State of the Field of Artificial Neural Networks

Andrew Mikos

Contents

1 Status Quo 1

2 Needs Addressed 1

3 Creation and Development 2

4 Refinement and Advancement 2

5 Current 2

6 Biological Foundations 2
- 6.1 Neuron Structure and function . . . 2
- 6.2 Artificial Neurons 3
- 6.3 Learning 3

7 technical components and composition 4
- 7.1 Neuron Layers 4
- 7.2 Feed Forward and Feed back 4
- 7.3 Neuron Processing 4

8 Perceptron 5
- 8.1 Multi-Layer Perceptrons 6

9 Adaline 6

10 Image Recognition 7
- 10.1 Structure 7
- 10.2 Recognition 7

11 Open Research Questions 7

12 Growing Applications 7
- 12.1 Banking and Financial Modeling . . 7
- 12.2 Facial Recognition 8
- 12.3 Biometrics 8

13 Learning Algorithms 9

14 Biological Possibilities 9

15 Conclusions 9

References 9

Appendix 9

1 Status Quo

The idea of a neural network dates all the way back to the beginning of computers. Even in the early 1940's, McCulloch and Pitts had already conceptualized a device to mimic the way animals process information [14]. The problem at the time was that computer technology itself was still very new. Most computers then couldn't do anything more then a modern calculator. Computers continued to develop however, and over time the standard for programming became functional based. A programmer would explicitly define the necessary steps to solve a problem and their order, the machine then takes explicitly those steps, and if the data is formatted incorrectly, or if the nature of the problem changes even slightly, the program is no longer valid. This changed somewhat with the development of object-oriented programming languages. Different from functional languages which execute code line by line, occasionally jumping to other specified locations in the program, object oriented languages create functions and object classes which then can be called upon to perform specific functions. This means that the actual order of execution of the code can become more dynamic, with functions called an non-definite number of times before the conclusion of the program. These changes however only mean that the code that is already written can be executed more flexibly, allowing the program to perform more adaptively. The program itself is still completely limited to the code as it is written by the programmers.

2 Needs Addressed

Neural networks arose out of the desire to create programs that are adaptive and flexible. This means that they can be applied to circumstances or problems that they were not originally designed for. This has many potential applications, from modeling complex systems such as weather and financial returns [4], and also handling facial recognition [11], a problem where the precise variables required to identify a face are not fully understood. This level of flexibility in development is a secondary need served by neural programs, the ability to process problems where an algorithm is not obvious, but by giving many examples, the program arrives at an algorithm itself. This allows programmers to attack problems that are not completely understood, or address flexible problems, both areas have a lot of room for improvement within computer science

3 Creation and Development

The idea of simulating the way in which living animals process data and information existed as early as the 1940s [14]. These early theories were limited by the understanding of neuroscience and no actual products were made. All work was theoretical and their schematics of neurons were effectively equivalent to simplified diagrams of modern logic gates, though they are not capable of as many functions. In 1954 advances in technology allowed Farley and Clark to simulate a network of neurons, which grew based on a learning rule purposed by Donald Hebb. "The assumption can be precisely stated as follows: When an axon of cell A is near enough to excite cell B and repeatedly or persistently takes part in firing it, some growth process or metabolic change takes place on one or both cells so that A's efficiency as one of the cells firing B is increased" [9]. These threshold based neuron models were capable of pattern recognition, albeit inefficiently. In 1958 Frank Rosenblatt created the Perceptron [8]. This simulation organized neurons into layers and limited connections between neurons to adjacent layers. Multiple binary inputs are fed into summation nodes, the output of which passes through a threshold gate, which converts this summed value into a binary output. From there, a mutable weight, or multiplication factor, operates on the binary output, and finally, two values are subject to a comparator gate, the result feeding back to increase the weight of the most used sub-path. Conventional tests for pattern recognition involve subset recognition. On this task, the Perceptron performs somewhat erratically. While it is possible if it involves negative input values, essentially not gates, if it includes strictly positive inputs it falls to luck and the order of pattern presentation in order to correctly determine between the original pattern, and the pattern which is a subset of it. In 1960 a model called Adaptive Linear Element was developed by Widrow and Hoff working at Stanford University [8]. This device used a least mean squares algorithm to, based on a given sample filtering process which serves as a teacher, arrive at an optimal values for the multiplication factors for a given pattern. This allowed a network to do more then recognize similar patterns. It created a process for inputing an optimal pattern, causing the network to evaluate the similarities as a sort of optimization problem. This increased the practical usefulness of neural network models.

4 Refinement and Advancement

While initial neural models were useful for pattern recognition, eventually the technology stagnated. This was largely due to the impossibility of modeling exclusive or gates [9], logic gates representing a function that is true only when one of the two inputs is true, and the other is false. This problem was due to the nature of the comparator gates, gates which evaluated the electrical signal put into the gate compared to the expected value for true or false, which were only capable of operating on a dual input-output level while feeding in positive or negative inputs. In other words the gates were incapable of discerning the difference between a zero signal and a negative and positive signal that canceled each other out which lead to difficulties in constructing the structures necessary to properly model logical processes [9]. However, these issues were initially glossed over. The issue was finally resolved by expanding the operation of the network series beyond simple binary positive or negative inputs. These included shifting weight factors to affect individual inputs, as well as allowing negative weights. This introduced the idea of an inhibitor factor, expanding on the processing ability of the networks themselves. These changes promoted multilayer networks in order to expand on these concepts. With an increase in layers, many wondered how the Least Means Squared rule, which is explained in section 9 would be applied across such a network. This resulted in the first Back-Propagation network. This method evaluates the error rate in each layer, with larger proportionality effects in the front-most layer, and decreasing towards the back-most layer closest to the input. These rates are then used to re-evaluate and reconfigure the weights, slowly but more precisely arriving at the desired pattern.

5 Current

Modern neural networks use back-propagation techniques to evaluate pattern matching, process control, decision making and other modeling problems [4], [15]. This has become the generally accepted method of constructing a neural net, although the precise process for teaching and formatting a given neural network remains specific to the problem itself, as with any sort of computational method. More recent developments in recurrent neural networks, which loop-back to the degree of becoming a dynamic temporal model, have achieved accolades in recent years, and show promise for future development.

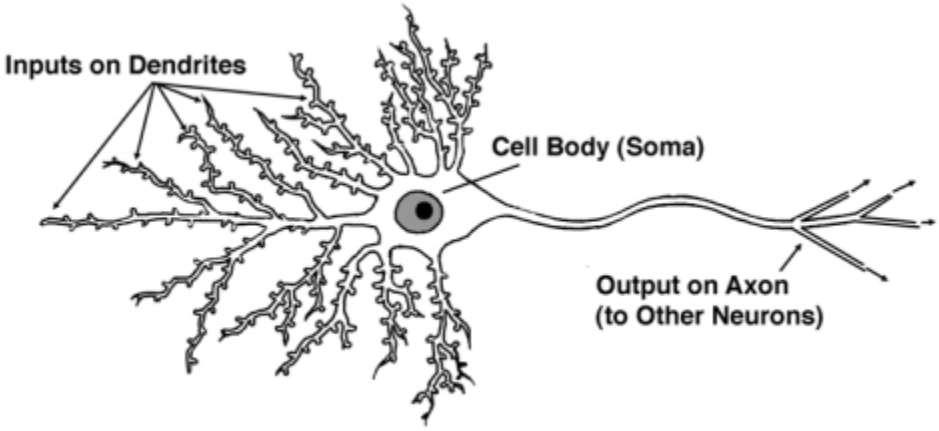
6 Biological Foundations

The original purpose of developing artificial neurons and networks was to emulate the biological processes in humans and animals and thus there are many similarities between artificially created neural networks and biological ones. [17] Because of these similarities and underpinnings, a brief explanation of the biological aspects will further the understanding of the technical aspects.

6.1 Neuron Structure and function

A neuron cell, Figure 1, is composed of three primary structures, the cell body, the axon, and multiple dendrites. The cell body performs standard cellular functions the details of which are unimportant to this topic, and also produces the chemical agents required for the reactions that facilitate communication between neuron cells. The axon is the transmitter of the neuron, conducting electrical impulses generated in the cell body to other

Fig. 1. A Neuron Cell



neurons. The dendrites are the receptors of the signals transmitted by either other neurons, or specialized cells, such as the rod and cone cells that form up the retina of the human eye. These 'sensor' cells serve to collect data and pass it along to the neurons composing the central nervous system [7]. A neuron cell functions by receiving electrochemical signals to the cell's dendrites, which transmit the signal to the cell body. Inside the cell body, reactions take place that determine whether or not the neuron initiates a chemical reaction to trigger a transmission through the cell's axon [7]. The exact nature of the chemical reactions and biological processes are a matter not concerning this topic and are discussed elsewhere, what is relevant is the process of reception of input signals, an evaluation of these inputs leading to a binary yes/no decision, and then the possible transmission of an outgoing signal. These three processes define the operation of both biological and artificial neurons.

6.2 Artificial Neurons

Artificial neurons are constructed to adhere to the same basic structure as their biological counterparts. An artificial neuron also has three primary components, the inputs, the body, and the outputs. Also, there exist in artificial neural networks components that are not full neurons, but simply input interfaces, similar to the rod and cone cells mentioned previously. These structures allow initial input values to be fed to the network. Artificial neurons as stated function in the same way as biological neurons; an input signal is transmitted to the neuron, which receives the signal and, based on a set of predetermined conditions referred to as a firing rule, decides whether or not to 'fire' transmitting a signal. This signal passes on to other neurons which make subsequent decisions based on their firing rules and by compounding many such operations networks of many cooperating neurons can solve complex problems such as pattern matching and recognition[14], [8].

6.3 Learning

The goal of creating artificial neural networks is to emulated the capability of biological neural nets to learn to solve problems. In creating artificial neural nets, the objective is to create a computational tool that is better equipped to handle certain types of problems. For example, while a human being has little difficulty sorting between pictures of cats and dogs even as a child, a computer scientist told to program an algorithm to allow a computer to do the same is faced immediately with many difficult questions. First how to classify dogs and cats from the computer's point of view, then how to analyze the images in order to differentiate between the two. Concepts such as the number of legs, or the shape of ears or tails are inadequate information for a computer to make an analysis, as such a device can only examine the images as arrays of pixels with varying color intensities. And even if it could, with the many varieties of cats and dogs, it would be hard to pin down which features to use to draw the line. Compounding the issue is that the pictures may be taken from different angles, or with different lighting, changing the pixel data without greatly affecting the overall nature of the image. While it may not be impossible to produce an algorithm that is able to take all of these variables into consideration and accomplish the task effectively, the level of complexity required will likely be significant. Artificial neural nets attempt to bypass this problem by removing the need to directly compose the algorithm[15]. Just like a living organism, neural nets can be trained to behave in a particular manner. The exact process involved varies between different types of neural nets and is discussed further in 7.3, but the general process is the same. By using a series of training inputs, the network creates associative patterns in the connections between neurons that correlate to the entire collection of training inputs. Thus the network itself composes the algorithm by forming associations between inputs presented in the training phase and the desired output

for each given training input[9], [10], [13] Related to the original example of dogs and cats, the training process would involve a network with two output neurons, one for dog, and one for cat. By composing the pixel data into the input neurons for a series of pictures of dogs and cats, each presented along with the data for the desired input, either dog or cat, eventually the neural net builds up information on the similarities between each set. In my own research I have conducted a similar experiment with simple shapes, the findings of which are addressed in 10.

7 TECHNICAL COMPONENTS AND COMPOSITION

In order to turn several interconnected neurons into a problem solving tool the neurons need to be properly organized. These organizational features play role in the learning processes and execution of the functions of the neural network itself, beyond the structure of individual neurons. The most critical of these can be divided into two groups based on whether they relate to neuron organization, or processing functions. The concept of layers, feed forward, and feedback structuring are related to the organization of neurons composing the network. Learning and firing rules are related to processing functions.

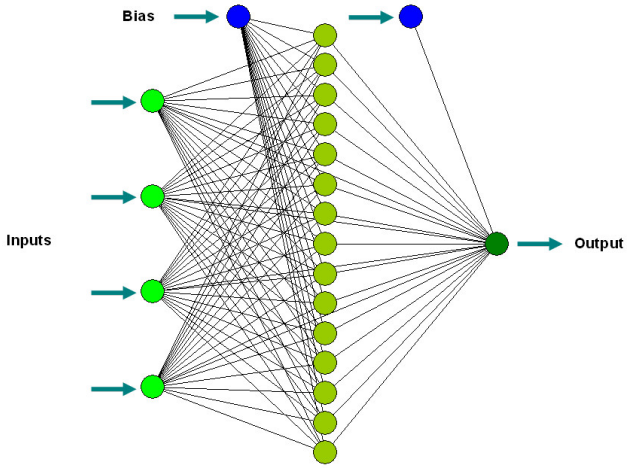
7.1 Neuron Layers

Neurons in a neural network are organized into layers as illustrated in Figure 2. These layers divide neurons based on function. In the most basic of neural networks there are only two layers, the input layer and an output layer. In this case, the first layer is directly controlled by input data. In other words, whether or not this first layer of neurons fires or not is a preset condition, and by altering the configuration of these neurons a user controls the information input into the neural net. The input from these neurons is passed on to the last layer, the output layer, and the firing behavior of the neurons in this layer constitute the results of the neural net's processing. For an example of how this process occurs, see 8. In more complex networks such as the net depicted in Figure 2 there are one or more intermediate, also know as hidden, layers which allow the neural net to perform more complicated logical calculations, as mentioned in section 4. This process will be further explained in 8.1

7.2 Feed Forward and Feed back

The layers of a neural network, once assembled from neurons, can be themselves structured in one of two ways: either consecutively, with each layer taking input from the layer before, closer to the original input neurons, and outputting to the layer after, closer to the output neurons, one after another in a stacked arrangement as displayed in 2. This is referred to as a feed forward neural net. Alternatively, the layers may be interconnected without concern for directionality. This configuration, know as a feed back neural net, produces

Fig. 2. A Three Layer Neural Network [2]



more complex interactions throughout the network because of the possibility for inputs to feed back upon each other. This results in a fundamental difference in network behavior; while a feed forward network receives inputs then computes them as the inputs filter down through the neurons in each layer eventually arriving at a definite output, a feed back network constantly recalculates each neuron's condition, firing or not, and any output is valid only at that time. Depending on whether the network achieved a state of equilibrium or not, the output of such a network may change several times or constantly[16], [9] A possible configuration for a feed back network is displayed in Figure 3. In this report, all of the neural network formats examined in detail are examples of feed forward networks, mainly due to the complexity of operating feed back networks, as well as the limitations of the experimental tools we possess.

7.3 Neuron Processing

In order to turn layers of neurons into a computational tool, the firing of neurons and the passing of signals between layers needs to be organized constructively. These signals are controlled by the use of firing and learning rules. A firing rule is the concrete process by which a neuron decides based on the inputs it receives whether or not to fire. Most frequently this rule takes the form of a Heaviside Function[14] Figure 4, also know as a unit step function. This is simply to say that a variety of inputs are reduced to an output of either 0 or 1, 0 indicating in this case that the neuron does not fire, a 1 indicating that it does. Thus a neuron receives inputs from other neurons then combines them, usually through summation, and then uses its own firing function to determine the appropriate behavior. In order for the firing rules of each neuron to function as expected, the inputs feeding into the neurons need to be properly calibrated. While the exact process varies from

Fig. 3. Feedback Neural Network[3]

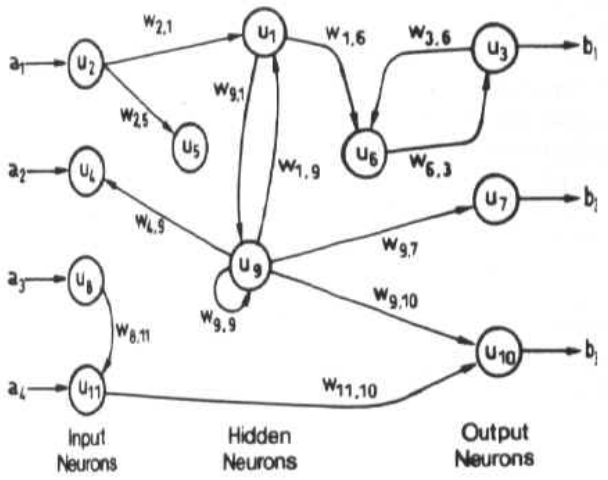
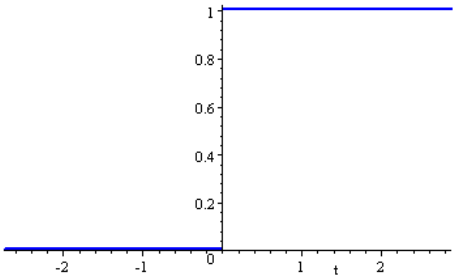


Fig. 4. Heaviside Step Function[1]



network to network, examples of which can be found in figures 9 and 8, the process is always accomplished via the adjustment of input weights through a learning rule. A learning rule dictates how input weights are updated during the training process of a neural network. An input weight is a decimal number between 0 and 1 assigned to each connection between neurons, and affects how great an impact the value of that input affects the calculations of the neuron it is feeding into. The exact equations vary between different types of neural networks and will be addressed in each respective section. During the training process, with each training input the learning rule determines how to balance the weights of all of the connections in order to best adapt the network towards the goal of conforming to the training patterns, thus affecting the overall behavior of the network.

8 Perceptron

Perceptrons were the first type of neural network to be developed and are also one of the simplest. A simple perceptron is depicted in Figure 5. A perceptron consists of two layers of neurons, an input layer and an output layer consisting of a single neuron. The math behind perceptron learning is based on linear separation of points. The connections between neurons in a perceptron work to form a simple linear equation and by adjusting the slope and intercepts of this line, a perceptron network learns to separate data into two groups, which is why only one output neuron is needed. When learning each test input is like a point plotted on the graph. With the desired output associated with this point, the perceptron network can use its current equation to decide whether or not the point is landing on the side of the line it is supposed to. If it is not, the line is adjusted so that it does. The exact nature of these adjustments depends on the particular learning rule and the learning rate, with a higher learning rate leading to more dramatic adjustments. This process continues until every point of training data is in the appropriate zone as depicted in 6. This method causes one substantial limitation, and that is that perceptrons are only capable of processing linearly separable data sets.

Fig. 5. Perceptron

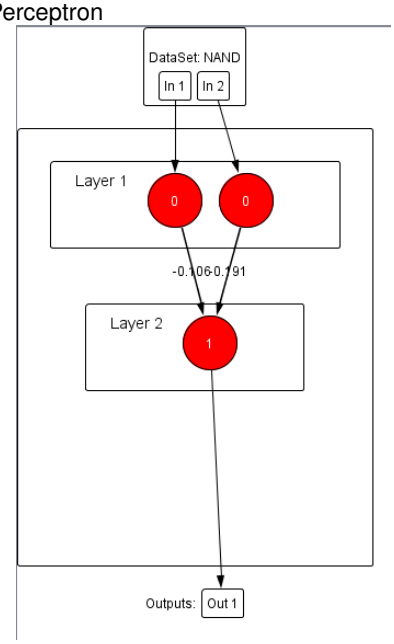


Fig. 6. Linearly Separable Data

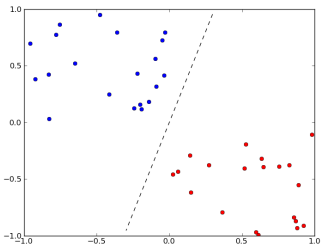
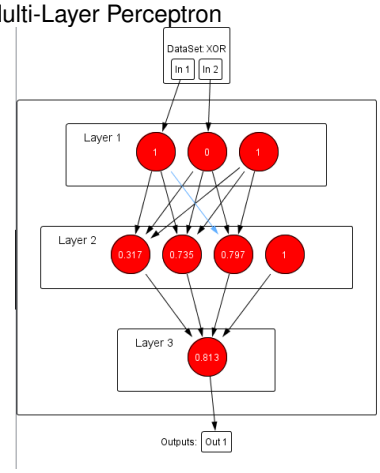


Fig. 7. Multi-Layer Perceptron



8.1 Multi-Layer Perceptrons

Multi-layer perceptrons overcome the limitation of linearly separable data sets by using a more complex structure of neurons that includes one or more hidden layers, as well as bias neurons. Multi-layer perceptrons use nonlinear activation functions. The primary firing rules are sigmoid functions, which allow for varied output functions as opposed to linear or step firing rules. This is the one of the features that allows multi-layer Perceptrons to compute data sets that are not linearly separable. The other reason multi-layer networks are able to overcome this limitation is a different learning rule. Mult-layer networks do not use Perceptron learning but instead us backpropogation. Backpropogation involves calculating the rate of change of the error based on a given weight. This is functionally calculating the derivative of the function of error as a factor of the given weight. Then, the weight is adjusted relative to the learning rate towards the direction of the highest gradient of decrease in error. When this equation is generalized across every weight in the network, a direction of greatest descent for total error in the network can be discovered, and all weights adjusted in this direction[15], [16]. The only problem with this learning rule is that while it always leads in the adjustment of weights towards a more favorable solution, it is impossible to tell whether or not it is the most favorable solution. It may be possible that the rule adjusts the weights towards a local minimum, when a more optimal configuration exists. That drawback aside, the multi-layer Perceptron is preferable due to its ability to process a much wider variety of data types.

9 Adaline

Adaline neural networks were devised shortly after the invention of perceptrons, well before multi-layer perceptrons were considered, by researchers also interested in creating an artificial version of a biological neural network. Adaline neural networks are built around a series of n inputs X_{1-n} each with a corresponding weight denoted by W_{1-n}. The singular output is the summation of each of these weighted inputs in the form of

$$Y = \sum_{j=0}^{n} X_j W_j + \Theta$$

Where Y is the output, and Θ is some constant, referred to as the bias. In other words, this equation is a mathematical representation of the neural network itself. To minimize the error, a learning rule called least mean squared is used. This rule is aimed at reducing the error in the network to, as the name suggests, the least mean squared error possible. The least because error is undesired, mean meaning the average across all patterns, and squared to treat positive and negative errors equally, as the original devices used binary $+1$ or -1 inputs. Thus for the error E, Y^{actual}, and $Y^{desired}$

$$E = (Y^{actual} - Y^{desired})^2$$

For all $Y_{1\to n}$ the training algorithm is then arrived at through algebraic calculation

$$Y = X^T W = W^T X$$

By substituting this value for Y^{actual} in the error equation

$$E = Y^{desired} - X^T W = Y^{desired} - W^T X$$

Now, because of the desire to weight positive and negative error equally, the square of E is used

$$E^2 = Y^2 - 2YX^T W + W^T X X^T W$$

This represents the squared error. To arrive at the Mean Squared error, ξ, the calculations must be applied across all of the example sets with fixed weights.

$$\xi = Ex[E^2] = Ex[Y^2] - 2Ex[YX^T]W + W^T Ex[XX^T]W$$

Because of the unwieldiness of this statement, $Ex[YX^T]$ and $Ex[XX^T]$ are usually rewritten, in this case as Q and R respectively

$$\xi = Ex[E^2] = Ex[Y^2] - 2QW + W^T RW$$

When written in this fashion the formula for the Least Mean Squared Error is more easily recognized as a quadratic function of the sets of weights [17]. This function can be plotted on an n+1 dimensional field where n is the number of weights, with the final axis representing ξ. By calculating the minimum value for ξ, or a value within the acceptable margin of error, a point can be found which the coordinates thereof are the ideal weight values for the training set. These networks are capable of computing some nonlinear datasets, but not all. This depends on the specifics of the equations involved, and generalizations are difficult. However, the ability to solve for some nonlinear datasets means that Adaline networks are more capable the basic Perceptrons, if not without any drawbacks.

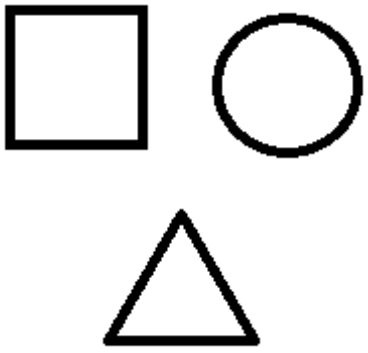
10 Image Recognition

One of the most effective demonstrations of neural networks and their varied strengths, weaknesses, and applications is image recognition. Image recognition is a very difficult problem for the reasons explained in the example of cats and dogs in section 6.3. Despite these difficulties, image recognition has many applications and so is still a desirable problem to solve. As mentioned before, the ability of neural networks to create their own classification systems makes neural nets particularly apt for image recognition. We have personally conducted a degree of research into image recognition neural networks using Neuroph Studio 2.8, a Netbeans based IDE exclusively for creating neural nets.

10.1 Structure

The structure of an image recognition neural network is based on the number of pixels in the images to be analyzed. In the tests we performed, we used a standard of 20x20 pixels in gray scale. This leads to an input layer of 401 neurons, one for each pixel and one bias neuron. If we were to have used color imaging, it would have required four neurons for each pixel in order to handle the different color weights. The images we used for testing are all 100x100 pixels, which means that they must be compressed down to the appropriate size in order to be analyzed by the network, which Neuroph does automatically. The neural net in Figure 8 is the net we constructed and trained to differentiate between the outlines of basic geometric shapes. we used the images in Figure 9 as the training images, and the weights in the network were constructed off of these. One of the interesting aspects of an image recognition network as opposed to the other networks discussed so far is that there are multiple output nodes. In this case, there is one output node for each training image, and the value outputted to that neuron represents the level of similarity between the input image and that training image. In Figure 8 there is one additional output neuron that was put in as a control during my experimentation.

Fig. 9. Training Images



10.2 Recognition

Once the structure of the network is in place, training involves a method similar to backpropogation by feeding in the values of the pixels of a training image, along with an input of 1 in the corresponding output neuron and 0 in the other and then propagating the errors in the network back towards the inputs, adjusting the weights accordingly. When this process is complete, a new image's pixel data will output the affinity, or degree of similarity, of the test image for each of the training images. Figure 10 shows the results of a series of test images fed through the network. While this network proved capable of easily recognizing ring shapes, it had some difficulty distinguishing between triangular and square outlines. This is likely a result of similarities in the compressed images.

11 Open Research Questions

Current research topics in neural networks tend towards tuning the neural net frameworks already developed toward new problem types. While much of my report has glossed over the specifics of the different activation functions and especially with regard to multi-layer Perceptrons, these functions play a significant role in the behavior and learning patterns of the network, and thus choosing the most optimal function plays a very important role in solving specialized problems[4], [6], [10]. Therefore researchers are constantly testing new activation functions to look for further applications. The other ongoing field for research is learning rules. In this report we focused on four different network types and learning rules, but in reality there are many dozens of different learning rules, and many networks have more then one option to chose from. There is constantly ongoing research into which rules are most efficient and what new rules might be devised.

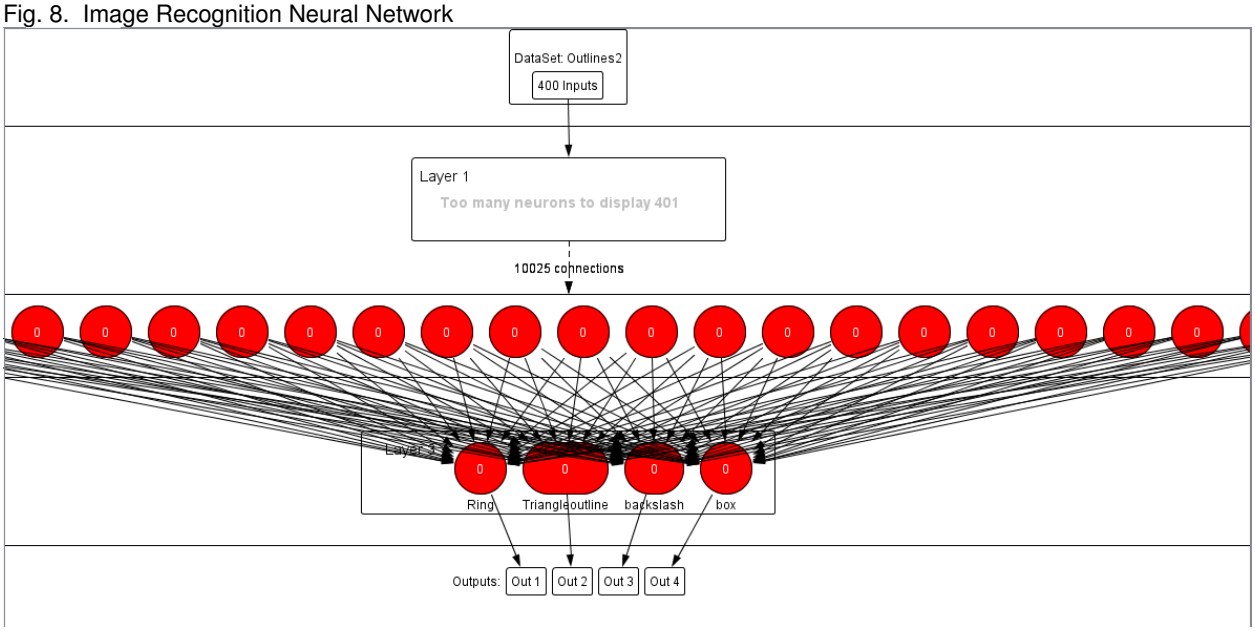
12 Growing Applications

One of the predominant areas of interest in neural networks in the future is the potential for more applications for the technology.[5], [12], [4] There are also many possible areas of expansion in existing uses for neural network technology.

12.1 Banking and Financial Modeling

One such example is a study conducted in the Egyptian financial sector in 2008[4]. The experiment compared neural networks with conventional algorithmic methods for predicting the probability on loans given out by banks. In this case, neural networks proved superior in both accuracy and in processing speed. This due to two reasons. The first is that financial transactions are extremely well documented, and there is an almost excessively large data pool from which to choose the training data. The second factor is that neural processing is very front heavy, meaning that while training the neural network can take a great deal of time depending

Fig. 8. Image Recognition Neural Network



on the training set used, the actual processing undertaken by a trained network is fast. These two factors make neural networks particularly suited to financial modeling problems which often include difficult to model principles and require taking account of complex factors such as herd mentalities which can be difficult to calculate explicitly[4]. The recent political upheaval in Egypt has prevented the implementation of any of the systems suggested by this experiment, however as the region hopefully stabilizes over the next few years, many of these advances may see public use and, if they prove successful, could be adopted by other countries.

12.2 Facial Recognition

Another area that has seen substantial and promising experimentation with neural networks, facial recognition and image recognition are areas that have proven difficult to overcome for traditional algorithmic processes, but neural networks are adept at solving[15]. There are many practical applications for accurate facial recognition software such as biometric security and identification. Security is the most driven field of research in this area due to the number of highly desirable applications. Facial recognition technology is already used in many security camera systems in banks, airports, retail outlets, and government faculties[15]. However there have been many circumstances where such systems have failed to detect targeted individuals due to changes in the angle of the photograph from those in the database. The current areas of research are aimed towards producing a combined system that uses neural networks along with 3D cameras and modeling software to create a digital figure that can be recognized from a variety of angles[15]. Such a system will hopefully be capable of distinguishing possible individuals regardless of their clothing by gauging bodily proportions, and will also foil many of the traditional methods for bypassing facial recognition software, such as minor adjustments to the height of the ears or nose which while they seem almost identical to a human observer, will not register on traditional algorithmic image searches. The developers of this software hope to overcome this obstacle by using neural networks to recognize these images in a way more similar to the process a human would use, and thus provide more accurate matches against a database. This software is already being tested in a handful of airports, operating alongside current systems and if it proves successful could become the dominant security feature in airports in the next five years, providing an alternative to the current methods which are slower and more invasive to travelers privacy.

12.3 Biometrics

Similar to facial recognition, biometrics is the process of verifying an individuals identity through the use of unique biological features. The main reason neural networks could prove useful to this industry is one of the characteristics of neural networks stated previously, namely that they are very front-heavy in processing. This means that while it may take some time for the initial learning process to teach the network software to recognize an individual, once so imprinted, the actual process for verifying their identity is very quick. Also, once the network for recognition has been generated, it no longer requires a powerful processor to run, and could be easily transfered to much simpler, power-efficient, and mobile devices. This could provide an alternative for more cumbersome passkey and code systems, and also

reduce or eliminate the need for physical identification products[15]. This technology is already available but due to the lack of testing and publicity has yet to enter into wide scale use however, with advances in mobile technology and a desire for increased security, it is very likely that in the next five to ten years biometric identification will become much more popular.

13 Learning Algorithms

The effectiveness of any given neural network is strictly dependent on the learning rule used to form the connections within it. Considering more effective learning rules can produce more accurate networks faster, advancements in these rules are important to the growth of neural networks. There are too many different learning rules already devised to list here, and there are just as many more in development. Many of these rules are attempting to apply probabilistic prediction rules to learn not only from the test data, but also from extrapolations of that test data[10]. This research began three years ago and should be nearing completion within a year or two however as of the last report the results are still inconclusive. It proves to be an effective tool at more accurately tuning networks that already had a great deal of consistency in the data. To that end, it makes networks that are already highly accurate even more accurate, but has little effect on networks that are initially more inconsistent due to limited sample size.

14 Biological Possibilities

One of the most interesting possibilities for artificial neural networks is the possibility for a change in the structure medium. There are a handful of neurobiologists working together with computer scientists towards creating an artificial neural network made from cloned neuron cells. So far such networks are limited to no more then 30 neurons, and there is also nearly no control over the firing rules of the neurons. This research is still in its earliest stages, however scientists hope that in the next year or two, they will be able to more successfully control the firing behavior of the neurons and in the process make structuring and training these artificial biological neural networks possible [13]. Such networks are unlikely to prove useful however as they difficult to create and control, and thus not cost effective.

15 Conclusions

Neural networks provide a method of computation that is distinctly different from traditional algorithmic programming, and allow a machine to do the difficult work of hashing out the details of algorithms itself by learning through example. This idea draws on inspiration from biological sources which have proven to be highly adaptable, animal brains. There are many different types of neural neural networks such as perceptrons and adaline networks which have clear limits on their data processing capabilities and yet still possess the ability to be adaptively developed by through learning rules. There are also more powerful networks like the backpropogating multi-layer perceptrons that have fewer limits on data processing, but no matter the type of neural network it will always be defined by the distinctive method of development through learning rules and training data, and this trait is what makes artificial neural networks a useful tool for solving problems that cannot be addressed by other programming methods.

References

[1] Image from http://intmstat.com/laplace-transformation/$1_lap_unitstepfns18pt_{10.png}$.

[2] Image from http://www.biomedcentral.com/content/figures/1472-6750-7-53-2-l.jpg.

[3] Image from http://www.doc.ic.ac.uk/ nd/surprise$_96/journal/vol4/cs11/report.n$

[4] Hussein Abdou, John Pointon, and Ahmed El-Masry. Neural nets versus conventional techniques in credit scoring in egyptian banking. *Expert Systems with Applications*, 35(3):1275 – 1292, 2008.

[5] Gail A. Carpenter and Stephen Grossberg. The art of adaptive pattern recognition by a self-organizing neural network. *Computer*, 21(3):77–88, 1988.

[6] Patrick Connally, Kang Li, and George W. Irwin. Prediction-and simulation-error based perceptron training: Solution space analysis and a novel combined training scheme. *Neurocomputing*, 70(46):819 – 827, 2007. Advanced Neurocomputing Theory and Methodology Selected papers from the International Conference on Intelligent Computing 2005 (ICIC 2005) International Conference on Intelligent Computing 2005.

[7] Ellen Dr. Covey. Structure and cell biology of the neuron.

[8] S.I. Gallant. *Neural Network Learning and Expert Systems*. A Bradford book. MIT Press, 1993.

[9] Stephen Grossberg. Contour enhancement, short-term memory, and constancies in reverberating neural networks. In *Studies in Applied Math*, pages 213–257, 1973.

[10] Milton Roberto Heinen and Paulo Martins Engel. An incremental probabilistic neural network for regression and reinforcement learning tasks. In *Proceedings of the 20th International Conference on Artificial Neural Networks: Part II*, ICANN'10, pages 170–179, Berlin, Heidelberg, 2010. Springer-Verlag.

[11] Geoffrey E. Hinton. A better way to learn features: Technical perspective. *Commun. ACM*, 54(10):94–94, October 2011.

[12] G.W. Irwin, K. Warwick, K.J. Hunt, and Institution of Electrical Engineers. *Neural Network Applications in Control*. IEE control engineering series. Institution of Electrical Engineers, 1995.

[13] Gwang S. Jung and Venkat N. Gudivada. Automatic determination and visualization of relationships among symptoms for building medical knowledge bases. In *Proceedings of the 1995 ACM Symposium on Applied Computing*, SAC '95, pages 101–107, New York, NY, USA, 1995. ACM.

[14] Nikola K. Kasabov. *Foundations of Neural Networks, Fuzzy Systems, and Knowledge Engineering*. MIT Press, Cambridge, MA, USA, 1st edition, 1996.

[15] S. Lawrence, C.L. Giles, Ah Chung Tsoi, and A.D. Back. Face recognition: a convolutional neural-network approach. *Neural Networks, IEEE Transactions on*, 8(1):98–113, Jan 1997.

[16] C Stergiou and D Siganos. Neural networks.

[17] Bernard Widrow. The lms algorithm and adaline. part i - the lms algorithm.

Appendix

This paper depended considerably on the fundamentals we learned in CSCI 310 and 317. Algorithms in par-

ticular gave us the experience necessary to understand the traditional methods of computing and computational methods like finite state automata and Touring machines. These provided an exceptional contrast to artificial neural networks when it came time to discuss the strengths and advantages as well as the possibilities that neural networks offer. It also was some of what motivated the choice of topic. In algorithms we studied genetic algorithms and in doing so briefly touched on neural networks, but we felt we never covered the topic and so held onto a lingering curiosity. Studying bioinformatics was also a necessity for this project. That class provided the actual fundamentals on neural networks themselves, as well as the insight into the development process for technologies inspired by nature. Bioinformatics also provided some background on possible applications for neural networks given that we studied problems that traditional computational methods proved highly inefficient at addressing. This is another area where Algorithms proved useful, in analyzing and understanding the difficulty of solving given problem types. Furthermore, our entire study of computer science in general, considering the labs and projects, that provided the fundamental understanding of the structure and functioning of computer code. Without this at a minimum it would not have been possible to synthesize any of the comparisons. Finally, this experience also proved necessary in conducting the experiments with image recognition neural networks. The software, Neuroph Studio, used to construct the network is based in the Netbeans IDE, which we used in software development to generate the CMC (chose my college) website project. Without this experience, it would not have been possible to accomplish any of the research due to how difficult to operate and unreliable Neuroph studio proved to be. This project has proven enlightening in several areas. First It has offered the opportunity to explore a new programming method that none of the classes ever provided the opportunity to use. This proved helpful in grasping the distinct limitations of traditional methods. Second off, it provided an opportunity to actually program in a completely new type of language. Thus far courses had only addressed functional and object oriented languages, apart from brief forays into Touring machines and finite state automata on tests questions, but never actually to the level of a usable program. Hopefully this experience will lead to a better understanding of the potential of other computing methods, and an increased level of adaptability when coding.

Fig. 10. Image Tests