

# Genetically Evolving Optimal Neural Networks

Chad A. Williams  
cwilli43@students.depaul.edu  
November 20th, 2005

## Abstract

*One of the greatest challenges of neural networks is determining an efficient network configuration that also is highly accurate. Due to the number of possible configurations and the significant difference in similar networks, the search for an optimal configuration is not well suited to traditional search techniques. Genetic algorithms seem a natural fit for this type of complex search. This paper examines genetic algorithms research that has focused on addressing these challenges. Specific focus is paid to techniques that have been used to encode the problem space in order to optimize neural network configurations. A summary of the results as well as areas for future research in this field are also discussed.*

## 1 Introduction

One of the main challenges of successfully deploying neural networks is determining the network configuration that is best suited for the particular problem at hand. Often the number of hidden layers and nodes is selected using rule of thumb heuristics and any revisions are made using trial and error. This approach results in empirically supported educated guesses at best and provide little confidence that a close to optimal configuration has been selected. The second problem with this approach is that as problem complexity increases, the number of potential configurations increases rapidly as well, decreasing the likelihood of selecting the best configuration in a timely manner. This paper examines the use of genetic algorithms to systematically select a neural network configuration suited for an arbitrary problem space in an automated fashion.

The process of selecting a “good” neural network configuration can be thought of as a search problem with the hypothesis space being the set of all possible network configurations with the specified number of input and output nodes. This problem presents a number challenges: first, the number of possible configurations is infinite; second, the decision surface is plagued with local minima resulting in significant problems for traditional search techniques such as hill climbing or gradient descent. However, this combination of attributes makes the search well suited for a genetic approach since the randomization introduced by cross-over and mutation are more likely to find a global minima. This realization has led to considerable research in using genetic algorithms (**GAs**) to configure numerous aspects of neural networks. This paper will survey different genetic approaches and encodings that have been used to select optimal neural network configurations, as well as a discussion of these results. Specifically this paper will focus on how GAs have been used to learn network weights and network architecture.

## 2 Encoding neural networks

In order to apply GAs to the neural network space, the configuration must be encoded in a fashion that lends itself to common genetic operations such as cross-over and mutation in a meaningful way. These encodings depend largely on the goal of the learning process which has included learning network weights, network architecture, and other network aspects.

### 2.1 Network weights

Several researchers have focused on using GAs to learn network weights assuming a static network architecture. The motivation for this approach has been two fold. First, researchers have seen GAs as an alternate method to search the decision surface represented by a neural network that is less sensitive to local minima and the selection of initial weights. Second, it has been hypothesized for complex non-differentiable, non-continuous decision surfaces, GA may be better suited for the search than back propagation (**BP**) techniques. To approach this problem in a genetic fashion, researchers have looked at several ways of encoding the network weights. The two main approaches have varied based on whether the network should be restricted in order to reduce the number space to integers or use traditional networks and encode real numbers in binary strings. The integer based approach has the advantage of much smaller encoding genes, but tends to have many more nodes to train due to the more constrained network weights requiring a larger network to represent the same decision surface. Despite these rather fundamental differences, the two different approaches have been shown to produce similar results for both training speed and accuracy [1]. For nearly all approaches the fitness function is based on the error between the training set and the feed-forward network output using the encoded weights. The termination criteria for this approach are similar to BP approach, namely if a maximum number of epochs have been reached or an error/variance threshold has been reached.

For small networks with few layers, GA approaches using the encodings described above have been shown to have similar performance to BP techniques. For larger networks with decision surfaces commonly seen in real world problems BP has been shown to converge much faster raising doubts about whether GA makes sense for this type of problem [2]. For more complex surfaces that have been primarily created as theoretical examples to mimic the XOR decision surface, GA has been shown to converge marginally faster than basic gradient descent BP techniques. However when compared to more state of the art techniques such as QuickProp, the differences were not significant. It is worth noting that although GAs were less efficient for large feed forward networks, GAs have been shown to be quite competitive for large recurrent networks [1]. Regardless of convergence speed, GA solutions have been shown to produce at least as good generalization for unseen data as BP techniques.

### 2.2 Network architecture

A second, arguably more interesting, area of research is using GAs to optimize the neural network architecture. For these approaches, the number of nodes and the connectivity of those nodes have been the target of the GA's search. For this problem researchers have tried a number of different encoding strategies. These encodings fall into two main categories, "direct" encodings that encapsulate all information required to rebuild an identical network

and “indirect” encodings which focus instead on patterns for building the network.

The direct encoding approach attempts to learn both the weights and architecture in parallel and thus encode all network nodes, connections between the nodes, and the weights for all connections. With these schemes the goal is to have the GA select the optimal overall network configuration. This approach has been very successful at producing novel efficient networks for small problems. As would be expected, the addition of the architecture information in conjunction with the weight information leads to a longer convergence time than the weight information alone. As a result, due to the large amount of information that must be encoded, this approach does not scale well to larger problem spaces.

The second school of thought, indirect encoding, focuses the GA search strictly on the extremely large non-differential hypothesis space of the architecture itself and uses BP for the weight training of the encoded architecture [3]. With this approach, the genetic encoding contains only the information to create the number of nodes and depending on the specific approach may contain connection information directly or may instead contain general rules of connectivity between layers. Supporters have argued that this approach is more biologically sound as well since complex organisms do not have all of their knowledge encoded in their genes but rather instructions on how to form the general brain structure that is appropriate [1]. By eliminating the weight training from the GA search, the encoding length is significantly reduced and the resulting combination of GA and BP algorithms converges significantly faster than using strictly the GA approach. Regardless of encoding approach, network architectures that have been derived through evolutionary techniques have been shown to far exceed hand fabrication methods of development in both development time and performance [3].

For approaches that address network architecture, one termination criteria that has been successful is based on fitness variance,  $V(g)$ , within the population falling below a threshold such as the equation:

$$V(g) = \frac{1}{M} \sum_{i=1}^M (F_i(g) - \bar{F}(g))^2 \leq V_{\min}$$

from Zhang and Mühlenbein [4]. Where  $M$  is the size of the population,  $F(g)$  is the fitness of a particular encoding and  $\bar{F}(g)$  is the average fitness of the generation. Another difference between the weight focused method and the architecture focused method has to do with the calculation of their fitness function. For algorithms where GA is used to train the weights it is common to use a single large training set as training and testing are essentially the same operation. For the techniques that combine GA architecture learning with BP weight learning, it has been shown to be superior to perform network training on the training set, while basing the fitness function on the error the network produces compared to a separate test set [5]. This approach has been shown to be effective at selecting architectures that reduce over fitting for unseen data. It is also common for architecture learning approaches to use fitness functions that combine accuracy measures with other various measures. One combination that seems to offer excellent promise is based on Occam’s razor and is introduced in [4]. This fitness function combines accuracy with a complexity measure of the architecture based on the magnitude of the network weights, such that  $C(W|A) = \sum_{k=1}^K w_k^2$ . The net result is a search that biases towards the simpler solution being the more likely solution. By using

this function, Zhang and Mühlenbein were able to produce networks that were significantly smaller and trained faster while performing at least as well on unseen data compared to fitness based on training error alone. As Table 1 depicts, the networks learned using a

Table 1: Comparison of fitness functions based on error vs. error and complexity

method	layers	units	weights	training	generalization	learning time
$F = E$	9.7	153.0	1026.0	95.2%	92.4%	20294.7
$F = E + C$	3.7	19.7	123.1	92.9%	92.7%	5607.2

fitness function based on both error and complexity( $F = E + C$ ), had a factor of 10 fewer nodes and converged in about a quarter of the time despite producing similar generalization accuracy [4].

The primary approach for both the direct and indirect encoding techniques has been to use variable length descriptions as the number of nodes in the optimal network configuration are unknown ahead of time. As a result, these encodings tend to be more complex than those used for the network weight problem alone as more intelligence must be used to ensure valid hypothesis are produced from cross-over operations. To address this complexity two main approaches have been used. One approach has been to use variable length binary strings and build the logic into the cross-over logic. A second approach has been to extend to encoding to be more of a genetic programming solution constructing, LISP S-expression, grammar trees [6]. Using this technique, cross-over is performed by swapping sub-trees as seen in Figure 1.

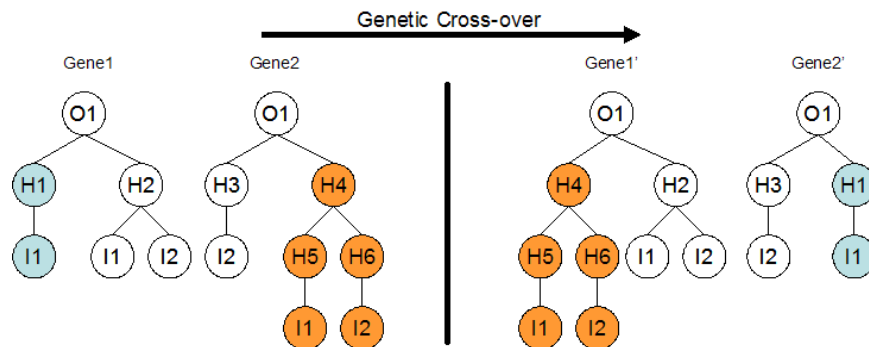


Figure 1: Cross-over using genetic programming approach for a neural architecture

A second approach that has also been used, assumes that the optimal network is no larger than a pre-specified size and thus is able to use a fixed-length encoding. Although the individual generations with this approach run faster due to the simplification, the approach suffers from having to start with an assumption about the maximum network size and as a result is not guaranteed to find the optimal network architecture. This is particularly problematic as the larger the assumed maximum network size, the slower the convergence. The variable length encodings, on the other hand, are not limited by this assumption and their convergence rate, although slower per generation, is proportional to the solution size. As a result variable length encodings are more frequently used due to both the theoretic benefit and their additional flexibility.

### 3 Conclusion

There is a significant amount of research currently going on in this area particularly related to optimizing other aspects of network configuration. Other areas gaining momentum are related to evolving the transfer functions of the individual nodes as well as the learning rules for the particular problem space [3]. For transfer functions, the most common technique has been to use a single transfer function, such as sigmoid or gaussian, throughout the network or less often have a single transfer function per layer which may vary across layers. Initial results that have focused on optimizing the transfer function at the node level have shown that for some datasets this results in a higher generalization than using any single transfer function alone. The second area of research, learning rules, is related to optimizing the training of the network given a particular architecture. This task involves selecting the individual type of BP algorithm, a GA or some other method as well as the learning parameters for that algorithm that produce the fastest convergence with the best generalization. Although some have argued that using GAs to find the optimal BP learning parameters is just swapping requiring designers to be familiar with GA parameters rather than BP parameters, proponents of this have argued this could help develop additional theoretic insight into BP parameter selection that could lead to better heuristics for BP parameter selection long term.

In addition to these new areas, there is still a significant amount of work focused on additional encodings and fitness functions that may help further efficient neural network architecture selection. In particular encoding recurrent networks has been more challenging due to the genetic programming techniques used to simplify crossover, as seen in Figure 1, not easily translating to cyclical relations. Another aspect of this which is just being investigated is developing a theoretic basis for comparing the capabilities of different encoding schemes and fitness functions.

As can be seen from this review of the research in this area, GA approaches offer significant potential for addressing some of the more complex problems associated with neural networks. Architecture evolution in particular seems particularly promising in addressing some of the key challenges that have often scared industry away from more widespread use of neural networks for large scale real world problems. It will be interesting to see whether long term GAs are used widespread for neural architecture selection or if the insights gained by this research will instead be used to develop better heuristics that can be employed by more traditional search techniques.

### References

- [1] X. Yao, A review of evolutionary artificial neural networks, *International Journal of Intelligent Systems* Vol. 8 (1992) 539–567.  
URL [citeseer.ist.psu.edu/yao93review.html](http://citeseer.ist.psu.edu/yao93review.html)
- [2] P. Koehn, Genetic encoding strategies for neural networks, in: *Proceedings of Information Processing and Management of Uncertainty in Knowledge-Based Systems, Vol. II*, Granada, Spain, 1996, pp. 947–950.  
URL [citeseer.ist.psu.edu/koehn96genetic.html](http://citeseer.ist.psu.edu/koehn96genetic.html)

- [3] D. Curran, C. O’Riordan, Applying evolutionary computation to designing neural networks: A study of the state of the art, Tech. rep., National University of Ireland, Galway (2002).  
URL [citeseer.ist.psu.edu/curran02applying.html](http://citeseer.ist.psu.edu/curran02applying.html)
- [4] B.-T. Zhang, H. Mühlenbein, Evolving optimal neural networks using genetic algorithms with Occam’s razor, *Complex Systems* Vol. 7 (1993) 199–220.  
URL [citeseer.ist.psu.edu/zhang93evolving.html](http://citeseer.ist.psu.edu/zhang93evolving.html)
- [5] P. Koehn, Combining genetic algorithms and neural networks: The encoding problem, Master’s thesis, University of Erlangen and The University of Tennessee, Knoxville (1994).  
URL [citeseer.ist.psu.edu/article/koehn94combining.html](http://citeseer.ist.psu.edu/article/koehn94combining.html)
- [6] J. R. Koza, J. P. Rice, Genetic generation of both the weights and architecture for a neural network, in: *International Joint Conference on Neural Networks, IJCNN-91, Vol. II*, IEEE Computer Society Press, Washington State Convention and Trade Center, Seattle, WA, USA, 1991, pp. 397–404.  
URL [citeseer.ist.psu.edu/koza91genetic.html](http://citeseer.ist.psu.edu/koza91genetic.html)