The Relative Neuroplasticity of Artificial and Biological Neural Network Configurations

Oliver Balfour

Alfred Deakin High School

June 20, 2018

Abstract

In this experiment the neuroplasticity of neural networks, both artificial and biological, was tested. Several configurations of computing machines, ranging from simple dense feed-forward Multi-Layer Perceptrons (MLPs) and Artificial Neural Networks (ANNs) to more complex Recurrent Neural Networks (RNNs), were tested, with the human brain as a baseline for performance. Also tested was a Support Vector Machine (SVM), a much older algorithm with a mathematical and not biologically-inspired approach to classification, to gauge how far the field of Machine Learning has come in the past decades both in terms of performance and neuroplasticity. Each machine was tested (with minor programmatic alterations to the input and output layers of each artificial model to account for differing input sizes) against several different classification and regression problems including but not limited to classifying handwritten digits (using the MNIST dataset) and performing sentiment analysis on highly polarized movie reviews (using the SAR14 IMDb review dataset). The neuroplasticity of each model was then evaluated based on the standard deviation of the error rates of each model in each of the problems tested.

Contents

Introduction

The human brain is capable of rewiring itself throughout its lifetime (Draganski et al. 2004). This feature, known as neuroplasticity, is the driving force behind the process of learning, and is present in a wide range of animals. It was first discovered in rats in the 1960s (Bennett et al. 1964), and the theory behind it has applications as far as artificial intelligence and machine learning. In this experiment, the extent to which various configurations of Artificial Neural Networks (ANNs) exhibit neuroplasticity is tested, by teaching the same ANNs to perform very diverse tasks, compared to a baseline of the performance of the human brain.

The aim of this experiment is to determine how far behind in terms of neuroplasticity modern Artificial Neural Networks (ANNs) are when compared with the human brain. This is done by training the same artificial and biological neural network configurations to perform several different classification and regression tasks, and measuring the standard deviation of the error rates across each problem as a measure of neuroplasticity.

It is hypothesised that:

1. The human brain will marginally surpass state-of-the-art ANN configurations in all problem domains, and will perform exceptionally well in all problem domains. (Refer to the method on page 4 for a full list of problems tested.)
2. The Recurrent Neural Network (RNN) will perform well on all problems, but particularly the sentiment analysis problem.
3. The standard densely-layered feed-forward Artificial Neural Network (ANN) will perform acceptably on all problems.
4. The Support Vector Machine (SVM) will perform well recognising handwritten digits and classifying iris genuses, but will perform sub-par in all other problem domains.

Literature Review

Whilst there is plenty of literature on machine learning and neuroplasticity as separate topics, there is very little directly relating to both. The only publicly available scientific paper pertaining to both merely documents a method for making an ANN neuroplastic (Perwej and Parwej 2012), however its approach is essentially a synaptic pruning method for reducing overfitting in ANNs identical to the well-established method of neural network dropout, which randomly discards nodes from a neural network to reduce overfitting (Srivastava et al. 2014). The only other paper referring to both topics is behind an expensive journal paywall.

This experiment focuses on a neglected area within the field of machine learning: as artificial intelligence research is primarily focused on highly specialised cutting-edge techniques, investigation into the re-usability and adaptivity of neural networks has been left behind. Considering that a true artificial intelligence requires an extremely plastic brain to be capable of learning without requiring a considerable number of nested neural networks for each individual task it is capable of performing, this investigation is important as a measure of the level of progress that the field of machine learning has achieved toward an Artificial General Intelligence (AGI), not in terms of specialised narrow intelligence feats such as those of AIs designed solely to play games like Chess or Go at a highly competitive level, but in terms of how specialised narrow AI achievements can be generalised, or bridged across to one another to create a neuroplastic general intelligence.

The techniques used to create neural networks in this experiment, however, are very well documented. The perceptron, an outdated artificial model of a neuron, was documented thoroughly by its creator (Rosenblatt 1958), the multi-layer perceptron model, an improved model that featured hidden layers and the backpropagation algorithm was documented in (Rumelhart, Hinton, and Williams 1986); likewise, the original Support Vector Machine algorithm was well documented (Cortes and Vapnik 1995). All of the neural network configurations used in this experiment rely on well-established methods that have been hot topics in the world of AI research over the past few decades. Simple feed-forward backpropagated sigmoid ANNs have been in

wide use since the late 1980s, and CNNs and RNNs were both designed as more specialised improvements on the simpler ANN and MLP structures.

Background Information

A neural network is essentially a simplified computer abstraction of a brain, with primitive and computationally inexpensive representations of the core functions of neurons in a real life brain. Neural networks are organised into layers, usually at least three, which contain distinct neurons. The first layer takes binary or floating-point input from the host computer program, and the last returns binary output back to the host program. In feed-forward neural networks, the most common type of neural network, simulated signals pass forward through neurons layer by layer without any form of recursion.

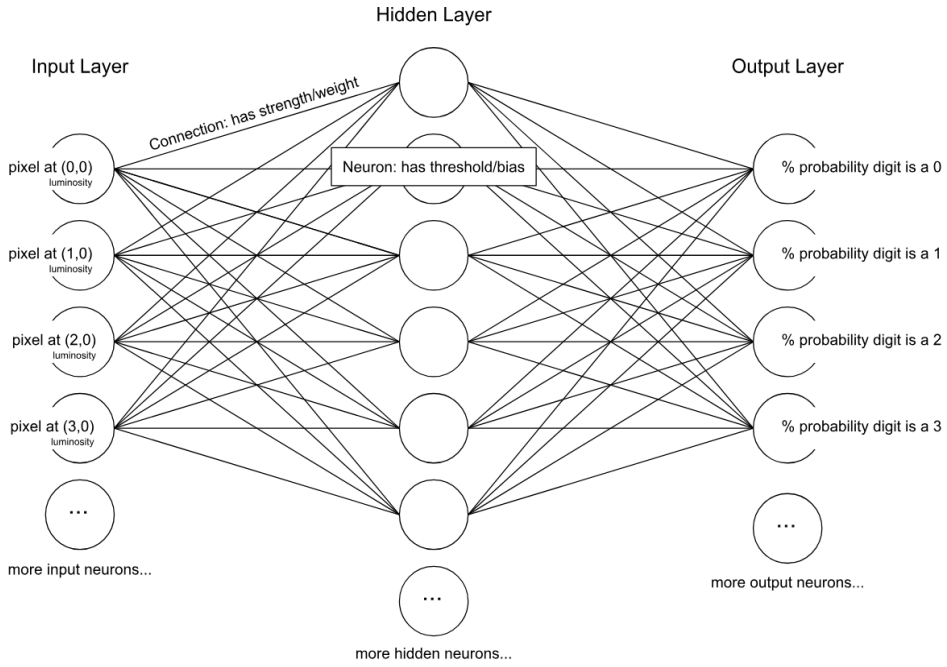

Figure 1: Diagram of a feed-forward Artificial Neural Network trained to recognise digits.

A neuron is typically represented by a series of numbers: an activation threshold (bias), and a series of numbers representing the strengths of connections with neurons in the previous layer. Given the outputs (activations) of the neurons in the previous layer (or given the input data from the host program in the case of the input layer), a neuron's output (activation) will equal the sum of all of previous layer's neurons activations multiplied by the connection strengths (weights), with the activation threshold (bias) of the neuron in question being subtracted. In a more traditional perceptron model, the output would be transformed such that if it is positive it is a binary 1, and if it is negative it is a binary 0; however more modern sigmoid models usually transform this value into a floating-point decimal between 0 and 1 using a learning function.

In this experiment, both perceptron and sigmoid neurons are tested, as well as a Support Vector Machine (SVM), which is an older algorithm designed for classification problems.

$$n_a = \begin{cases} 0 & \text{if } \sum_i M_{i_a} M_{i_w} \leq n_b \\ 1 & \text{if } \sum_i M_{i_a} M_{i_w} > n_b \end{cases} \quad (1)$$

Equation 1: Activation (output value) of a perceptron.

$$n_a = \sum_{i=0}^{|M|} (M_{i_a} M_{i_w}) - n_b \quad (2)$$

Equation 2: Activation of an artificial sigmoid neuron, before applying a learning function.

In the above equations, M is the set of neurons from the previous layer, M_{i_w} is the weight (or axon-dendrite connection strength) of the ith neuron, and M_{i_a} denotes the activation (output value, a floating-point number from 0 to 1 transformed by a learning function); n is the neuron whose output is being calculated and n_b its bias (activation/firing threshold.)so the above formula boils down to adding all of the connection weights multiplied by the outputs, and then subtracting the bias. In the sigmoid model, a negative value means the neuron does not fire, as the threshold has not been reached, and a positive the opposite. The output is wrapped in a function to make the output smoother and more efficient at learning, but that is an irrelevant implementation detail. In the perceptron model, a negative value is replaced with a 0 and a positive with a 1.

The weights and biases of each neuron and neuronal connection are optimised to serve a specific purpose through the process of training. Usually, these values will start as random numbers, and through a process known as Stochastic Gradient Descent the values are slowly optimised until the neural network model is sufficiently good at its task. (Refer to the glossary on page 9 for more information on Stochastic Gradient Descent.) There are many applications for neural networks: A neural network may be trained to classify images (computer vision and optical character recognition), to estimate data such as house or stock prices given data about their respective markets, or even to control robots. This adaptivity, or neuroplasticity, is investigated in this report.

Nodes in neural networks are quite similar to their biological counterparts, in that they have a constant activation threshold (in biological neurons this threshold is -55mV, whereas in artificial sigmoid neurons this is a unique number which is recursively optimised during the training process), take input from other neurons they have connections with (although sigmoid neurons have differing connection strengths and are connected to all nodes in the previous layer; biological neurons simply have binary connection strenghts — they either are or are not connected), and produce one output which is used as an input by additional nodes in the network. The primary difference between neural networks and brains overall is that brains, even in rodents and insects, have many orders of magnitude more neurons, and learning algorithms in biological networks are the fine-tuned result of billions of years of evolutionary processes.

Method

There are 6 neural networks tested in this experiment, against 4 different classification and regression problems, each in different problem domains. The neural network configurations are listed below:

1. A Support Vector Machine (SVM), to show the inefficiency of traditional mathematical or logical approaches compared to biologically inspired.
2. 8 different densely-layered feed-forward Multi-Layer Perceptrons (MLPs) using perceptrons instead of sigmoid neurons (2 sets of 4 different models with differing hidden layer configurations, one set with dropout and one without).
   The 4 hidden layer configurations are:

(a) 1 hidden layer with 128 neurons

(b) 1 hidden layer with 512 neurons

(c) 2 hidden layers with 256 neurons each

(d) 2 hidden layers with 512 neurons each

All dropout-enabled models have the dropout percentage set to 20% (so 20% of neurons are randomly discarded throughout the learning process.)

3. 8 standard densely-layered feed-forward sigmoid Artificial Neural Networks (ANNs), with hidden layer configurations identical to those of the MLPs above, aside from the nonbinary activation function.
4. 2 Recurrent Neural Networks (RNNs), with 1 Long Short-Term Memory (LSTM) hidden layer containing 128 neurons. One model has dropout, the other does not.
5. And as a baseline for performance and neuroplasticity, several people are used.

It was decided that Convolutional Neural Networks (CNNs) should not be included in the experiment, as although they are a very popular type of neural network, they only work with multidimensional spatial or temporal data (their primary use-cases are in computer vision, optical character recognition, and audio analysis.) This means that they cannot feasibly compute linear regression or non-temporal classification functions without having their kernel layers configured to apply no transformation to the data (by having nil values for all cells except the centre, which would have a 1, such that each kernel operation merely acts as a cat operation.)

Each of the above models will be tested against the following classification and regression problems:

1. Handwritten digit recognition (using the MNIST dataset)[1]
2. Iris flower classification (using Fisher's dataset[2] (A. Asuncion 2007))
3. Sentiment analysis (using the SAR14 dataset of IMDb movie reviews)
4. Breast cancer diagnosis (using the University of Wisconsin dataset[3])

The human brain will be tested against each of the aforementioned problems, although due to the varying nature and medium in which the datasets above are presented in, which means different cortices of the human brain will be used for different problem domains, the human test subjects will have a not insignificant advantage over their machine counterparts.

To avoid allowing the human test subjects to utilise areas of their brains which have already been trained in the problem domains this experiment is testing (for example, letting a native English speaker perform the sentiment analysis problem on polarised English movie reviews), all of the textual samples have been machine-translated (using Google Translate) into TODO, a lanuage with a different semantic structure, which does not share any major linguistic roots, and which is reliably machine-translated. Additionally, in the digit recognition problem, all images of handwritten digits are transformed/permuted using a randomly generated feature-extraction algorithm so as to be different enough from their source images as to be indiscernable by an untrained eye previously aquainted with Arabic digits.

This way, whilst different cortices within the neocortex will be used by test subjects, the human subjects will not have a significant unfair advantage over their machine counterparts beyond the scale of their brains and the level of structural and/or learning algorithm optimisation (through natural selection or through stochastic gradient descent algorithms) accessible to them.

The experiment happened in two phases for each combination of model (artificial or biological) and problem: the training phase, and the testing phase.

[1]The MNIST dataset is available at: http://yann.lecun.com/exdb/mnist/

[2]The Iris dataset is available at: http://archive.ics.uci.edu/ml/datasets/Iris

[3]The Breast Cancer diagnosis dataset is available at: http://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+(Diagnostic)

In the training phase, the artificial models are fed data along with the desired results from the respective datasets and left to their own devices. The people involved in the experiment are given the data similarly; however the data is printed and arranged in a more familiar tabular format.

In the testing phase, each model and participant is given data as in the previous stage, however the desired results are omitted, and the models and participants have to make predictions (or classifications, depending on the nature of the problem) based on inferences made from the training data. Note that participants are allowed to keep a copy of the training data for reference.

For each model-problem combination, two metrics were measured:

1. Performance, a measure of the accuracy of the model or participant in the testing phase.
2. Neuroplasticity, calculated as the standard deviation of the performance of the model or participant across all of the problems tested.

$$\sqrt{\frac{(\sum_{i=0}^{|P|} P_i - \bar{P})^2}{|P|}} \tag{3}$$

Equation 3: Calculation of neuroplasticity; a derivative of the forumla for population standard deviation.

In the above equation, P is the set of performance metrics for a model or participant, and m is the mean (average) of the set P.

Results

According to Simard, LeCun, and Denker 1993, the average human error against the MNIST digit recognition problem is about 1.5%.

TODO

Discussion

Summarize — averages, trends, inferences and generalisations

Explain/interpret — relating the results to relevant research

Experimental error and how to reduce it

Difficulties faced possible improvements for future experimentation

TODO

Conclusion

Respond directly to the aim

This should be a concise statement of WHAT you found out. It should state what your results showed in relation to the Aim and Hypothesis (supported or contradicted-NOT proved or disproved).The conclusion should not introduce any new ideas that have not already been mentioned elsewhere in your report.

TODO

Appendices

Appendix I: Implementation Details

The Python 3 source code files for all of the aforementioned artificial models are available online through the Git repository hosted at following URL: https://github.com/Tobsta/ScienceFair

All of the deep learning models tested were implemented in the Python 3 programming language, using the Keras machine learning library with a TensorFlow backend.

Glossary

Activation function: An activation/learning function is used to transform the output value of neurons to adapt the rate at which they learn.

Artificial General Intelligence (AGI): An AGI is an intelligent agent capable of learning to accomplish many general tasks at least as well as a human, as opposed to specialising in a particular task. At this stage, AGIs are mostly a topic of science fiction and radical speculation; however many experts believe that general intelligence is mere decades away. In a survey of prominent AI researchers, the median estimation for when there is a 50% chance of AGIs having been developed was 100 years from 2016 (Grace et al. 2017).

Artificial Neural Network (ANN): An interconnected network of artificial (ie. simulated) neurons that receive signals (in the form of binary or floating point numbers) as input, which pass through the connections between nodes in the network toward the output layer of neurons which emits a computed result. Simple ANNs typically contain an input layer, one or more hidden layers, and an output layer.

Classification problem: A classification problem is a type of machine learning problem in which a model is trained to classify data according to a few predefined classes. Examples include Optical Character Recognition and Computer Vision problems; the most common of which is handwritten digit recognition.

Convolutional Neural Network (CNN): A neural network in which some layers are not composed of artificial perceptrons.

Cost function: The cost function estimates the accuracy (error frequency) of a model; lower is better.

Dropout: Dropout is a technique to avoid overfitting in ANNs. It works by randomly removing nodes from the neural network. (Srivastava et al. 2014)

Feed-Forward ANN: Feed-forward ANNs are a class of ANN which have distinct layers of neurons whose signals are only fed forward. This significantly reduces the complexity of neural networks, and means that the neural network is not capable of retaining information.

Hidden Layer: Hidden layers are layers of perceptrons in an ANN that do not directly deal with input or output, but instead perform highly complex logic through their unique combinations of neural connections (weights) and activation thresholds (biases).

Input Layer: The input layer of an ANN is the layer which doesn't take input from the previous layer, but rather from a sample in a dataset specific to the problem domain the ANN is being trained for.

K-Means: K-Means is a clustering algorithm, which attempts to find groups/clusters in a dataset with a known amount of classifications by recursively estimating the centre points (centroids) of each cluster.

Learning function: See *Activation Function*

Machine Learning: Machine learning is a subfield of computer science that deals with artificial intelligence, pattern recognition and the creation of inference engines.

Neuroplasticity: The ability of a brain or machine learning model to adapt to fit different problem domains without modifying the overall structure (i.e. by changing only the synapse connections and activation thresholds for neurons, and not the underlying algorithms).

One-Hot Encoding: One-hot encoding is the process of converting a class ID to an array of numbers equal to the number of classes, populated with zeros, aside from the number at the index of the class ID, which is a one. To illustrate, the classes "Cat", "Dog", and "Horse" could be represented as [1,0,0], [0,1,0], and [0,0,1] with one-hot encoding. One-hot encoding is much easier for an ANN to work with, especially in an output layer.

Output Layer: The output layer of an ANN is the layer which calculates the final result. In classification problems, each neuron in the output layer will typically map to a sample class. The result of the output layer is then compared to the one-hot encoded class of the sample in the cost function to determine the error rate.

Overfitting: If a model is overfitted, it is so fine-tuned to its training dataset that it performs poorly during validation and testing.

Perceptron: A perceptron is a common, but outdated, artificial model of a neuron which has

a binary output as opposed to a floating-point one. This means that they are harder to train, as a small change in inputs creates a much larger change in output.

Recurrent Neural Network (RNN): TODO

Regression problem: A regression problem is a style of problem where a model is trained to find the relationship between input and output data. One example of a regression problem is estimating the selling price of a house given data on it (number of bedrooms, size of the block, etc.)

ReLU ReLUs (Rectified Linear Units) are a simple activation (learning) function that returns the input or 0, whichever is greater.

Sigmoid function: The sigmoid function is a learning function that creates an S-shaped curve.

Stochastic Gradient Descent (SGD): A method of minimising the error function of a neural network model (or other machine learning model) given multivariate input, in which the the input parameters are viewed as axes on an n-dimensional graph with the cost function as the Y-axis. SGD attempts to find the global minimum of the cost function without calculating it for every possible combination of parameters by rolling the ball (vector representing the weights and biases of the neural network) down the hill (this analogy only really works with 2 parameters as the X and Z axes and the cost function as the Y axis, but it can be expanded to work with n dimensions.) Local minimums are avoided by the learning rate hyper-parameter, indicative of the length of the step taken (ideally toward the global minimum) in each iteration of the SGD algorithm. The direction gradient toward the local minimum is estimated by calculating the partial derivative of the cost function with respect to each parameter, and determining which parameter, plus-minus the learning rate hyper-parameter, has the lowest result of the cost function.

Softmax: Softmax is an activation (learning) function that scales all outputs from a layer from 0 to 1, such that the sum of all outputs is 1. It is typically used for the output layer of a classification neural network, as a kind of floating-point one-hot encoding method.

Support Vector Machine (SVM): SVMs are an older method of machine learning which uses... (TODO)

Bibliography

A. Asuncion, D.J. Newman (2007). *UCI Machine Learning Repository*. URL: `http://archive.ics.uci.edu/ml/index.php`.

Bennett, Edward L et al. (1964). "Chemical and anatomical plasticity of brain". In: *Science* 146.3644, pp. 610–619.

Cortes, Corinna and Vladimir Vapnik (1995). "Support-vector networks". In: *Machine learning* 20.3, pp. 273–297.

Draganski, Bogdan et al. (2004). "Neuroplasticity: changes in grey matter induced by training". In: *Nature* 427.6972, p. 311.

Grace, Katja et al. (2017). "When will AI exceed human performance? Evidence from AI experts". In: *arXiv preprint arXiv:1705.08807*.

Perwej, Yusuf and Firoj Parwej (2012). "A Neuroplasticity (Brain Plasticity) Approach to Use in Artificial Neural Network". In: *International Journal of Scientific & Engineering Research*.

Rosenblatt, Frank (1958). "The perceptron: a probabilistic model for information storage and organization in the brain." In: *Psychological review* 65.6, p. 386.

Rumelhart, David E, Geoffrey E Hinton, and Ronald J Williams (1986). "Learning representations by back-propagating errors". In: *nature* 323.6088, p. 533.

Simard, Patrice, Yann LeCun, and John S Denker (1993). "Efficient pattern recognition using a new transformation distance". In: *Advances in neural information processing systems*, pp. 50–58.

Srivastava, Nitish et al. (2014). "Dropout: A Simple Way to Prevent Neural Networks from Overfitting". In: *Journal of Machine Learning* 15. URL: `https://www.cs.toronto.edu/~hinton/absps/JMLRdropout.pdf`.