Evolutionary Artificial Neural Networks:

An Application to Facial Recognition

David Needell, Sawyer Bowman, Phoebe Bumsted, Ryan Barrett

Computer Science 3445: Nature Inspired Computation

Bowdoin College, Spring 2015

Professor Majercik

Abstract:

We explore the effect of a limited connection topology for an Artificial Neural Network (ANN) with respect to a facial recognition application. From this initial configuration of the multilayered feed-forward network (MLFN), the topology is then updated through use of a Genetic Algorithm (GA). From this hybridization of both the Artificial Neural Network Algorithm (ANNA) and the Genetic Algorithm (GA), we find that the Evolutionary Artificial Neural Network Algorithm (EANNA) resulted in a higher percentage of correctly classified images than the standalone ANNA.

Keywords:

Artificial Neural Networks, Multi-layered Feed-Forward Network, Genetic Algorithm, Evolutionary Algorithm, Topologies

1. Introduction

The cognitive architecture of the human brain remains one of the most widely studied topics in biology. Simplistically, the brain can be described as a static biological neural network - a web of neurons that send electrical signals through channels to other neurons. How the brain functions on a deeper level and its architectural intricacies and neural plasticities still remain very much a mystery. This simplified model represents neural nodes with static channels - receiving the same inputs from the same set of synapses constantly. In reality, the brain's neural connections are not constant. Over time, neural pathways change and evolve from a given input. This research attempts to hybridize the simplified model of Artificial Neural Network Algorithms (ANNA's) and Evolutionary Algorithms (EA's) to establish a more accurate representation of the brain and its evolving neural topological connections.

This research combines a multi-layered ANNA with a Genetic Algorithm (GA) and applies this hybridization to a facial recognition task. Therefore, we aim to illustrate the differences in how accurate the ANNA algorithm is at recognizing faces when *(i)* the ANNA is not evolved and *(ii)* the ANNA is evolved under the GA with respect to topological connections. This comparison will illuminate whether or not evolving the topologies of a given ANNA benefits learning.

For this research, the testing files consist of 20 different human faces, each black and white picture of each face taken from different angles, with different emotions, at different resolutions[1]. An individual image consists of a two dimensional grid of integer values. The grid size changes depending on the resolution of the image. The integer values are in the range of *[0, 255]*, where *0* is completely black and *255* is completely white. The objective of the algorithm is to correctly identify which image belongs to which face.

Thus, the ANNA structure begins with an input layer, where a single input node corresponds to an integer value of the image grid. The input layer is then connected to a hidden node layer. For a standalone ANNA, the input layer is fully connected to the hidden node layer. The hidden node layer, consisting of 20 nodes, then connects to the output node layer, also consisting of 20 nodes. A given node in the output node layer represents a face from the data set (e.g. output node 13 would represent the 13^{th} person in the data set).

Applying a GA to this ANNA implies evolving either/both the topological connections between the input node layer and the hidden node layer or evolving the edge weight updates of the input node layer to the hidden node layer. For this

research only the topological structure of the ANNA is updated in order to isolate how the effect of varying topologies influence the learning of the algorithm.

Our results reveal the increased accuracy in the EANNA over the ANNA. A .3 learning rate performed consistently better than a .2 learning rate. Furthermore, while the hybridized EANNA outperformed the traditional ANNA with respect to image correctly categorized, the cost of this increase in performance came from the runtime of the algorithm.

Section 2 describes the ANNA algorithm and structure with mathematical rigor. Section 3 discusses the GA. Section 4 describes how this research hybridizes both the ANNA and GA into one algorithm. Section 5 discusses the problem files in greater detail. Section 6 outlines the experimental methodology of this research by discussing parameters used, tests run, and statistical analyses applied. Section 7 describes the results found from these tests, and Section 8 concludes these results and discusses future directions for this research.

2. Artificial Neural Networks[1]

As previously mentioned, ANNA's derive from the cognitive architecture of the brain and how neurons communicate. Physically, neurons in the brain receive electrochemical inputs from thousands of other neurons through channels, known as synapses. One of the fundamental functions inherent to a neuron lies in its firing mechanism. Specifically, a neuron will only fire if the sum of all electrochemical inputs reach a level such that it exceeds a particular threshold. If the threshold is reached or exceeded, the neuron will fire and it will allow the electrochemical signal to pass through to other neurons.

While this is by no means an exhaustive conceptualization of communication throughout the brain, this simplified model allows for various ANN algorithms to be created. From this concept of neurons in the brain, a rudimentary algorithm can be adapted simulating neurons as nodes in a network. Figure 1 shows a depiction of an artificial node used in ANNA's. From this neural representation, we can see that the node will have an output value of a_i determined from the activation function applied to the sum of all weighted inputs.

[1] §2 is taken partly from *Artificial Neural Networks: an Application to Handwritten Digit Recognition*. D. Needell, et. al. Bowdoin College 2015.

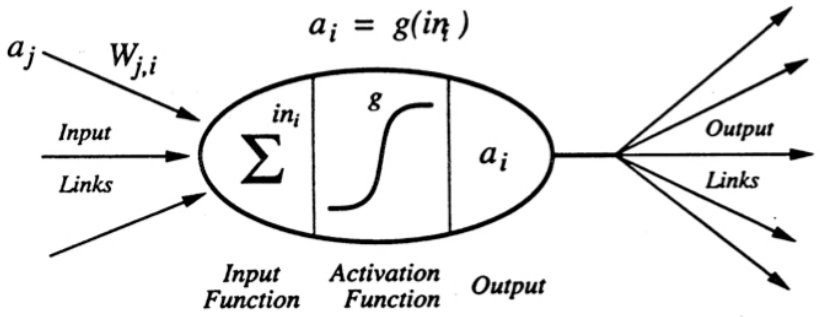

Figure 1: an artificial representation of a neuron. Note that a_j is the activation level coming from node j, $W_{i,j}$ is the weight of the edge between nodes i and j, and g is the activation function used to determine if the intrinsic threshold is exceeded.

For a neuron in a brain, the activation function is typically quantized as a one or a zero - i.e. either the neuron fires or it does not. Mathematically, this function can be represented as a step function such that:

$$g(x) = 1, \text{ if } x \geq \text{threshold value}$$
$$g(x) = 0, \text{ if } x < \text{threshold value}$$

However, this research makes use of the sigmoid function as the activation function shifted slightly, such that:

$$g(x) = \frac{1}{1+e^{(-x+0.5)}}$$

The sigmoid function is used for two reasons: *(1)* the function allows nodes to output small values even when the threshold is not reached, and *(2)* the function is a continuous, differentiable curve. Contrastingly, notice that the step function is not differentiable at $x = t$.

Furthermore, this research makes use of the neural structure known as the multi-layered feed-forward network (MLFN). Fundamentally, a MLFN is synonymous with two single-layered, feed-forward networks combined into one. As shown in Figure 2, a MLFN consists of an input node layer of neurons connected to a hidden node layer of neurons. Finally, the hidden node layer of neurons then connects to the output node layer. Figure 2 shows an example of a fully connected MLFN with only one output node.

The next step in a neural network is to train the nodes such that the edge weights adapt to the specific problem in order to output the correct response. To wit, there are two sets of edge weights that need to be updated accordingly: *(i)* the edge weights between the input nodes and hidden nodes, and *(ii)* the edge weights between the hidden nodes and output nodes.

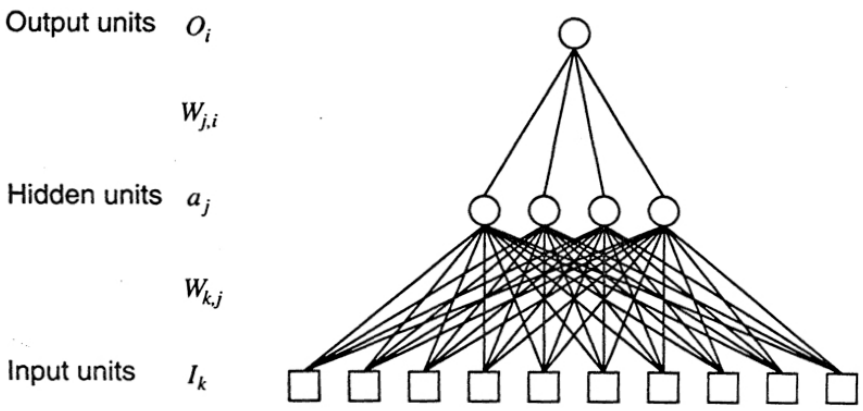

Figure 2: a simplified conceptualization of a MLFN. In this diagram, there exists only one output node and the topological connection from the input nodes to the hidden nodes and hidden nodes to the output node are fully connected, respectively.

Specifically, the Edge Weight Update Rules follow:

Hidden-to-Output Edge Weight Update:

$$W_{j,i} \leftarrow W_{j,i} + (\alpha * a_j * Err_i * g'(in_i))$$

where:

$W_{j,i}$ *is the weight from hidden node j to output node i,*
α *is the learning rate parameter,*
a_j *is the activation level of the hidden node j,*
Err_i *is the (Target Value - Output Value) of output node i,*
$g'(in_i)$ *is the first derivative of the activation function of the in (not shifted), and in is the summed inputs multiplied by their respective weights of output node i.*

Input-to-Hidden Edge Weight Update:

$$W_{k,j} \leftarrow W_{k,j} + (\alpha * I_k * \sum_i (W_{j,i} * Err_i * g'(in_i) * g'(in_j)))$$

Also note that the *Output Value* is defined as the maximum activation level across the output node layer, such that:

$$Output,\ O = max\{\ g(\sum_i W_{j,i} a_j)\ \}$$

Therefore, as more tests are run, the edge weights change in accordance with the error and the derivative of the activation function in order to train the MLFN network. Figure 3 shows an example of a partially connected topological ANNA.

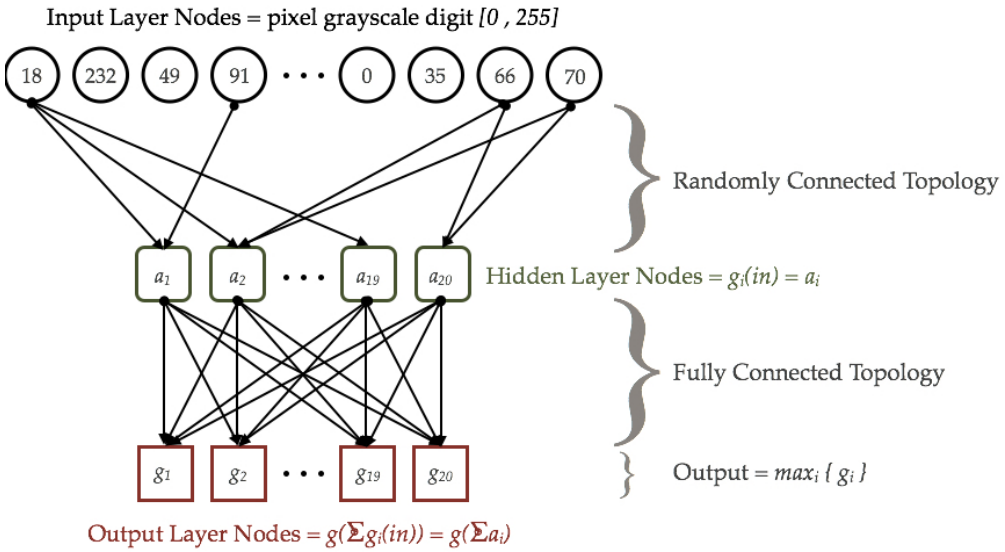

Figure 3: An example of a partially connected ANN.

3. Evolutionary Algorithms and Artificial Neural Networks

Evolution is a process by which a group of individuals in a given population adapt to their environment in order to optimize their chances of survival. Evolution occurs in all living organisms and dictates which individuals within a population survive and pass on their genetic information to their offspring. From this natural phenomenon, several Evolutionary Algorithms (EA's) have been designed in order to imitate this evolutionary process and apply it to relevant problems in computer science, mathematics, physics, and engineering.

This project adopts a version of an EA known as the Genetic Algorithm (GA). The GA derives from a close study of eukaryotic breeding and mutating habits - evolving a population of individuals based on selection, breeding, and mutation. These three fundamental components of the GA evolve a given population in order to optimize the overall best fitness found.

For this research, the GA is applied to evolving the topologies of a given MLFN. Specifically, the population of individuals used in the artificial evolutionary process is a population of various input-to-hidden node topological connections. An individual represents one possible organization of the neural connections between the set of input nodes and the set of hidden layer nodes. Note that for this research, the genomic material inherent to the GA stems solely from the topological organization between the input and hidden layer nodes. Thus, the hidden to output layer nodes remain fully connected throughout the entirety of the evolutionary process. Therefore, after a set amount of epochs of training an ANNA population, the population is thus evolved in accordance with the GA. Figure 4 shows a

conceptualization of the overall Evolutionary Artificial Neural Network Algorithm (EANNA).

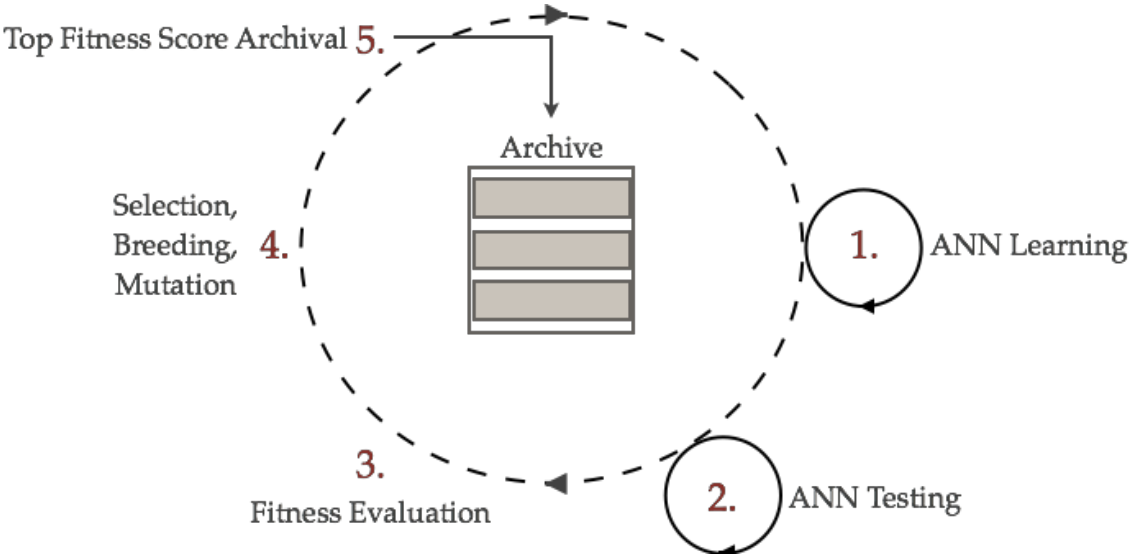

Figure 4: the EANNA conceptualization.

3.1 Fitness and Selection

Given a population of ANNA topologies, each individual of the population (i.e. each unique topological ANNA) is assigned a value corresponding to the relative fitness of that ANNA. For this research, the fitness value is related to the percentage classified correctly (PCC) from the resulting tests run. Specifically, the fitness is calculated such that[2]:

$$\Phi(a) = PCC_a^{(t)}(D_{test})$$

where:

$\Phi(a)$ *is the fitness of individual a,*

$PCC_a^{(t)}(D_{test})$ *is the percentage classified correctly of the individual a at generation t for the testing data set,*

From this fitness calculation, three different methods of the selection process are used to determine which individuals occupy the breeding pool. These methods are: Rank Selection (RS), Tournament Selection (TS), and Boltzmann Selection (BS). Common to all three selection methods, the fitness score of an individual, $\Phi(a)$, is minimized to reach optimality.

RS gives a probability to each individual proportional to that individual's rank. Specifically:

$$Prob_i = \frac{i}{\sum_{i=1}^{N} i}$$

Where $Prob_i$ is the probability of the i^{th} individual of the sorted, ranked list of N individuals – the more fit individuals occupying the bottom of the list (i.e. the individuals with the smallest scalar fitness score are placed at the bottom). An important facet of RS is that even lower ranked individuals are assigned a non-zero probability to enter the breeding pool.

This research also explores TS as another method for choosing the breeding pool subset. First, some constant integer number, *M*, of the population is chosen at random. From these *M* individuals, the subset is then ranked and the best *k* individuals are picked, where *k* is some constant natural number such that $k < M$.

BS is the final selection method applied to this GA. For BS, the selection technique operates closely to Rank Selection. Specifically:

$$Prob_i = \frac{e^{f_i}}{\sum_{i=1}^{N} e^{f_i}}$$

Where f_i is the fitness of the i^{th} individual.

3.2 Breeding

Once the breeding pool of individuals is selected, the next step is to breed the population subset in order to generate new individuals. It is important to note that two parents may produce offspring identical to themselves. This probability is determined by a parameter denoted as the crossover probability. For this research, a bit string represents the given topology of an individual.

There are two types of breeding that are considered for this particular problem: One-Point Crossover (OPC) and Uniform Crossover (UC). Given two individuals, i.e. two ANNA topological connections between input and hidden nodes, OPC breeds them by randomly selecting an index of the bit string – called a breakpoint. Given this breakpoint, the two individuals then exchange the equally spaced subsequences to form two new individuals – the offspring. Thus, each parent will produce one offspring. See figure 5 for a conceptualization of OPC. Alternatively, UC has the potential to swap every single index of one sequence with the other. UC can be thought of as having *N* breakpoints in the bit string, where *N* is the length of the string.

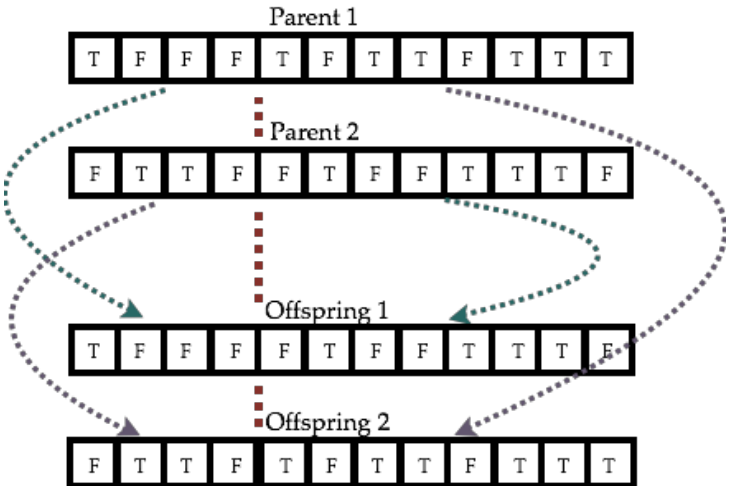

Figure 5: One Point Crossover (OPC) on a given input nodes' connection to the Hidden Node Layer

3.3 Mutation

Mutation only affects the offspring of two individuals. Whenever a new individual is created either by OPC or UC, the mutation probability determines whether or not to change an index's connection to another node. For example, if we are to mutate an individual's i^{th} index of the bit string, and the i^{th} index previously contained no connection to that index, we would then change it in order to signify a new connection. The mutation probability constant is held at relatively low values in order to provide a more stable population to evolve.

5. Problem Files

For this research, a data set consisting of 32 different photos for each of the 20 individuals is used in order to both train and test the EANNA. Specifically for each individual, the resolution of the image: *(128 x 120), (64 x 60), (32 x 30)*; the orientation of the individual: *straight, left, right up*; the facial expression of the individual: *neutral, happy, sad, angry*; and the eye state of the individual: *sunglasses, no sunglasses* are used to vary the style of the image. In total, 640 images are used to train and test the algorithm. Figure 6 shows a sample of these variations on one individual.

Note that the files are read in as a two dimensional array of pixel integers ranging from *[0, 255]* such that pixel integers of *0* and 255 correspond to black and white, respectively. Therefore, taking into account these problem file formats, we see that the input node layer consists entirely of the integer values corresponding to the gray

scale of a given pixel. Figure 5 shows a given configuration of the EANNA with input node values.

Figure 6: an example of the testing data used for the EANNA.

6. Experimental Methodology

This research focuses on how the percentage of faces classified correctly compares to: *(i)* the learning rate of the ANNA, *(ii)* the crossover probability, and *(iii)* the mutation probability. Therefore, each test presented is a permutation of these three factors. Two learning rates are used, those being: {.2, .3}; two crossover probabilities are used: {.5, .7}, and two mutation probabilities are used: {.01, .1}. Furthermore, each test is run 5 times in order to develop a statistically significant percentage of faces classified correctly. Note that these parameter values are adopted and slightly modified from previous research in both the ANNA and GA areas.

By varying the parameters in this way, testing illuminates which learning rate engenders a higher percentage of clauses satisfied, as well as which set of parameters evolves the neural topologies most effectively for this problem. For a given EANN, the neural network is trained for 50 epochs and subsequently tested; then the GA scores, breeds, and mutates the individuals. Note that both the number of epochs that each neural network is trained for and the number of generations the GA uses remains constant throughout for every cycle of the EANNA. Specifically, every EANN will be trained using 50 epochs and the number of generations will be 5.

Finally, in addition to reporting the results from the EANNA, a fully connected ANNA without topological evolution is compared side-by-side in order to develop

an understanding on the effects of the GA with respect to the accuracy of the EANNA. For this fully connected ANNA, the learning rate parameter varies as the EANNA; however, the number of epochs are altered. Namely, four epoch numbers are used: {25, 50, 75, 100}.

7. Results

From the data obtained through the experimental process, we compare the results obtained from different combinations of parameters for the neural network as well as the evolutionary algorithm. The parameter variations include:

1. Two possible ANNA learning rates (.2, .3)
2. Two possible GA crossover probabilities (.5, .7)
3. Two possible GA mutation probabilities (.01, .1)

In particular, we compared the performance in terms of classification success and runtime between the unevolved, fully connected ANNA and the EANNA. Furthermore, we analyzed the development of the ANNA and the EANNA throughout training and evolution.

An unevolved, fully connected ANNA was tested using varied learning rates and number of epochs (25, 50, 75, and 100). In particular, Figures 7.a) and 7.c) demonstrate the impact of both learning rate and number of epochs on testing and training results, respectively. Figure 7.c) communicates that a higher learning rate of .3, as opposed to .2, led to significant gains in performance throughout the training process, especially for training with fewer epochs. Similarly, Figure 7.c) indicates that an increase in the number of epochs also enhanced the percent of successful image classifications. Consequently, a higher learning rate paired with more epochs resulted in better training for the neural net.

As mentioned in the Experimental Methodology, a full cycle of the EANNA included five generations of evolution, each trained for a constant 50 epochs followed by testing. Each permutation of the tested parameters was repeated for five cycles. Figure 8 compares the performance of EANNA for successful image classification and runtime, with emphasis on variations in the learn rate, crossover probability and mutation probability, during testing. For example, Figure 8.a) shows that, generally, each unique EANNA experienced an increase in successful image classification percentage from the first generation to the last. Notably, the learn rate significantly contributed to the success of the EANNA during training. For example, an EANNA with a higher learn rate generally ended with a more successful image classification percentage.

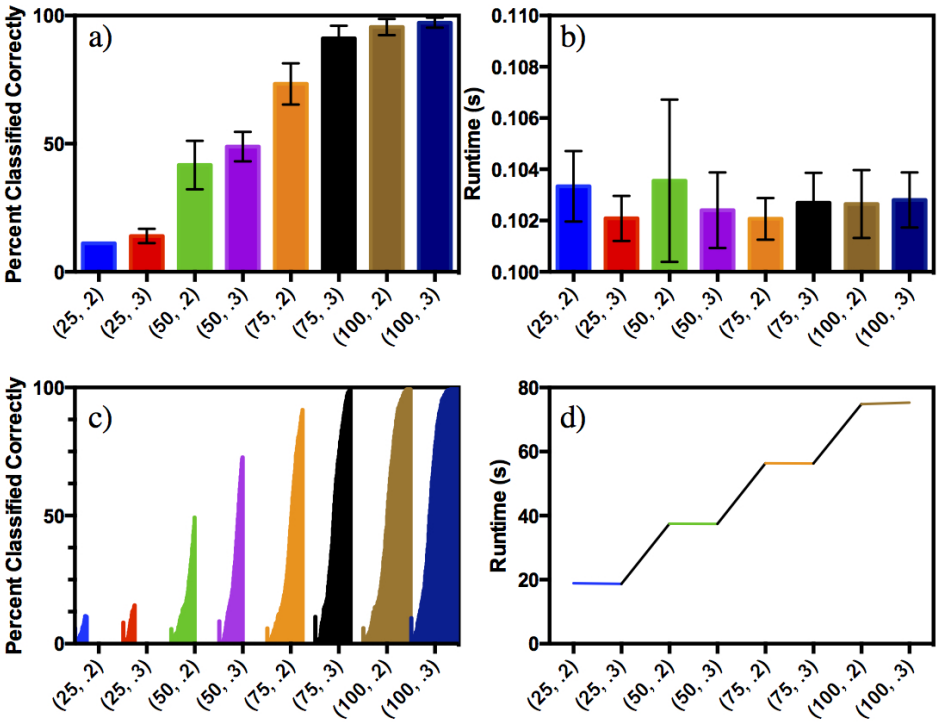

Figure 7: A comparison of all tests run on an unevolved ANNA. a) A comparison of the testing results of the ANNA. b) A comparison of runtimes of the ANNA testing. c) A comparison of the training results of the ANN. d) A comparison of runtimes of the ANNA training.

On average, the tests of a 50 epoch ANNA resulted in approximately 50% correct classification. In contrast, the EANNA consistently classified over 50% of the images correctly during testing, no matter the parameters used. Consequently, the EANNA outperforms the ANNA at 50 epochs of training in terms of percentage of successful image classifications.

Despite the clear advantage of EANNA over ANNA in image classification, Figure 8.b) demonstrates the sacrifice in runtime for the EANNA compared to the ANNA. For example, all testing of the ANNA, regardless of number of epochs, ran in under 0.108 seconds while almost all testing of the EANNA took between 0.1 and 0.3 seconds for 50 epochs. Thus, while the EANNA did outperform the ANNA for image classification, the ANNA offers faster runtime for higher epochs.

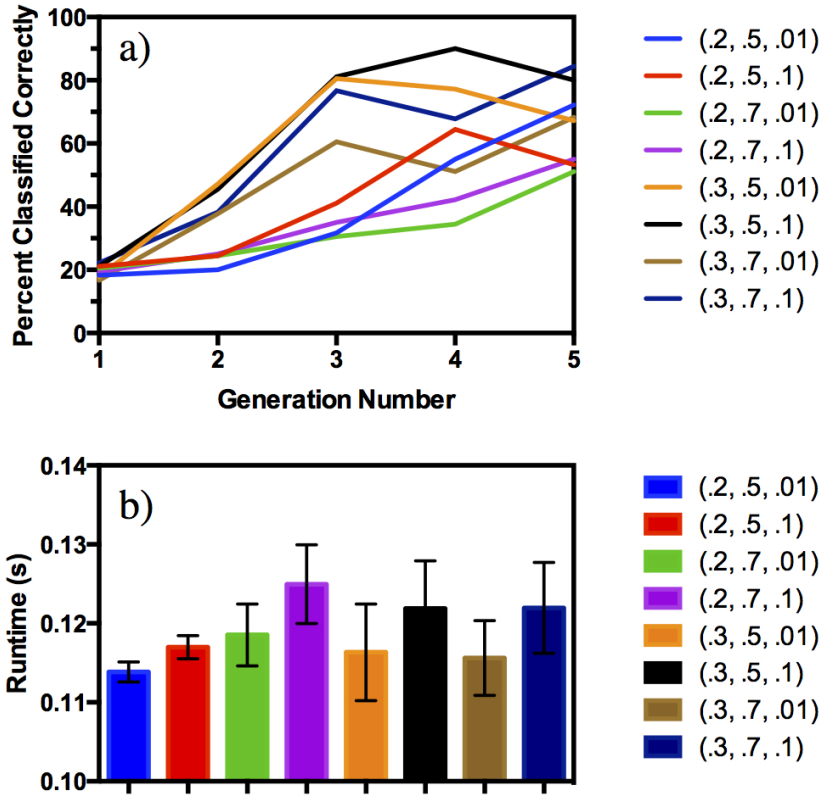

Figure 8: A comparison of the testing results of the EANNA. a) The percent of faces correctly classified during testing of the algorithm. b) the runtime comparison over all permutations of testing.

Figure 9 demonstrates the impact of varied parameters on training of the EANNA . Within each generation, the performance increases over the 50 epochs. A direct comparison of the difference in the percentage of correctly classified images between the first generation and fifth generation for all parameter permutations clearly shows that the GA dramatically increases the efficiency of the ANNA structure. The training process for each EANNA mirrored the testing in that the higher learn rate directly results in better image classification. Additionally, the crossover probability of .5 and mutation probability of .1 contributed to better image classification.

Although some permutations decrease in accuracy in the later generations, all correctly identify faces above 50% accuracy. This dramatic variation in later generational percentages stems from both *(i)* the small number of individuals in the populations and *(ii)* the limited number of generations used to evolve the network.

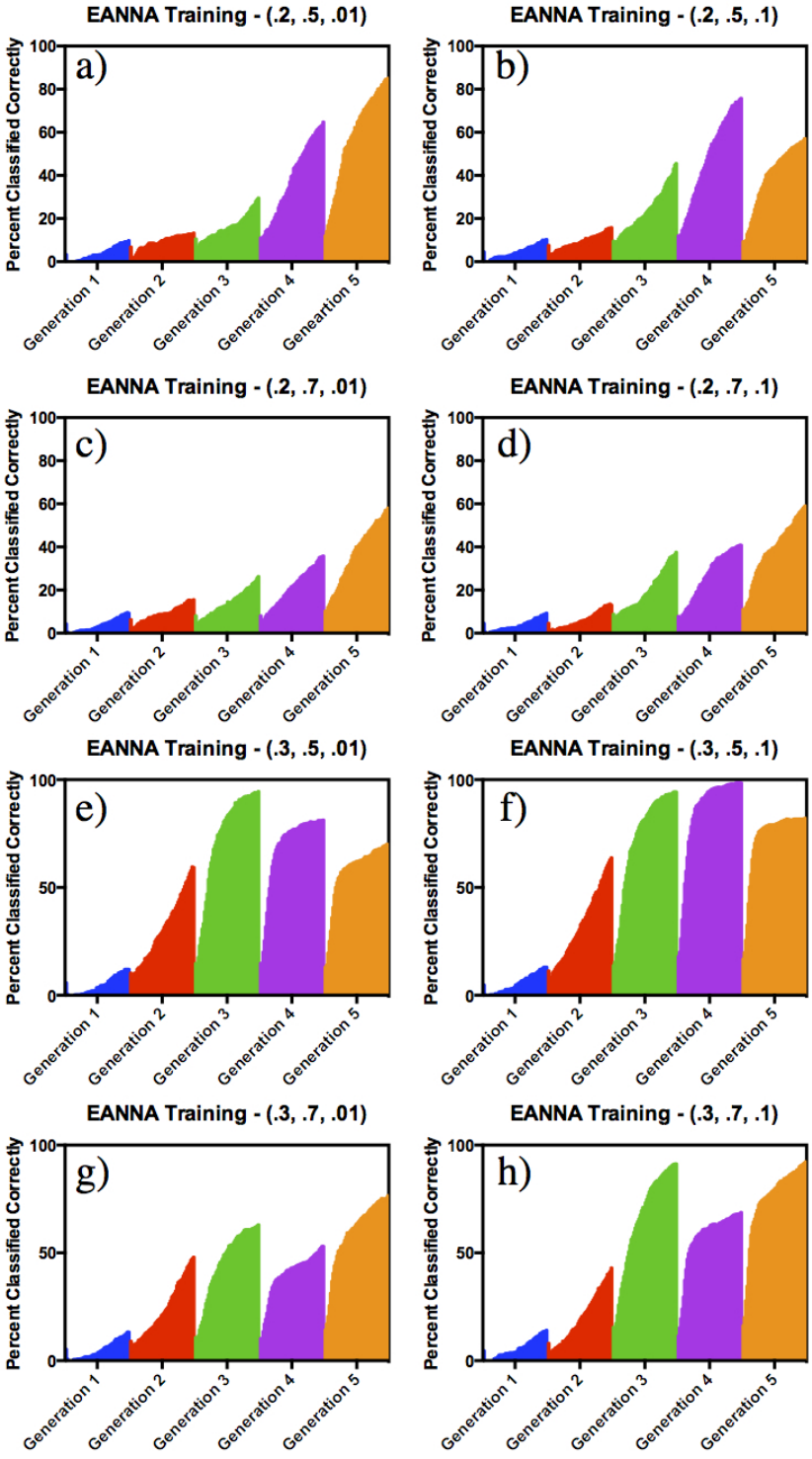

Figure 9: A comparison of all permutations of the EANNA during the training. Note that figures a) through h) represent different permutations of the parameters involved.

8. Conclusions

From our results, we have several takeaways:

1. When trained for 50 epochs and evolved through 5 generations, an EANN outperforms an unevolved, fully connected ANN.
2. An ANNA is more efficient in runtime than an EANNA but results in less successful image classification.
3. After five generations, a learning rate of .3, crossover probability of .7, and mutation probability of .1 performed the best for the EANNA.
4. A learning rate of .3, crossover probability of .5, and mutation probability of .1 performed the best over all generations for the EANNA.

The EANNA outperforms the ANNA because it evolves better connections between the input nodes and the hidden nodes. On average, the EANNA will create fewer connections per input node to the hidden node layer. As opposed to the fully connected neural net, where each input node is bound to each hidden node, the EANNA will evolve away superfluous connections, resulting in more accurate image classifications.

We found that a learning rate of .3 resulted in better classifications than a .2 learning rate. The .3 learning rate introduces change more rapidly in the edge weights, resulting in quicker learning in terms of penalties for non-matching output and rewards for matching output.

Further tests could explore the effects of the weight scaling, population size, selection method, breeding type, connection probability, number of epochs, and number of generations. These parameters were held constant throughout the tests, but each contribute to the overall performance of the EANNA. Looking forward, it would be especially beneficial to test the algorithm with greater population sizes. As a result of the small population size used and the limited number of generations, several training sets demonstrate that - purely as a result of probability - breeding and mutation can decrement the percentage of correctly satisfied images. In our tests, we only examined populations sizes of 4 and 8.

Another variation in the parameter space to consider is the implementation of different connection probabilities when initializing the topology between the input nodes and the hidden nodes. In our tests shown above, the connection probability was held constant at 10%. Ultimately, a wider exploration of the parameter space inherent to EANNA could lead to a higher percentage of images correctly classified.

9. References

[1] Mitchell, Toni. *The UCI KDD Archive*. Carnegie Mellon University. June 1995

[2] Wiegand, Stefan, Christian Igel, and Uwe Handmann. *Evolutionary Optimization of Neural Networks for Face Detection.* Reproduction 1.3 (2004): 2.

[3] Wiegand, Stefan, Christian Igel, and Uwe Handmann. *Evolutionary Multi-objective Optimisation of Neural Networks for Face Detection.* International Journal of Computational Intelligence and Applications 4.03 (2004): 237-253.

[4] Teller, Astro, and Manuela Veloso. *Algorithm Evolution for Face Recognition: what makes a picture difficult.* Evolutionary Computation, 1995., IEEE International Conference on. Vol. 2. IEEE, 1995.

[5] R. Caruana, S. Lawrence, and C. L. Giles. *Overfitting in Neural Networks: Backpropagation, Conjugate Gradient, and Early Stopping*. In Advances in Neural Information Processing Systems, volume 13, pages 402–408, Denver, Colorado, 2001. MIT Press.

[6] K. Deb. *Multi-Objective Optimization using Evolutionary Algorithms*. John Wiley & Sons, Chichester, UK, 2001.

[7] U. Handmann, T. Kalinke, C. Tzomakas, M. Werner, and W. von See- len. *An image processing system for driver assistance. Image and Vision Computing*, 18(5):367–376, 2000.

[8] C. Igel and M. Kreutz. *Operator adaptation in evolutionary computation and its application to structure optimization of neural networks.* Neurocomputing, 55(1–2):347–361, 2003.

[9] L. Prechelt. *Early stopping – but when?* In G. B. Orr and K.-R. Mu ller, editors, Neural Networks: Tricks of the Trade, volume 1524 of LNCS, chapter 2, pages 57–69. Springer-Verlag, 1999.

[10] M. Riedmiller. *Advanced supervised learning in multi-layer perceptrons – From backpropagation to adaptive learning algorithms.* Computer Standards and Interfaces, 16(5):265–278, 1994.

[11] H. A. Rowley, S. Baluja, and T. Kanade. *Neural network-based face detection*. IEEE Transactions on Pattern Analysis and Machine Intelligence, 20(1):23–38, 1998.

[12] X. Yao. *Evolving artificial neural networks*. Proceedings of the IEEE, 87(9):1423–1447, 1999.

[13] Gomez, Faustino J., and Risto Miikkulainen. *Active guidance for a finless rocket using neuroevolution*. Genetic and Evolutionary Computation—GECCO 2003. Springer Berlin Heidelberg, 2003.

[14] Moriarty, David E., and Risto Miikkulainen. *Hierarchical evolution of neural networks*. Evolutionary Computation Proceedings, 1998. IEEE World Congress on Computational Intelligence.