In this project, I implemented a convolutional neural network and used it to classify handwritten digits in the MNIST dataset. After roughly 8 hours of training, I was able to create a model that achieved a 96.4% accuracy on the test set. The final version of the model that I ended up going with had 3 convolutional layers with 32 2x2 filters in each layer and a dense layer that uses a single hidden layer that is 100 nodes wide.

My initial approach to tuning the hyper parameters for my model was making every possible decision a parameter that would be optimized. So activation functions, kernel sizes, strides, number of hidden layers, etc would all be something that can be tuned for. But if you just tried to iterate through a range of possible values for each of the parameters you would end up training several hundreds if not thousands of models in trying to find the optimal combination.

To avoid this issue, I used genetic algorithms to tune my model. An individual in the population would represent some combination of hyper parameters and their fitness score would be the corresponding validation error that the model that used their parameters achieved after training. Genetic algorithms can occasionally lead to exponentially increasing population sizes but we can actually keep the population size roughly stable by adjusting two parameters in the algorithm: the number of individuals (the rate of birth) added each generation and the number of generations that an individual is allowed to stay in our population (the age limit). After some trial and error, I found that if the birth rate is set to 25% and the age limit is 3 then the population doesn't decay or shoot off. It oscillates and doesn't vary too much from generation to generation.

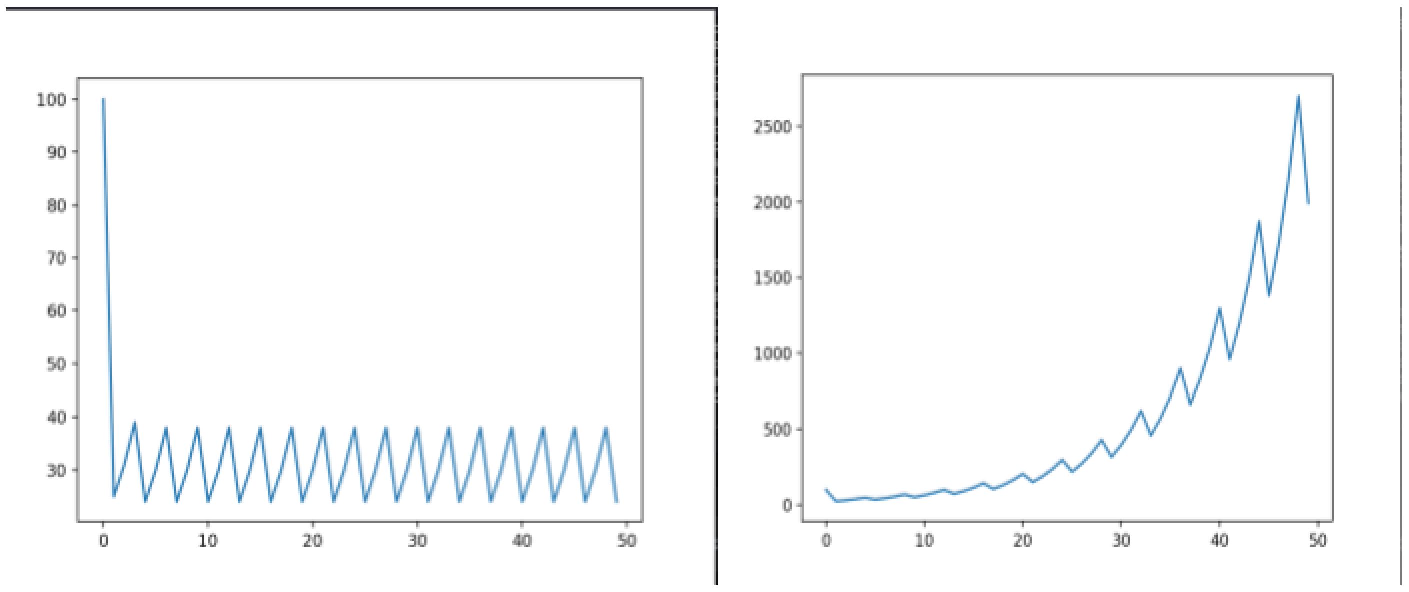

Figure 1: Population Size as a Function of Number of Generations with (a) an age limit of 3 and 1.25 birth rate and (b) an age limit of 4 and a 1.25 birth rate

However, while this did reduce the number of times I needed to perform training, the training itself was still fairly long so I wasn’t able to tune all of the parameters that I wanted. Instead I decided to fix everything except for the training parameters (regularization, dropout, and step size).