# CIS 600: Evolutionary Machine Learning

Analysis of DNN using GA

Ansuman Sasmal (asasmal@syr.edu), Bharat Jangama (bjangama@syr.edu)

May 3, 2023

**Abstract**

Tuning many hyperparameters is a hard and time-consuming part of training deep neural networks (DNNs). Using genetic algorithms (GAs) to search the hyperparameter space is one way to optimize this process. An evolutionary algorithm known as a GA evolves a population of potential answers to a problem using selection, crossover, and mutation. GAs can be used to optimize hyperparameters like the learning rate, quantity of hidden layers, and activation functions in the context of DNN training. The fundamental concept is to generate and analyze many sets of hyperparameters using GAs, with the aim of identifying the one that yields the best results for a particular task. Despite the fact that GAs can effectively optimize DNNs.

## 1 Introduction

A method that combines the strength of deep learning with the optimization abilities of genetic algorithms is training Deep Neural Networks (DNN) using Genetic Algorithms (GA). The basic idea behind this procedure is to initialize a population of neural networks with random weights and biases before assessing how well they perform on a specific dataset. Then, in a process that resembles natural selection, the networks that perform best are chosen to create offspring through crossover and mutation procedures. Several generations of the procedure are spent iterating until a successful network is discovered. The main benefit of employing genetic algorithms for DNN training is that they can prevent becoming stuck in local minima, a typical issue with conventional optimization techniques. This approach has shown promising results in various applications, including image and speech recognition, and can be applied to solve a wide range of real-world problems.

An artificial neural network known as a DNN, or deep neural network, is made up of several layers of interconnected nodes (neurons) arranged in a hierarchical framework. A DNN gradually transforms the input data into a more abstract and complicated representation that is appropriate for the job at hand by processing the output from the layer before it. An input layer, one or more hidden layers, and an output layer are frequently found as layers in a DNN. Depending on the particular problem being solved, the number of hidden layers and the number of nodes in each layer may change. In a variety of applications, such as speech and image recognition, natural language processing, and many more, DNNs have proven to be quite successful.

A genetic algorithm (GA) is a type of optimization algorithm that is inspired by the process of natural selection and evolution. In a GA, a population of potential solutions to a problem is generated and iteratively improved through a process of selection, crossover, and mutation, mimicking the biological process of reproduction and genetic variation.

The algorithm starts by generating an initial population of potential solutions, each represented as a set of parameters or a "chromosome". The fitness of each chromosome is then evaluated based on how well it performs on the given task or problem. The fitter chromosomes are selected to undergo crossover, where two or more chromosomes are combined to generate a new set of offspring. Mutation is then applied to introduce random changes to the offspring's chromosomes, which adds diversity to

the population.

The process of selection, crossover, and mutation is repeated iteratively over multiple generations, with the aim of producing increasingly better solutions to the problem. The algorithm terminates when a stopping criterion is met, such as a maximum number of iterations or when a satisfactory solution is found.

GAs have been used to solve a wide range of optimization problems, including function optimization, parameter tuning, and feature selection, among others. They are particularly useful for problems that have a large and complex search space, where traditional optimization techniques may not be effective.

# 2 Dataset

Dataset: Diabetic Retinopathy Debrecen Data Set

link:https://archive.ics.uci.edu/ml/datasets/Diabetic+Retinopathy+Debrecen+Data+Set

This dataset contains features extracted from the Messidor image set to predict whether an image contains signs of diabetic retinopathy or not.

# 3 Criteria

To compare multiple DNN architecture using GA on Diabetic Retinopathy Debrecen dataset and compare the results. Results are averaged over ten trials, comparing confusion matrix and comparing with shallow feed forward neural network.

# 4 Results

## 4.1 Model Analysis

TABLE I
MODEL ANALYSIS

| Model | Model 1 | Model 2 | Model 3 | Model 4 | Model 5 | Model 6 |
|---|---|---|---|---|---|---|
| NN config | (19,38),(38,19), (19,1) | (19,128),(128,256), (256,512), (512,256), (256,128),(128,1) | (19,128),(128,256), (256,128), (128,1) | (19,128),(128,256), (256,128), (128,1) | (19,128),(128,256), (256,128), (128,1) | (19,128),(128,256), (256,128), (128,1) |
| selection | roulette | roulette | roulette | rouletteinv | roulette | rouletteinv |
| Mutation | gaussian(0.1) | gaussian(0.1) | gaussian(0.1) | gaussian(0.1) | gaussian(0.1) | gaussian(0.1) |
| crossover | uniform crossover | uniform crossover | uniform crossover | uniform crossover | uniform crossover | uniform crossover |
| mutation rate | 0.95 | 0.95 | 0.95 | 0.95 | 0.95 | 0.95 |
| crossover rate | 0.9 | 0.9 | 0.9 | 0.9 | 0.9 | 0.9 |
| population size | 2*19*39 | 2*19*129 | 2*19*129 | 2*19*129 | 2*19*129 | 2*19*129 |
| epsilon | 0.02 | 0.02 | 0.02 | 0.02 | 0.02 | 0.02 |
| iteration | 200 | 40 | 40 | 40 | 100 | 30 |
| accuracy | 52.3 | 60.61 | 60.03 | 60.18 | 61.73 | 62.9 |
| time | 0.66 hrs | 10.5 hrs | 4.4 hrs | 3.8 hrs | 5.3 hrs | 3.9 hrs |
| confusion matrix | 61.8 100.2<br>57.4 104.6 | 79.6 82.4<br>78.3 83.7 | 72.1 89.9<br>67.8 94.2 | 79.3 82.7<br>84.4 77.6 | 85.0 77.0<br>89.6 72.4 | 97.6 64.4<br>93.0 69.0 |

Figure 1: Model

## 4.2 Graphs

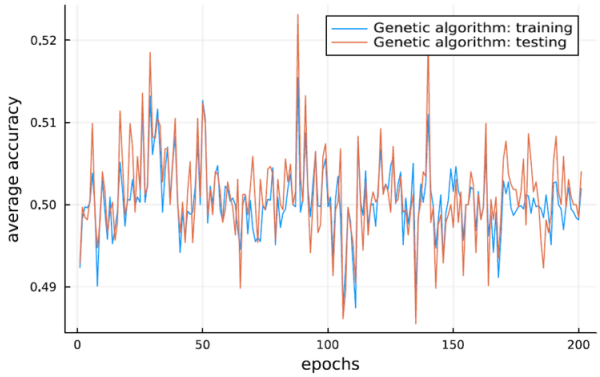


Figure 2: Model 1

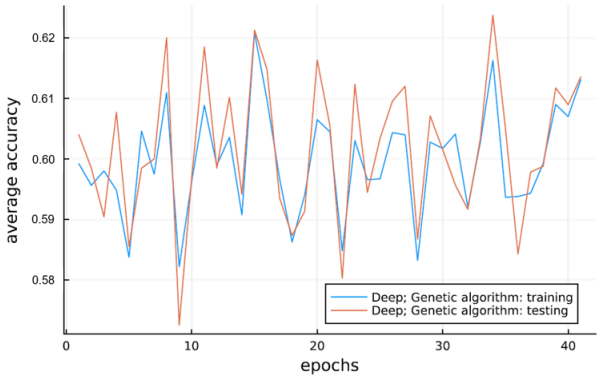


Figure 3: Model 2

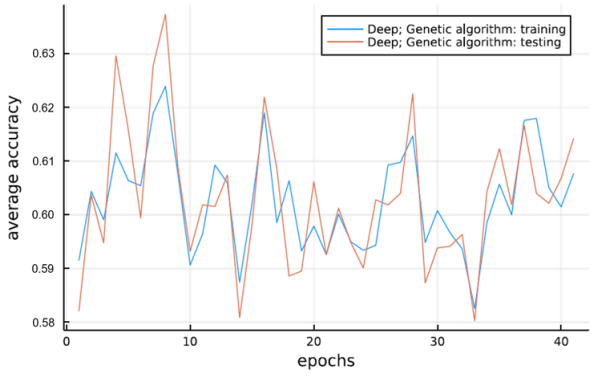


Figure 4: Model 3

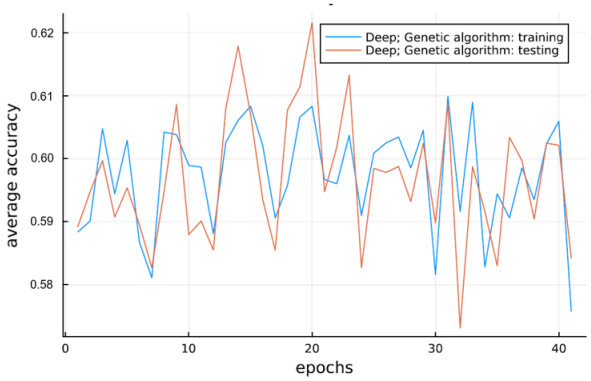


Figure 5: Model 4

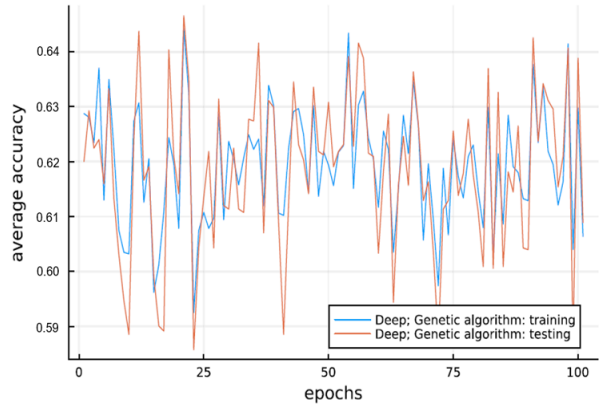


Figure 6: Model 5

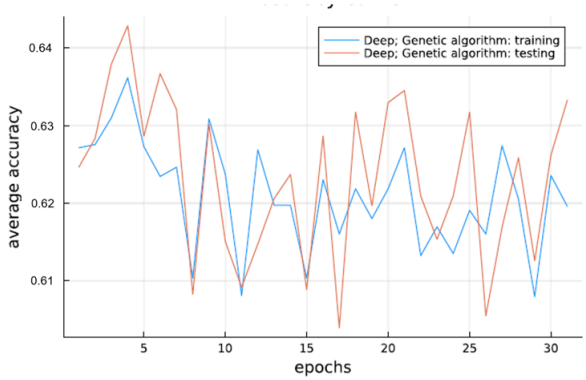


Figure 7: Model 6

### 4.3 Confusion Matrix

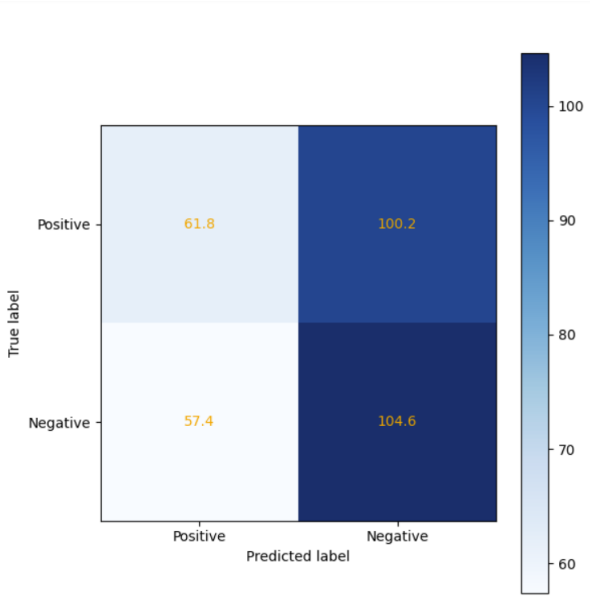


Figure 8: Confusion Matrix Model 1

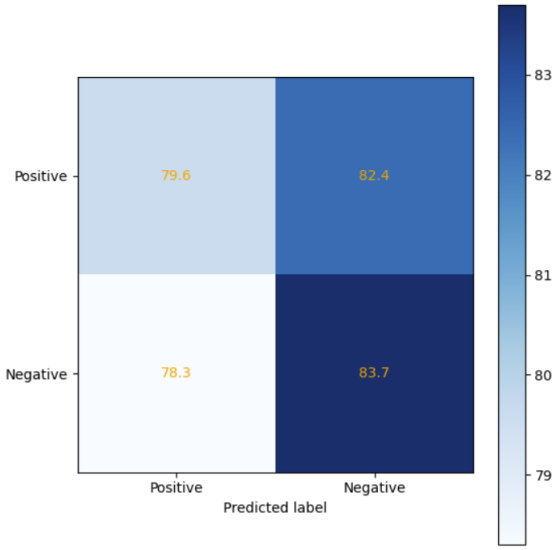


Figure 9: Confusion Matrix Model 2

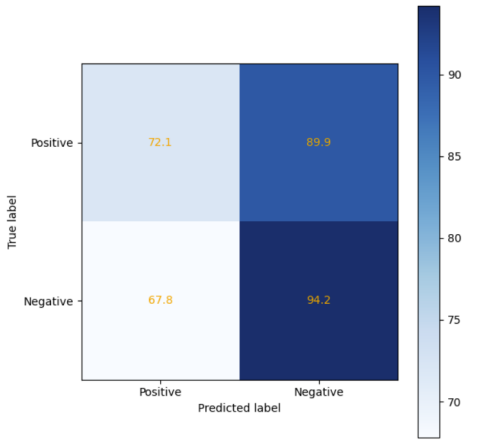


Figure 10: Confusion Matrix Model 3

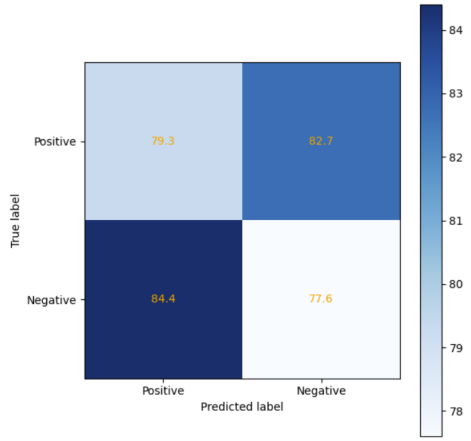


Figure 11: Confusion Matrix Model 4

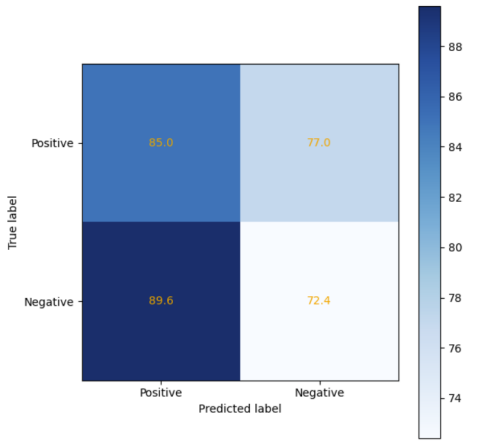


Figure 12: Confusion Matrix Model 5

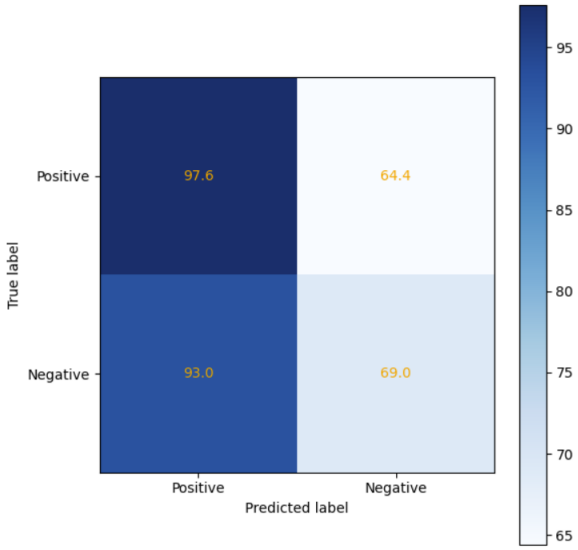


Figure 13: Confusion Matrix Model 6

### 4.4 Accuracy vs time taken

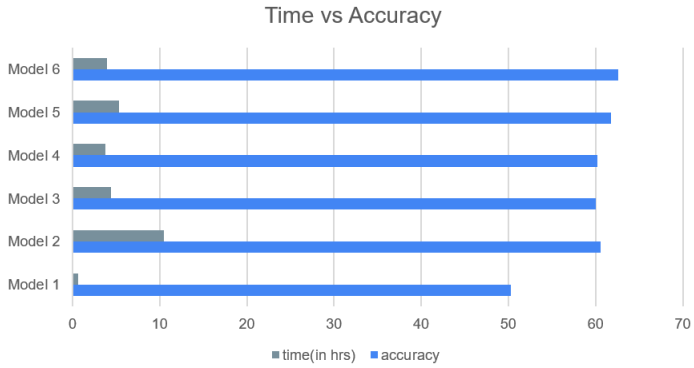


Figure 14: Time taken for execution vs accuracy

## 5 Conclusions

In this experiment in order to find a more accurate way to detect diabetic retinopathy, Deep Neural Network with GA integration is found to provide more accurate results compared to shallow forward feed model. While having the number of layers played an important role where we can observe in model 2 that there were 4 hidden layers and took 10.5 hours to train only to provide 60.61 percent accuracy, whereas model 6 with just 2 hidden layers took almost 4 hours to train to provide 62.9 percent accuracy from Figure 1. Finding the optimal Neural Network configuration plays a major role in finding the accuracy of the model. We can also conclude that the Genetic algorithm integrated with the Deep Neural Network improves the performance of the model.