Programming Assignment #1

Task 0

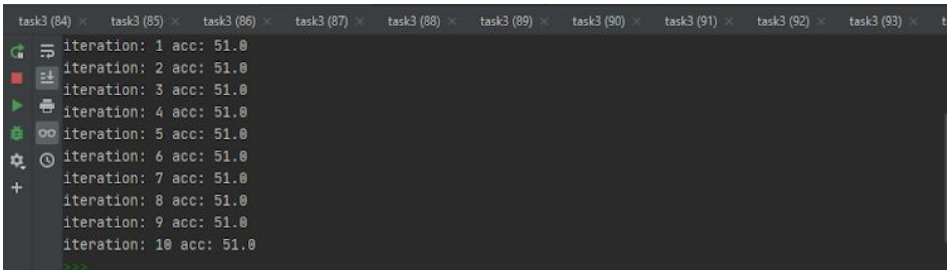
This task was simply to just generate training and test data using Gaussian distribution and an identity matrix. Process is explained in the code.

Task 1

This task was to build the fully connected layer model consisting of inputlayer, 2 fully connected layers to an output layer. Where ReLU was the only activation function. The weights to these layers was also initialized with Gaussian distribution.

Task 2

In this task I just had to implement a simple forward pass using the test data where the input data was calculated with the weights and biases of each neuron in the layers. The accuracy was calculated by dividing the amount of correct samples with total samples multiplied by 100 to get the percentage.

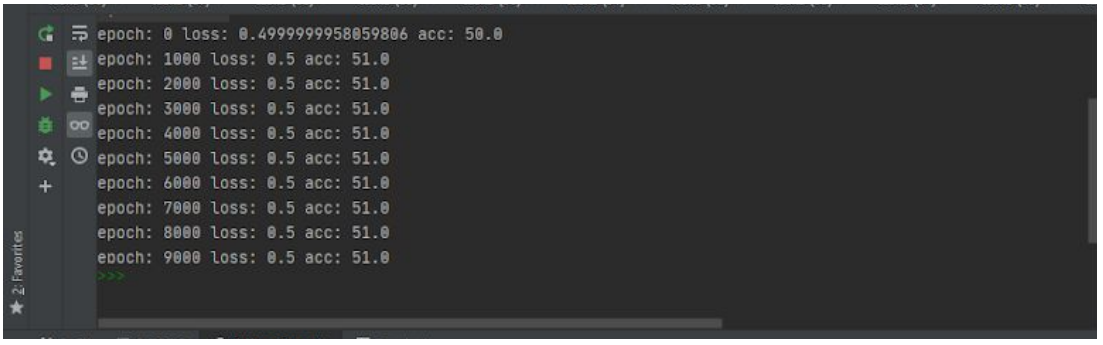


Task 3

In this last task it got more complicated. This was the last step for a neural network to be working correctly, and I had to implement a backward pass. The backward pass was implemented, and weights where updated through the Stochastic Gradient Descent method, but there was some problems using the ReLU as both activations during the backwards propagation making almost all the new weights close to 0. Then using the MSE as the loss function and getting the mean from one hot encoded labels only returns the average which came out as 0,5 everytime. By using SGD i didn't see any progress in my NN, as the loss and accuracy stayed the same. So i implemented my own optimizer which generated random weights and just kept the best ones with lowest loss. And I saw progress in my NN. The code is also commented, so you would have a good overview of my thoughts. I also found out that a learning rate of 0.003 worked best for my NN.

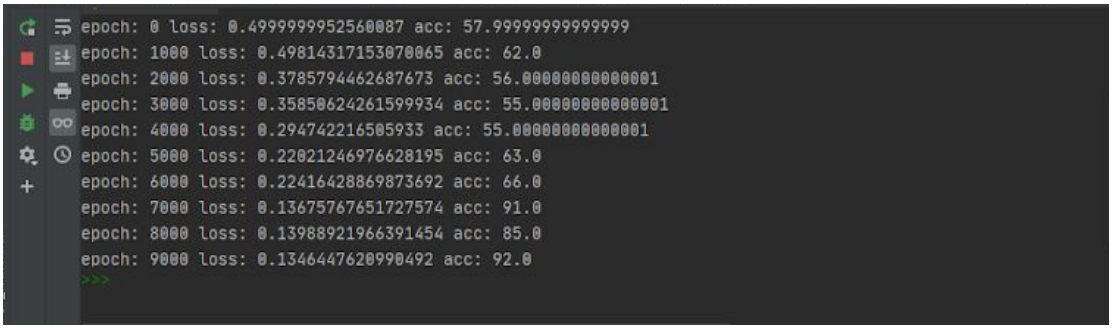
There are not a lot of documentation of using MSE online with one-hot-encoded labels, and most people suggest using softmax, since it is easier to work with one-hot-encoding.

With SGD



Accuracy and loss stays the same

With my own made optimizer



Both accuracy and loss gets better.