in every possible combination of position, orientation, and color. This is obviously an unrealistic expectation for learning. In this experiment, we test how a model that has learned to correctly classify objects in some positions transfers that knowledge to new, unseen locations. This experiment was set up to compare the performance of two conditions of models. Condition 1 was the "Interleaved Transfer Learning", where models were put through two phases of training: a "pre-training phase" of 2,000,000 trials of the same dataset as in Experiment 1, except only presented to half of the visual field (top half only or bottom half only; and a "transfer phase" of 2,000,000 trials of the exact dataset from Experiment 1. For the entirety of the 4,000,000 trials, the model was tested on its performance in the half of the visual field that was omitted during the pre-training phase. Condition 2 was the "No Pre-training" condition, which was effectively just a replication of Experiment 1. This condition was a model trained on the full Experiment 1 dataset, in the entire visual array, for 2,000,000 trials. Performance in this condition will be compared to the performance in the "transfer phase" of Condition 1, to see if learning in the transfer phase reached higher accuracy or more quickly achieved perfect accuracy.

We hypothesized that Experiment would act like a visual replication of the previously described AAB/ABB model made by Marcus (2000). Whatever the model may learn about the items in the top (or bottom) half of its visual array, there will be no way for it to demonstrate that knowledge on the test items during the pre-training phase. Because the test items always occur in a portion of the array where no items ever occur during training, the weights from that portion of the input layer to the hidden layer will remain unchanged or be set to zero. Thus, when the test items are used as input, the input arrays will either be multiplied by random numbers or zeros, resulting in chance performance. But like in Willits (2012), in the transfer learning phase of the Experiment, the model will learn to correctly classify the items in the previously untrained part of the visual array much more quickly than a model without the pre-training. This is because the pre-trained model will have already have learned a set of weights that usefully map hidden layer representations to correct classifications. All the model will need to do is learn to set the weights from the previously untrained part of the input to the hidden layer in a way that leads to similar hidden layer representations being created for items in the untrained half of the array as were created from the trained half of the array.

3.1. Method

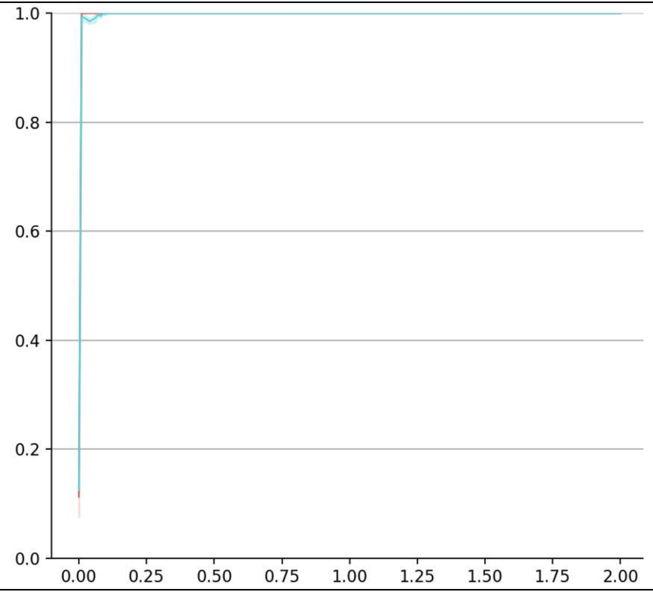
3.1.1. Stimuli and Design. In the "Interleaved Transfer Learning", during the pre-training phase five network were trained on the Experiment 1 dataset but restricted to the top half of the visual array, and five networks were trained on the dataset restricted to the bottom half of the visual array. In the transfer learning phase, the network was trained on the dataset in the entire visual array. These networks were tested on their performance in the half omitted during the pre-training phase. In the “No Pre-training” condition, 10 networks were trained on the dataset in the entire visual field, and five were tested on their performance in the top half only, and 5 were tested on their performance in the bottom half only, creating a set of control networks to compare to the Interleaved Transfer Learning condition. All trained models had 32 hidden units, used a batch size of 1, and used a learning rate of 0.5. These values were chosen because this set of parameters was among those that had perfect performance in Experiment 1, while still being relatively slow at learning to do so. Previous research has shown that models that use slower learning rates, incremental (batch size 1) learning, and fewer hidden layers tend to show better generalization (CITATION).

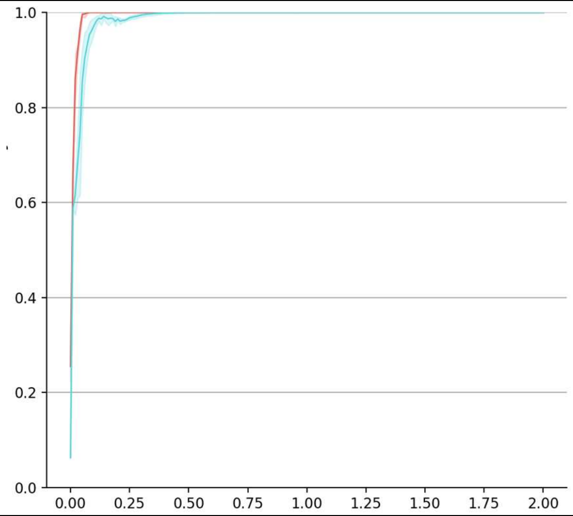
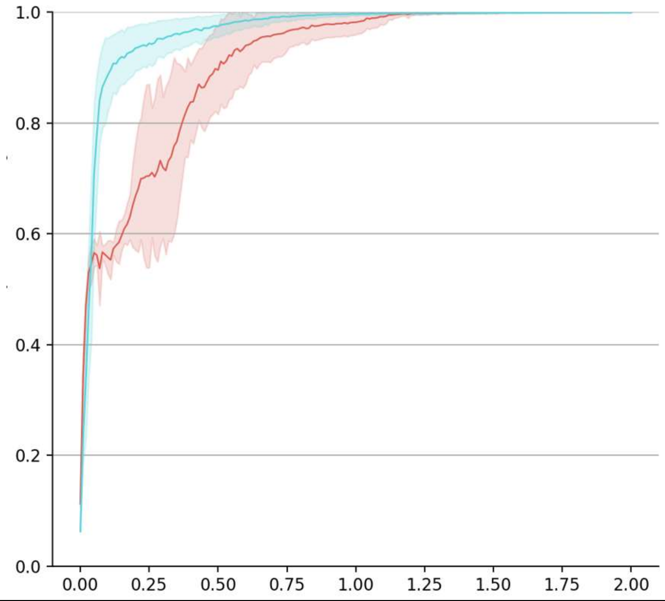
3.1.2. Procedure. All models were initially assigned random weights from a uniform distribution ranging from X to Y. Next, models in the Interleaved Transfer Learning were trained in their pre-training phase for 2,000,000 trials, and then models in both conditions were trained on the full dataset for 2,000,000 trials. Models were evaluated using balanced accuracy in the same manner as Experiment 1, but only on items from the half of the visual field corresponding to the omitted half of the visual field from the pre-training phase.

3.3. Results

In the Interleaved Transfer Learning condition’s pre-training phase, the 10 trained models (half on top only, half on bottom only) performed as predicted. While learning to correctly classify items in the trained half of the visual array (replicating Experiment 1), the models utterly failed to show any kind of transfer learning, as would be expected by a model being shown items so dramatically outside of the training set. These results are shown for color, size, and shape, in Figure 7.

The results for the “pre-training” phase are exactly as predicted, with balanced accuracy at chance (or below chance) for judgments about an object’s color, size, and shape, when models are tested on items in the part of the visual field in which they were not trained.



3.4. Discussion

4. Experiment 3

For Experiment 3, we were interested in the ability of the model to recognize translational variants of objects on which it had been trained. We designed experiment 3 to be much like experiment 2, with an “Interleaved Transfer Learning” condition and “No Pre-Training” condition. As in Experiment 2, in the Interleaved Transfer Learning condition, there was a pre-training phase and a transfer learning phase, and the performance in the transfer learning phase was compared to the performance in the No Pre-training condition.

The training and test sets in Experiment 3 were designed such that, for all shapes that had at least two variants (i.e. every shape except for monomino and tetromino5), the different rotations and flips of each shape were divided into two groups, with half put in the training set and half in the test set. For example, dominoes have two variants (vertical and horizontal), and so one was placed in the training set and the other in the test set. Tetromino3 shapes (the one that looks like a capital L) have 8 variants: all 90 degree rotations, plus the flipped reversal of each of those four. These were divided such that 4 were in each training set and 4 were in each test set.

Because of the shape results in Experiment 2, we again hypothesized that there would be failure to show transfer to the test set during the Pre-training phase, but that the learning in the Transfer learning phase would be faster in the Interleaved Transfer Learning condition compared to the No Pre-Training condition.

3.1. Method

3.1.1. Stimuli and Design. In the “Interleaved Transfer Learning”, during the pre-training phase five randomly initialized networks were trained on half of the variants of domino, tromino1, tromino2, tetromino2, tetromino3, tetromino4, and tetromino5 shapes, in all possible colors and in all possible conditions. The other half of the variants of each type were in the test set for those five networks. Five other networks were trained with the training and test sets reversed. All trained models had 32 hidden units, used a batch size of 1, and used a learning rate of 0.5.

3.1.2. Procedure. All models were initially assigned random weights from a uniform distribution ranging from X to Y. Next, models in the Interleaved Transfer Learning were trained in their pre-training phase for 2,000,000 trials, and then models in both conditions were trained on the full dataset for 2,000,000 trials. Models were evaluated using balanced accuracy in the same manner as Experiment 1, but only on items from the half of the visual field corresponding to the omitted half of the visual field from the pre-training phase.

3.3. Results

In the Interleaved Transfer Learning condition’s pre-training phase, the 10 trained models (half on top only, half on bottom only) performed as predicted. While learning to correctly classify items in the trained half of the visual array (replicating Experiment 1), the models utterly failed to show any kind of transfer learning, as would be expected by a model being shown items so dramatically outside of the training set. These results are shown for color, size, and shape, in Figure 7.

The results for the “pre-training” phase are exactly as predicted, with balanced accuracy at chance (or below chance) for judgments about an object’s color, size, and shape, when models are tested on items in the part of the visual field in which they were not trained.

5. General Discussion

6. References

Marcus, G. F. (1998). Rethinking eliminative connectionism. Cognitive psychology, 37(3), 243-282.