**Project1**

**Neural Networks**

**Name/ Mohamed Gomaa Moustafa**

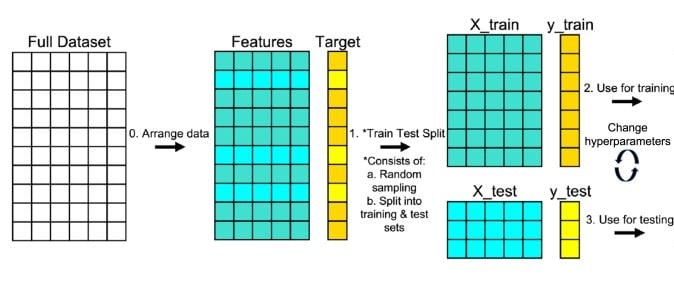
**Name/ Ali Gamal Mohamed Salem**

**Id/2103129**

**Id/2103128**

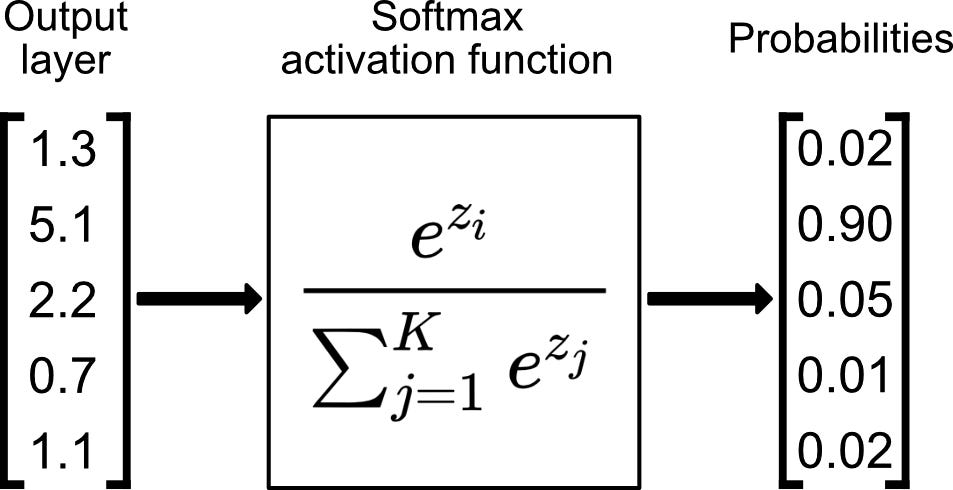
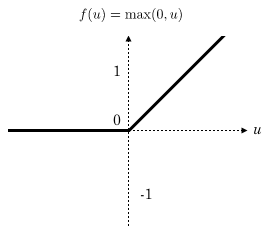
**Step1:**

Firstly, we downloaded the **minist** data from <https://www.kaggle.com/> . Before we start we checked our dataset if it included any null values and Elhamdollah there wasn’t any null values . After that we split our dataset x which is the feature and y is our label to 20 percent validation and 80 percent training . Then we set our batch size and used dataloader to split the training and validation sets into batches . We checked if our device has a gpu and if it has it we will use it in further computations which will make much faster and easier .

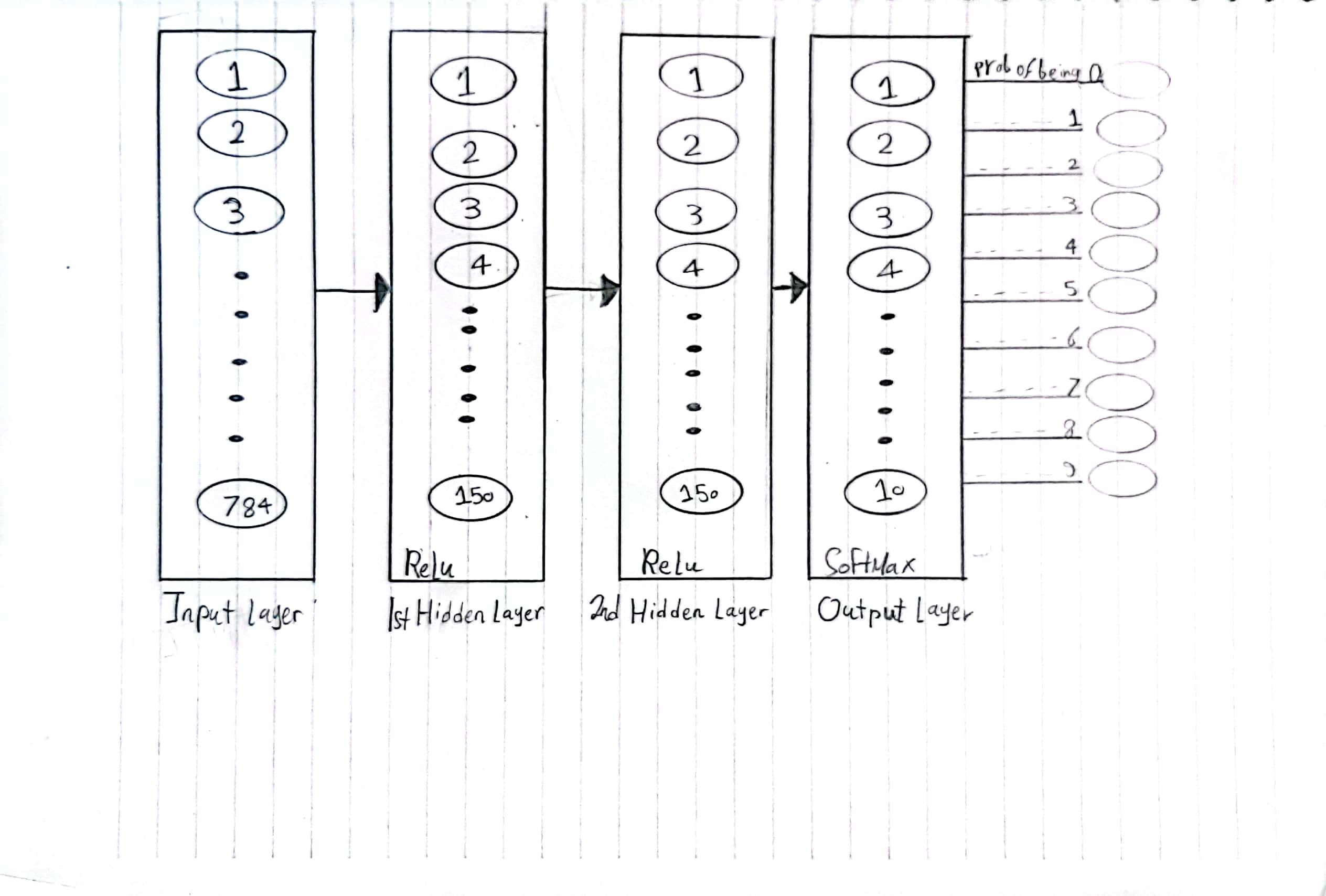


**Step2:**

Secondly, we created our model by 3 layers (2 hidden layers and one output). We take our picture with a shape of 28\*28 which equals to 784 which will be the number of features in our input layer . every hidden layer consists of 150 unit and linear function and a non linear activation(for non-linearity) function and we will choose relu in our case . The output layer consists of 10 units (The numbers between 0 and 9) and linear function and a non linear activation(for non-linearity) function and we will choose softmax to give us the probability of each number and it will take the maximum probability which will be our predict .



Our Model Architecture



As given, We will use Cross Entropy Loss function(**hint!!** In our implementation in the code we will not use softmax as the cross entropy already use it before calculating the loss). We set the learning rate to 0.01. We will use Adam optimizer to minimize our loss function.

We will train our model for 10 epochs and each epoch consists of 375 Iteration .

We will calculate the accuracy and loss every 75 Iteration which means 5 times per epoch for training and validation . We will print the loss and accuracy for the training and validation in every 250 Iteration. We reached 90 percent accuracy for the training and validation .

Now we will show our model performance on **minsit** data:

A black screen with white text

Description automatically generated with low confidence

Now will show the plots:

1-performace on training:

A picture containing text, screenshot, font, plot

Description automatically generatedA picture containing text, diagram, plot, line

Description automatically generated

2-performace on validation:

A picture containing text, font, screenshot, line

Description automatically generated A picture containing text, diagram, plot, line

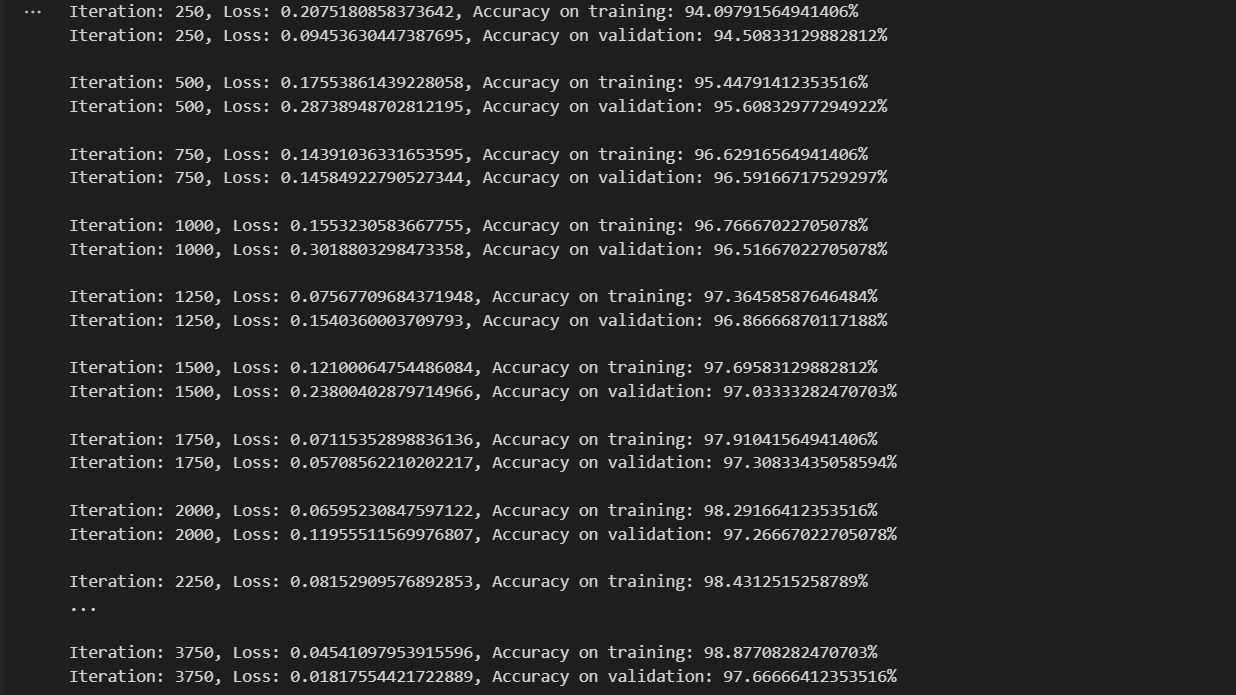
Description automatically generated

**Step 3**

Now we will add layer normalization and dropout to our model . Layer normalization enables smoother gradients, faster training, and better generalization accuracy. And dropout prevents overfitting. We will see later it will increase the accuracy of our model and decrease the loss.

After some trails we get the best performance when we use layer normalization after each hidden layer and use one dropout layer after the 2nd hidden layer but off course we put the dropout layer before the normalization layer so we make less computations(**Hint!!** We used the default probability of dropout which is 0.5)

Our model performance:



Now with the plots:

1-performance on training

A graph showing the difference between loss and loss

Description automatically generated with low confidence A picture containing text, diagram, plot, line

Description automatically generated

2-performance on validation :

A picture containing text, font, screenshot, handwriting

Description automatically generated A picture containing text, line, plot, diagram

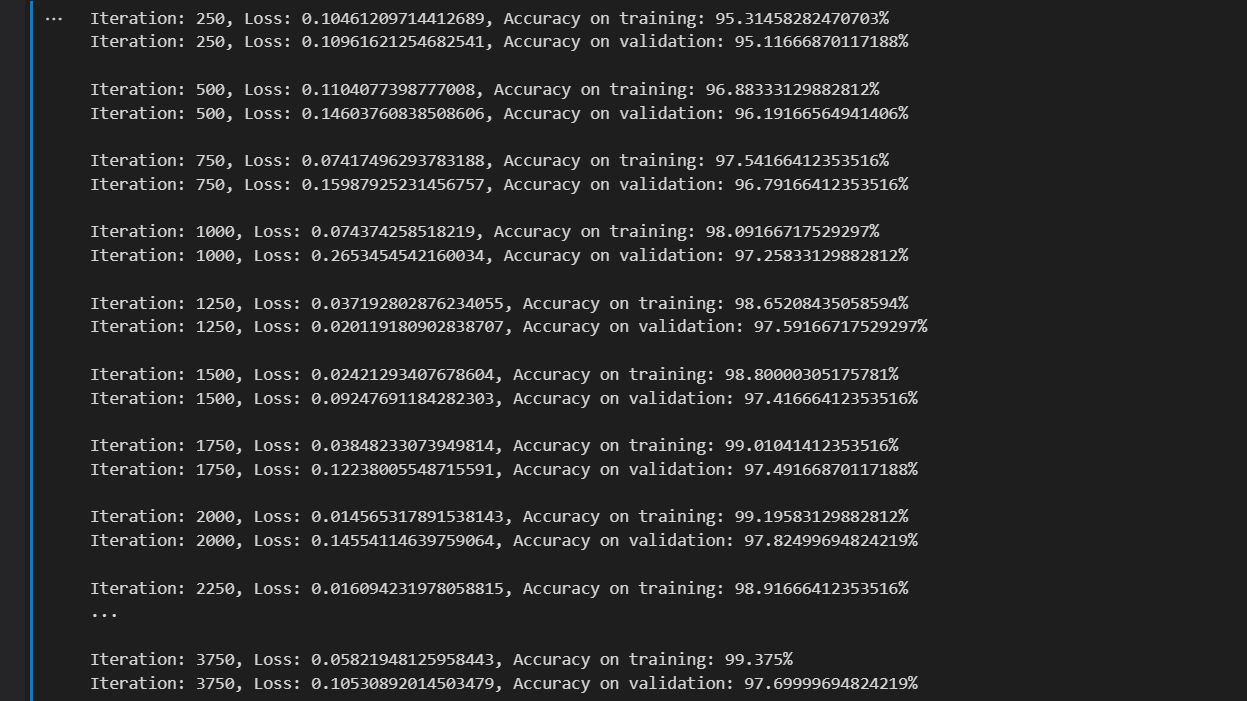
Description automatically generated

From the plots we can see that the accuracy of our model increased and the model become more generalization because we added the normalization and dropout layer as we said that previously.

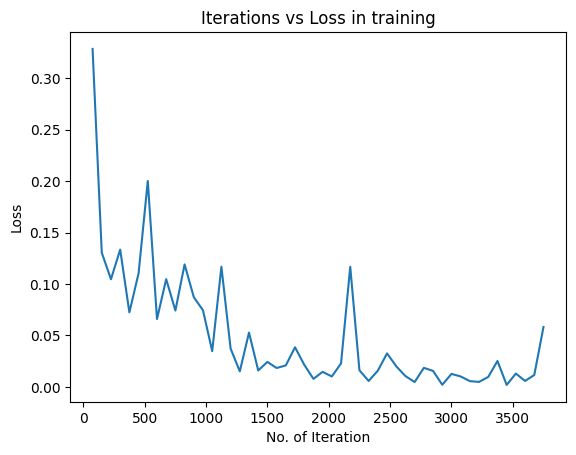
**Step 4**

Now we will compare between different learning parameters and the probability of the dropout layer .

1. Learning rate= 0.001 and probability = 0.2



And the plots:

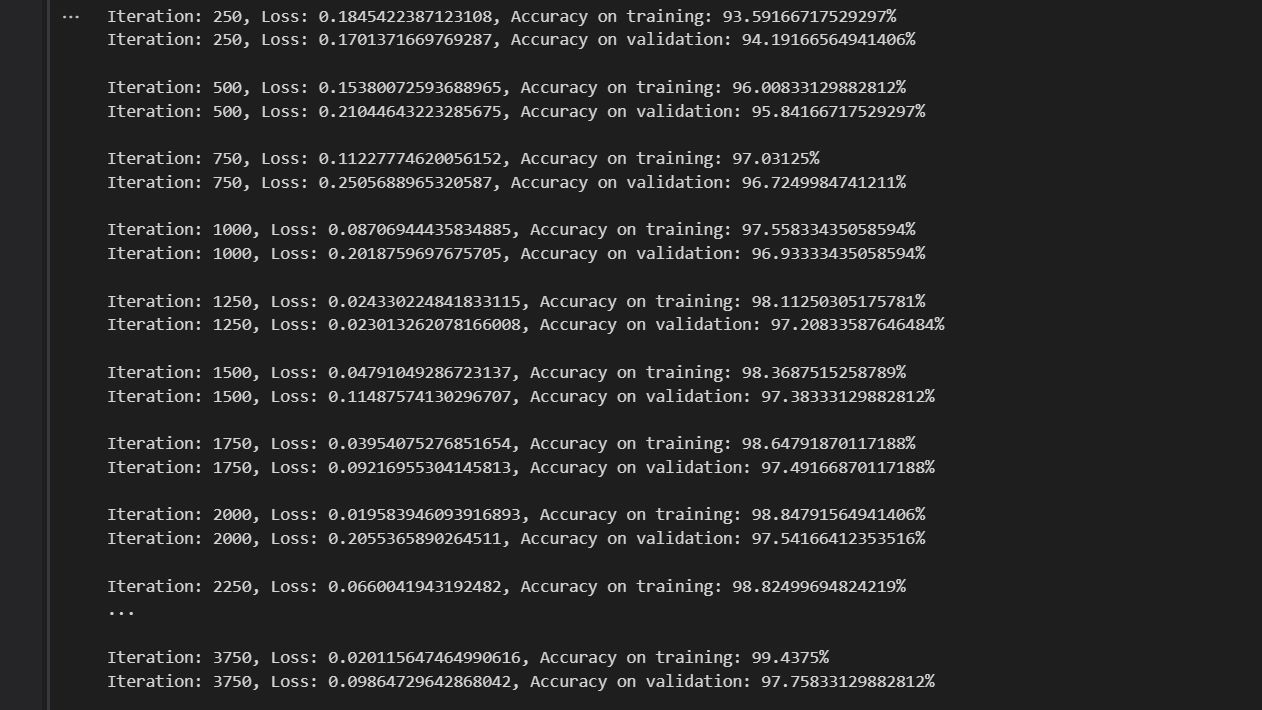
A picture containing text, font, screenshot, handwriting

Description automatically generated A picture containing text, line, plot, diagram

Description automatically generated

We can se that this hyperparameter are much better because it make our model have better performance and better accuracy on the training and validation set and also it make our model faster to find the min of our activation function and the probability of dropout will make a 20 percent of our unit to be zeroed

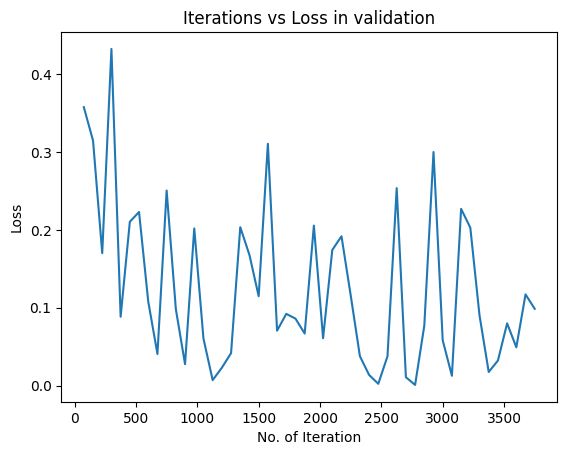
2- Learning rate= 0.0005 and probability = 0.4



The plots:

A picture containing text, diagram, plot, line

Description automatically generatedA picture containing text, diagram, plot, line

Description automatically generated  A picture containing text, plot, diagram, line

Description automatically generated

We can see that the model is a little slower in the start than previous hyperparameters to find the best solution but at the end it will have nearly the same accuracy and the probability of dropout will make a 40 percent of our unit to be zeroed

3-learning rate=0.0001 and probability = 0.6

A picture containing text, screenshot, font, document

Description automatically generated

The plots:

A picture containing text, diagram, plot, line

Description automatically generatedA picture containing text, line, diagram, plot

Description automatically generatedA graph showing the difference between loss and loss

Description automatically generated with low confidenceA picture containing text, diagram, line, plot

Description automatically generated

Now with these hyperparameter the learning rate is to small which will make our model much slower to find its goal but at the end it will reach the goal anyway and the probability of dropout will make a 60 percent of our unit to be zeroed