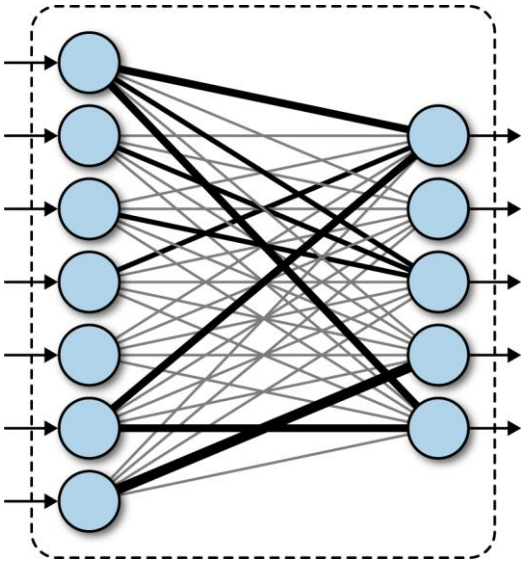
Amit Nagler & Noam Atia

Evolutionary Neural Networks

Introduction

Artificial Neural Networks are complex computing systems, that inspired by the nature of our brain. An ANN is based on a collection of connected computing units called artificial neurons, which loosely model the neurons in a biological brain. Each connection, like the synapses in a biological brain, can transmit a signal to other neurons. An artificial neuron that receives a signal then processes it and can signal neurons connected to it. The "signal" at a connection is a real number, and the output of each neuron is computed by some non-linear function of the sum of its inputs. The connections are called edges. Artificial Neural Networks were first proposed in 1944 by Warren McCullough and Walter Pitts from MIT, and their new name is Deep Learning, which is the best AI system for Machine Learning over the past 10 years.

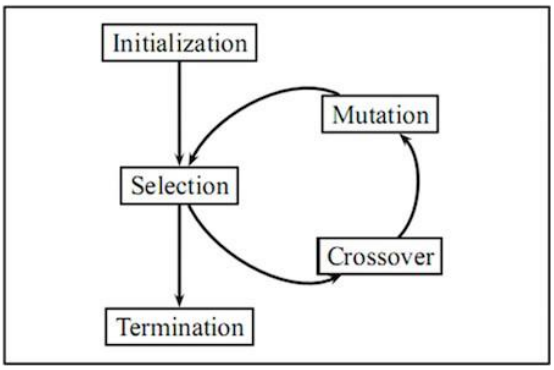


The neurons in ANN are typically organized into multiple layers. Neurons of one layer connect only to neurons of the immediately preceding and immediately following layers. The layer that receives external data is the input layer. The layer that produces the ultimate result is the output layer. In between them are zero or more hidden layers. Between two layers, multiple connection patterns are possible. They can be **fully connected**, with every neuron in one layer connecting to every neuron in the next layer.

ANN edges typically have a **weight** that adjusts as learning proceeds. Neural network models are fit using an optimization algorithm called stochastic gradient descent that incrementally changes the network weights to minimize a loss function, hopefully resulting in a set of weights for the mode that can make useful predictions. This optimization algorithm requires a starting point in the space of possible weight values from which to begin the optimization process. Weight initialization is a procedure to set the weights of a neural network to small random values that define the starting point for the optimization (learning or training) of the neural network model.

Weight initialization is an important design choice when developing deep learning neural network models. Each time, a neural network is initialized with a different set of weights, resulting in a different starting point for the optimization process, and potentially resulting in a different final set of weights with different performance characteristics.
Historically, weight initialization involved using small random numbers, although over the last decade, more specific heuristics have been developed that use information, such as the type of activation function that is being used and the number of inputs to the node.

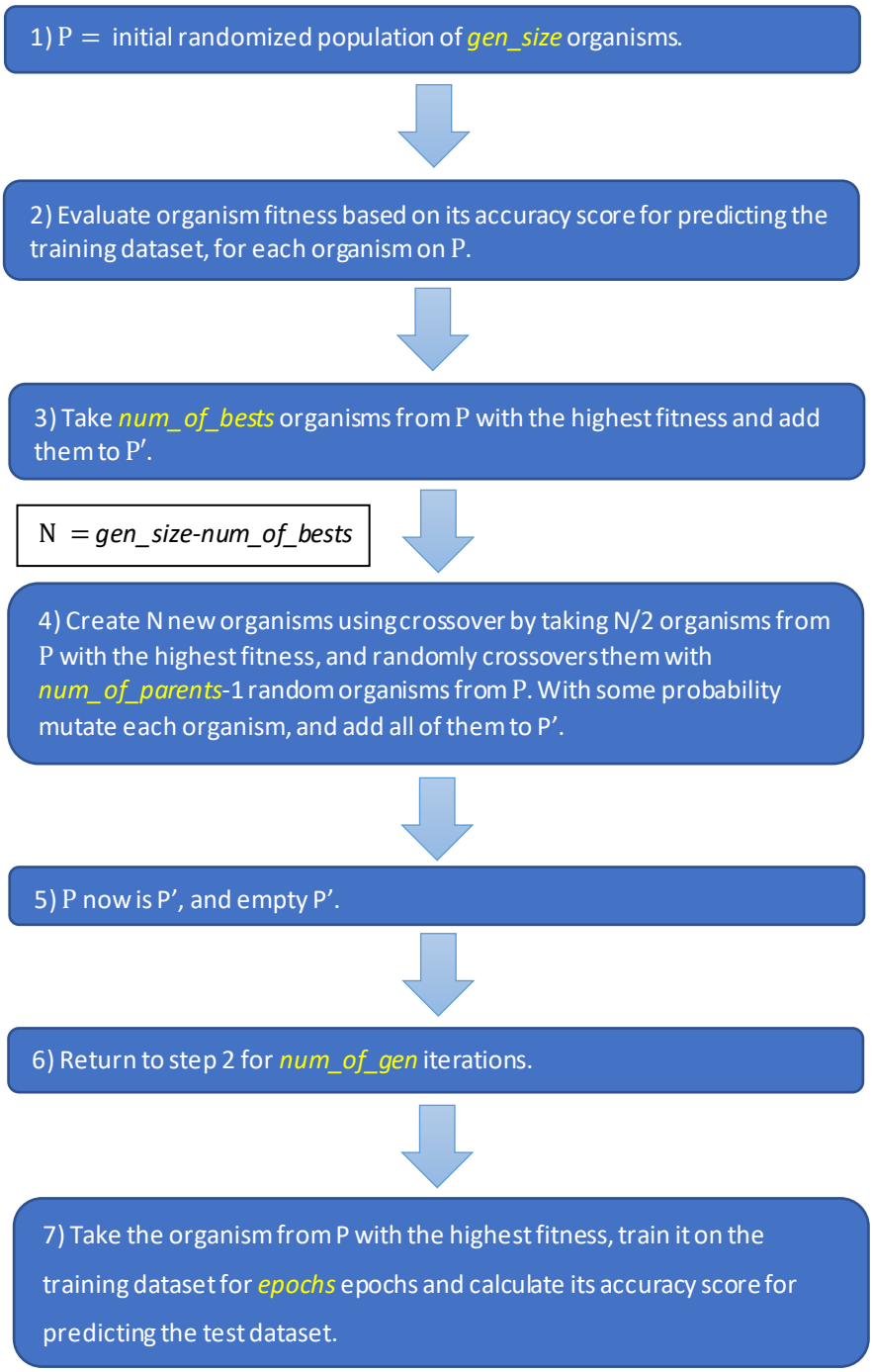
Evolutionary algorithms are a heuristic-based approach to solving problems that cannot be easily solved in polynomial time, such as classically NP-Hard problems. When used on their own, they are typically applied to combinatorial problems, however, genetic algorithms are often used in tandem with other methods, acting as a quick way to find a somewhat optimal starting place for another algorithm. The premise of an evolutionary algorithm is quite simple given that you are familiar with the process of natural selection. An EA contains four overall steps: initialization, selection, genetic operators, and termination. These steps each correspond, roughly, to a particular facet of natural selection, and provide easy ways to modularize implementations of this algorithm category.



Evolutional Neural Networks are Artificial Neural Networks that their weight initialization is done by evolutionary algorithms. These neural networks hold properties like fitness, and use a genetic algorithm to train randomly generated weights. Genetic optimization occurs prior to backpropagation which gives any type of gradient descent a better starting point.

Method

In a genetic neural network, the network is viewed as an **organism** with fields and fitness. These fields are the network weights, and they are considered genes, relative to each organism, to optimize by a genetic algorithm prior to backpropagation. This **weight initialization** (step 1-6 on the next diagram) gives gradient descent a better starting position and allows for fewer training epochs with higher model testing accuracy.



Experimental Setup

In our experiment Organism is a **Tensor Flow fully connected sequential model** with four input nodes, two hidden layers, and three output layers (to match the dataset), but this can be extended to any type of neural network.

The **crossover** between parents is a single-point crossover, when every weight matrix column will have an equal chance of getting selected as a crossover point for each parent to combine their genes and pass them to a child.

To ensure the population explores the solution space, there is a chance **mutation** will occur. The chance and power of mutation in this case is significantly higher than most other genetic algorithms as the solution space is very large. There is no particular way to mutate a matrix, in this case we randomly perform scalar multiplication on the matrix by a magnitude of 2–5.

The **dataset** we used is Iris flower multivariate dataset. Its purpose in to quantify the morphologic variation of Iris flowers of three related species. Each example comes with 4 features representing flower sepal length, sepal width, petal length, petal width.

The first goal of our experiment was to find the best **evolutionary algorithm parameters**:

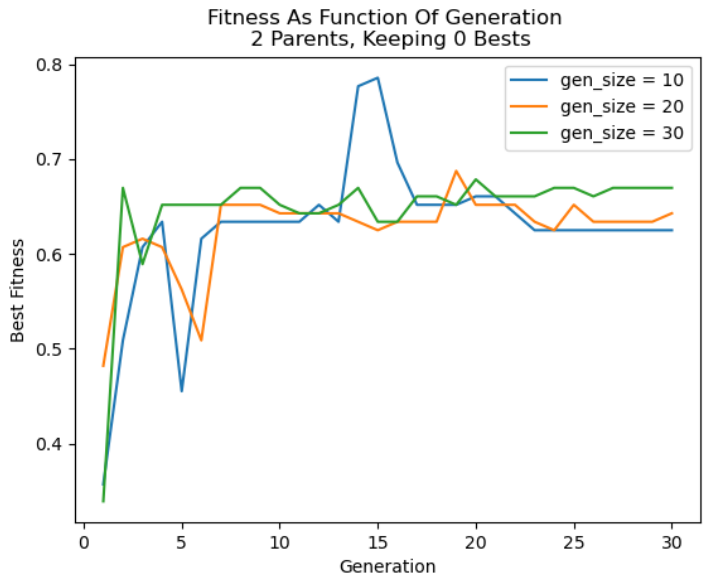
- *gen_size* ([1, 10, 20, 30]) = Number of organisms on each generation
- *num_of_gen* ([0, 10, 20, 30]) = Number of generations of the evolutionary algorithm. When 0 it means no evolution, randomized weight initialization.
- *num_of_parents* ([1, 2, 3]) = Number of organisms that used for creating next generation organisms on the crossover process. Self crossover, Monogamy crossover and Polyamory crossover.
- *num_of_bests* ([0, 2, 4]) = Number of best organisms that auto pass to the next generation.

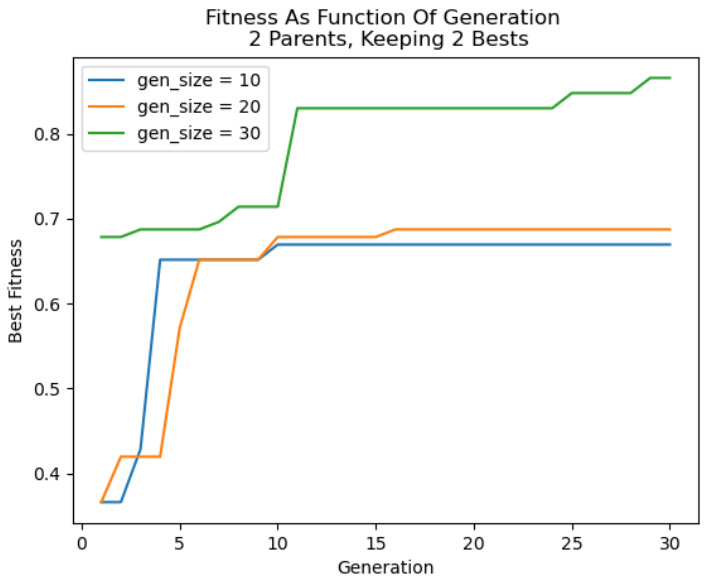
The second goal of out experiment was to compare between the **accuracy and the running time** of evolutionary neural networks (*num_of_gen*=30) and randomized weight initialized artificial neural networks (*num_of_gen*=0).

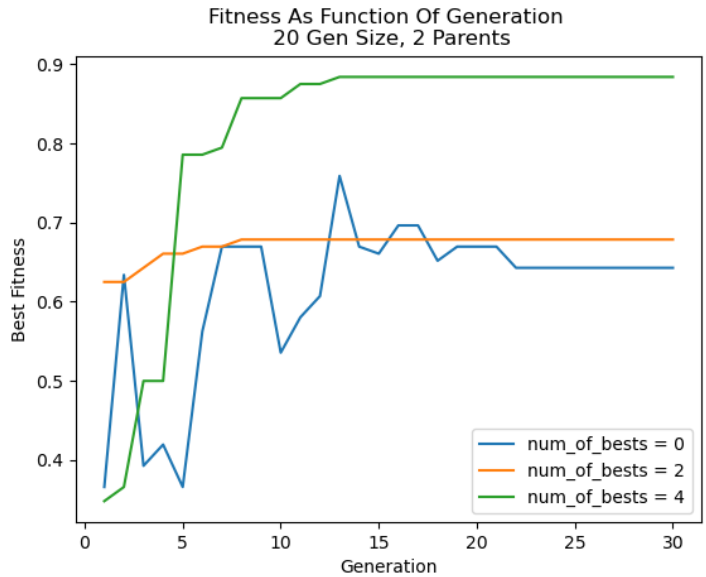
Results

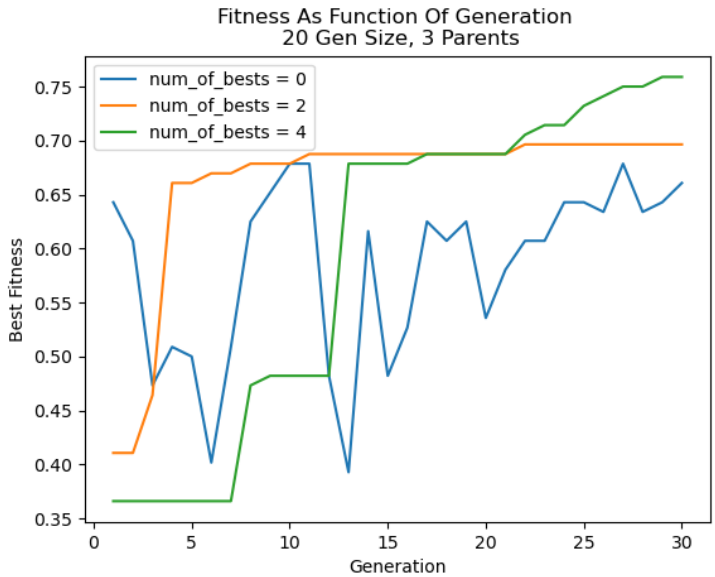
- Polyamory crossover creates organisms with better fitness then Monogamy crossover and Self crossover.
- The bigger the generation size and the number of generations are, more fitted organisms are created.
- Passing 4 organisms to the next generation creates organisms with better fitness then passing 2 or 0.

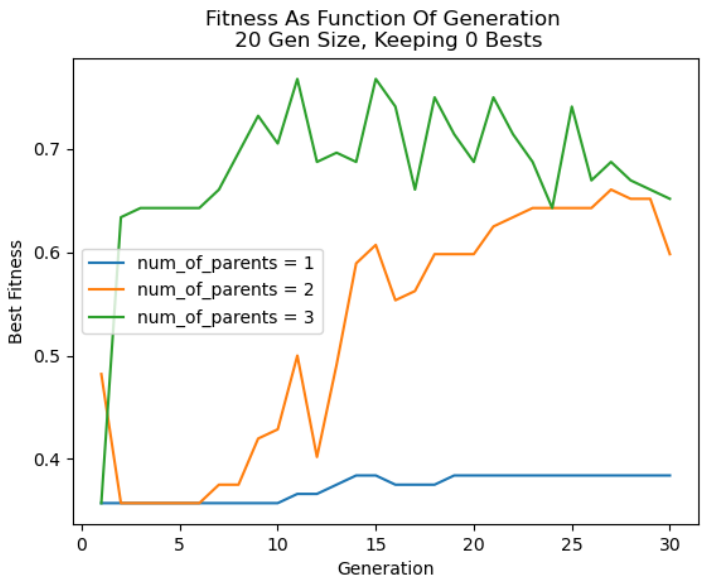
- Evolutionary neural networks are more accurate then randomized weight initialized artificial neural networks.
- The calculation time of evolutionary neural networks is much higher than the calculation time of randomized weight initialized artificial neural networks.

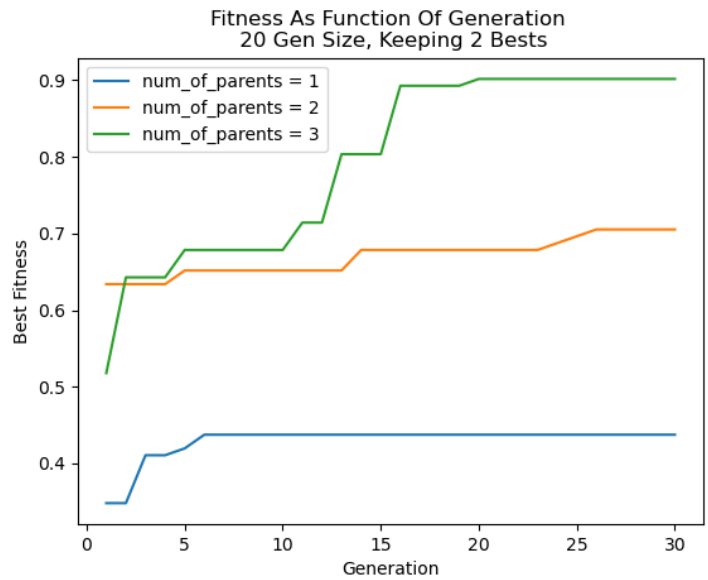

Fitness As Function Of Generation
2 Parents, Keeping 0 Bests
gen_size = 10
gen_size = 20
gen_size = 30
Best Fitness
Generation


Fitness As Function Of Generation
2 Parents, Keeping 2 Bests
gen_size = 10
gen_size = 20
gen_size = 30
Best Fitness
Generation


Fitness As Function Of Generation
20 Gen Size, 2 Parents
num_of_bests = 0
num_of_bests = 2
num_of_bests = 4
Best Fitness
Generation

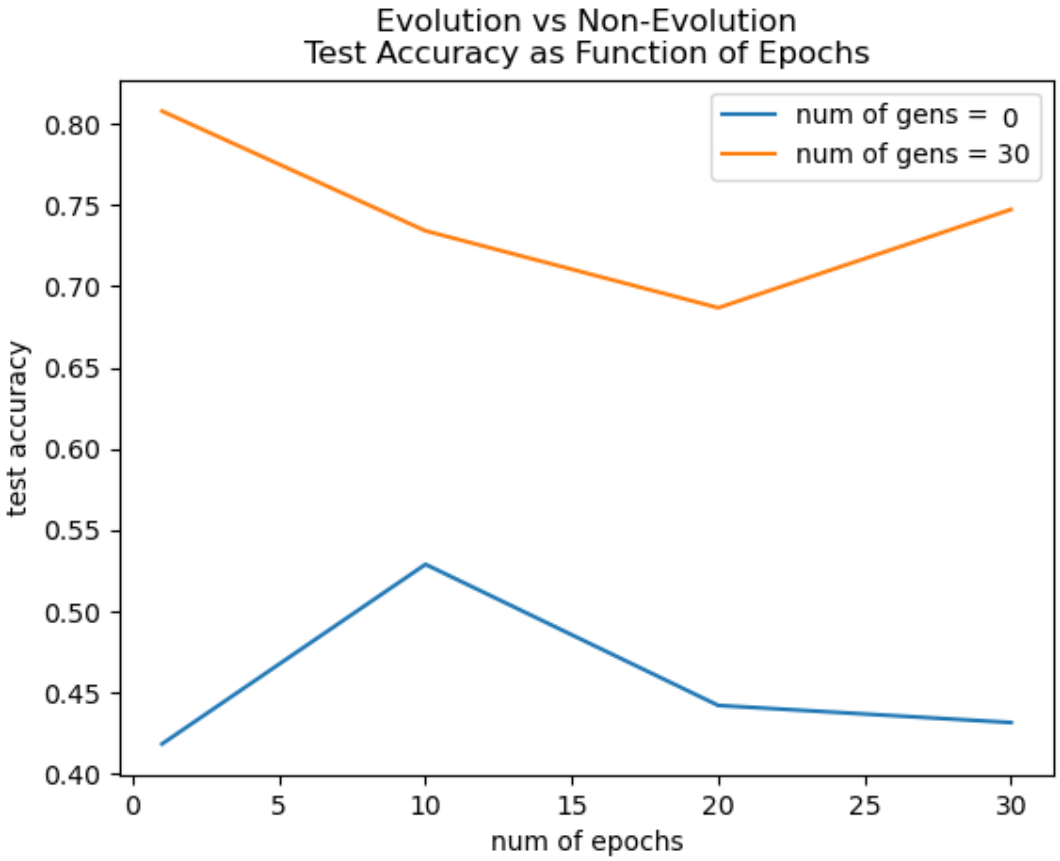

Fitness As Function Of Generation
20 Gen Size, 3 Parents
num_of_bests = 0
num_of_bests = 2
num_of_bests = 4
Best Fitness
Generation


Fitness As Function Of Generation
20 Gen Size, Keeping 0 Bests
num_of_parents = 1
num_of_parents = 2
num_of_parents = 3
Best Fitness
Generation


Fitness As Function Of Generation
20 Gen Size, Keeping 2 Bests
num_of_parents = 1
num_of_parents = 2
num_of_parents = 3
Best Fitness
Generation

100 runs average

gen size = 30, num of parents = 3, num of bests = 2



Num of Gens	Epochs	Avg Run Time (Sec)
0	1	3.92
0	10	4.33
0	20	3.97
0	30	4.24
30	1	81.82
30	10	80.27
30	20	91.19
30	30	265.22

Conclusions, lessons learned, thoughts on future improvement

Evolutionary algorithm as prior step to traditional backpropagation may save future learning time, although by itself takes an exclusive computation time. At the same time the algorithm was able to perform 30 generations, the net could have made about 1K epochs, and get the same accuracy.

We show one reservation to the last conclusion, that in larger scale, and with more threads and GPUs, the Evolutionary method has better ability to run in parallel then traditional fully connected NN backpropagation. That because every two organisms are independent, so one fitness score is not affected by the other. Moreover, two different crossover processes are also independent what make it possible to do those steps in parallel. We believe that this method worth testing in larger scale for future works.

Another experiment worth checking might be to try different activation functions, and crossover between two functions using arithmetic actions. This modification may cost much less computation time the method we explored by using methods as Fast Furrier Transform for functions multiplication.

References

https://medium.com/swlh/genetic-artificial-neural-networks-d6b85578ba99

https://towardsdatascience.com/evolving-neural-networks-b24517bb3701

https://en.wikipedia.org/wiki/Artificial_neural_network

https://machinelearningmastery.com/weight-initialization-for-deep-learning-neural-networks/#:~:text=Weight%20initialization%20is%20an%20important%20consideration%20in,of%20a%20neural%20network%20model.&text=Weight%20initialization%20is%20a%20procedure,of%20the%20neural%20network%20model

Rocha, M., Cortez, P., & Neves, J. (2003, December). Evolutionary neural network learning. In *Portuguese Conference on Artificial Intelligence* (pp. 24-28). Springer, Berlin, Heidelberg.

Our Code

https://github.com/noamatia/BS-CS/tree/main/Topics%20in%20BioInspired%20Computing