A Study of Neural Network Evolution for Vision Under Varying Input Formats

Jaime J. Dávila
Hampshire College
School of Cognitive Science
Amherst MA 01002

Abstract - The GENDALC (Genetic Evolution of Neuron Distribution and Layer Connection) system successfully develops neural networks for natural language tasks. In this paper I present results on the evolution of topologies for processing of visual input under different formats, and a discussion of which domains GENDALC appears best suited for.

I. INTRODUCTION

The GENDALC (Genetic Evolution of Neuron Distribution and Layer Connection) system has been previously used to evolve topologies for natural language processing tasks [1]. Neural networks evolved in those experiments performed better than other commonly used topologies such as fully connected networks, Simple Recurrent Networks [2], Frasconi-Gori-Soda networks (FGS) [3], and Narendra-Parthasarathy networks [4].

In order to determine the general ability of the GENDALC system to evolve solutions to different problems, it was presented with a problem in a different domain. In this case, the evolved networks were required to imitate the behavior of infants described by Spelke, Breinlinger, Macomber and Jacobson [5].

II. DESCRIPTION OF THE GENDALC SYSTEM

The GENDALC system determines NN topologies by distributing 75 hidden nodes among up to 30 hidden layers, and then determining how these hidden layers are connected to each other.

Each topology in the GENDALC system is configured by a genome in a genetic algorithm population. The exact number of hidden layers is determined by the first gene of the corresponding genome. This position stores a floating point number, with a value between 0 and 1. To determine how many hidden layer a network has, the value of this gene is multiplied by 30, and rounded to the next highest integer. If the result of this rounding up is 31, the network uses 30 hidden layers.

The number of hidden nodes in each of these hidden layers is also determined by the network's corresponding genome. The genome has 30 genes used to code the "relative worth" of each of the possible hidden layers. Once the number of hidden layers is determined to be N using the process described above, the N layers with the highest relative worth are identified. The 75 available hidden nodes are distributed among each of these N hidden layers according to each layer's worth relative to the sum of all N worth values.

Take, for example, the partially illustrated genome in figure 1, here showing genes 1 through 31. The first gene, with a value of .23, determines that this network will have 7 hidden layers. The relative worth for all 30 layers are represented here by genes 2 through 31. Among these, the seven with highest worth value are layers 18, 14, 1, 21, 7, 20, and 27 (with relative worth values of .99, .96, .91, .90, .84, .84, and .75). The addition of these relative worth gives 5.27, which means that each of these seven layers will have 75*X/5.27 nodes, where X is each layer's relative worth. Therefore, for this example layers 18, 14, 1, 21, 7, and 20 will have 13, 13, 11, 10, 10, 10, and 8 nodes, respectively.

Figure 1: genes 1-31 for a sample genome.

The connections between these seven layers are also determined by the network's genome. For each of the thirty possible layers, there is a gene that indicates where the layer takes its input from. Each of these genes stores a floating point value between 0 and 1. To determine where each hidden layer takes its input from, its "takes-its-input from" gene value is multiplied by N+2 (where N is the number of hidden layers this network will have, as determined with the procedure outlined previously), and rounded to the nearest integer. The resulting number points to which layer this one takes its input from. We multiply by N+2 to allow for hidden layers taking its input from any of the N hidden layers, as well as either the input or the output layer. A value of 0 means the layer takes its input from the input layer. A value of N+2 means the layer takes its input form the output layer. For values between 2 and N+1, the layer would take its input from the layer with the $(N-1)^{th}$ highest relative worth.

Where each layer sends its output to is determined in a similar way, using positions 62-91 of the genotype. Each of these genes stores a floating point value between 0 and 1. To determine where each layer sends its output to, its "sends-output-to" gene value is multiplied by N+1 and rounded to the nearest integer. The resulting number points to which other layer this one will send its output

to. We multiply by N+1 to allow for hidden layers sending its output to any of the N hidden layers, as well as to the output layer. A value of N+1 means the layer sends its output to the output layer. For values between 1 and N, the layer sends its output to the layer with the N^{th} highest relative worth. No layer sends its output back to the input layer.

III. DESCRIPTION OF THE TASK FOR THE NN

The task for the NN is based on experiments done by Spelke, Breinlinger, Macomber and Jacobson [5] on children. In their experiments, they habituated 2⅓ to 4 month old babies to a ball being dropped to the floor behind an occluding screen. After the ball fell to the floor, the screen was removed, revealing the ball on the floor. Once the babies were habituated to this falling ball, a second horizontal object was placed above the floor. Then both the floor and the horizontal bareer were occluded by a screen, and the ball was once again dropped from above. When the screen was removed, the ball was revealed to be either on the floor or on the horizontal bareer. The former was consistent with the habituating scenes, but inconsistent with solidity constraints. By measuring how long the infants looked at the scenes, the researchers estimated the level of surprise of the children.

The task for the NN in this experiment is to imitate the behavior described above. The input to the NN is a representation of the initial and final position of the ball and barrier it 'sees.' The output of the NN should be one if the scene sequence is surprising (i.e., the ball seems to have gone through the barrier) or zero when it is not surprising. In order to better corroborate GENDALC's ability to evolve topologies for different tasks, three different input formats will be used.

A. Format A

The first format to be used represents the position if the ball and barriers with simple coordinates. Therefore, only four numbers are needed to represent a scene. Both the initial and final scene are shown at the same time, for a total of eight input nodes.

B. Format B

The second input format represents each scene as a 10 by 10 matrix into which the position of the ball and the barrier are drawn. The ball is represented by a single input set to 1 at the ball's position. The barrier is represented by a horizontal row of 10 input nodes set to 1's at the barrier's position. All other input nodes are set to 0. Once again, both the initial and final scene are presented simultaneously, which requires 200 input nodes. An example of this format is shown in figure 2. Lines starting with a '#' are ignored by the NN simulator used.

```
# Input layer
# first scene.
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 1 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
1 1 1 1 1 1 1 1 1 1
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
# second scene.
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
1 1 1 1 1 1 1 1 1 1
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 1 0 0 0 0
# Output layer
1
```

Figure 2: example of format B input.

C. Format C

The third input format represents the balls and barriers the same way as format B, but only one scene is presented at a time. The output after the first scene is always of no surprise ('0'). The output of the NN should be one if the scene sequence is surprising (i.e., the ball seems to have gone through the barrier) or zero when it is not surprising. Figure 3 shows an example of this input format.

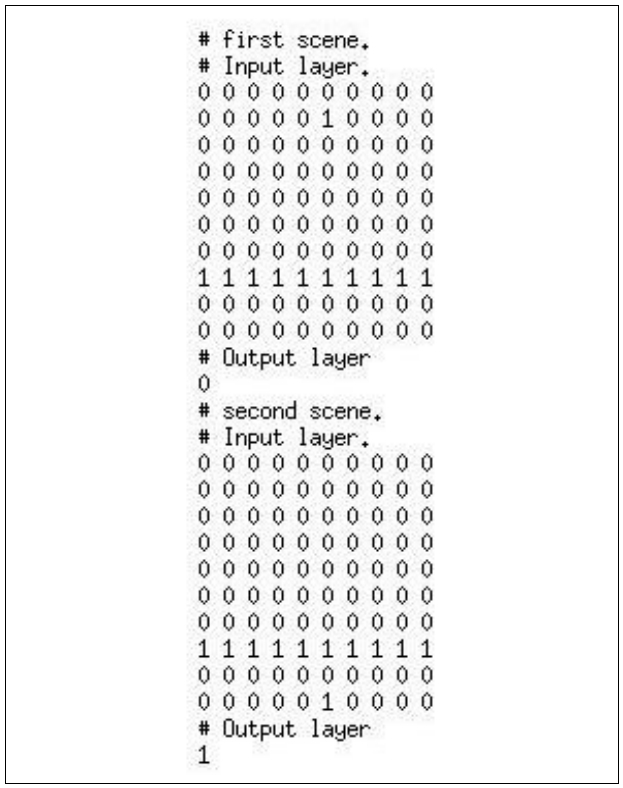
IV. EXPERIMENT PARAMETERS

The NN described in section II are evolved using a basic genetic algorithm (GA). Each network is trained for 500 epochs with 400 scenes of the type described in section III. Then its fitness is determined by evaluating its performance when processing 2000 scenes. The sum of square errors for the single output is computed and assigned as the network's fitness.

The NN aspect of the software runs is performed using the batchman language in the SNNS neural network simulation package. The genetic algorithm aspect is performed using my personal modifications to the GENITOR system developed by Darell Whitley [6]. GET REFERENCE

During each generation of the GA, all networks are selected a single time for mating. During mating, two NN are combined by selecting two crossover points and switching the part of their corresponding genomes that lies between them.

With a random probability of 4% a single gene on a network's genome gets altered, getting assigned a new random value between 0 and 1.



```
# first scene.
# Input layer.
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 1 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
1 1 1 1 1 1 1 1 1 1
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
# Output layer
0
# second scene.
# Input layer.
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
1 1 1 1 1 1 1 1 1 1
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 1 0 0 0 0
# Output layer
1
```

Figure 3: example of format C input.

All networks are trained with the batch backpropagation through time algorithm, with a learning parameter of 0.2. All nodes use a squashing logistic function defined as $f(x) = 1/(1+e^{-x})$, where x is the total weighted input to the node.

After evolving NN configurations, and in order to verify consistent performance, networks are validated by performing bootstrap validation [6]. During this process, a set of 2000 were randomly generated, of which 400 were used for training. Trained networks are then tested with the other 1600 patterns. This process was repeated 42 times. Numbers reported as networks performance are the average for all cross validation runs, except where noted.

V. RESULTS

VI. COMMENTS ON THE GENDALC SYSTEM

VII. FUTURE RESEARCH

VII. CONCLUSIONS

REFERENCES

[1] J. Davila, "Genetic Optimization of NN Topologies for the Task of Natural Language Processing," *Proceedings of the International Joint Conference on Neural Networks*, Washington, D.C., 1999

[2] Elman, J. L., Distributed Representations, Simple Recurrent Networks, and Grammatical Structure. *Machine Learning, 7,* 1991, pp. 195-224.

[3] Frasconi, P., Gori, M., and Soda, G., (1992) Local feedback multilayered networks. *Neural Computation, 4(1),* 1992, pp. 120-130.

[4] Narenda, K., Parthasarathy, K. Identification and control of dynamical systems using neural networks. *IEEE Transactions on Neural Networks, 1(1)*, 1994, pp. 4-27.

[5] E. Spelke, K. Breinlinger, J. Macomber, and K. Jacobson, Origins of Knowledge, *Psychological Review, 1992, Vol. 99, No. 4*, pp. 605-632.

[6] S. Weiss, and C. Kulikowski, Computer Systems that Learn. Classification and Prediction Methods, in *Statistics, Neural Nets, Machine Learning, and Expert Systems*, 1991 Morgan Kaufmann. San Mateo, California.