Deep Learning

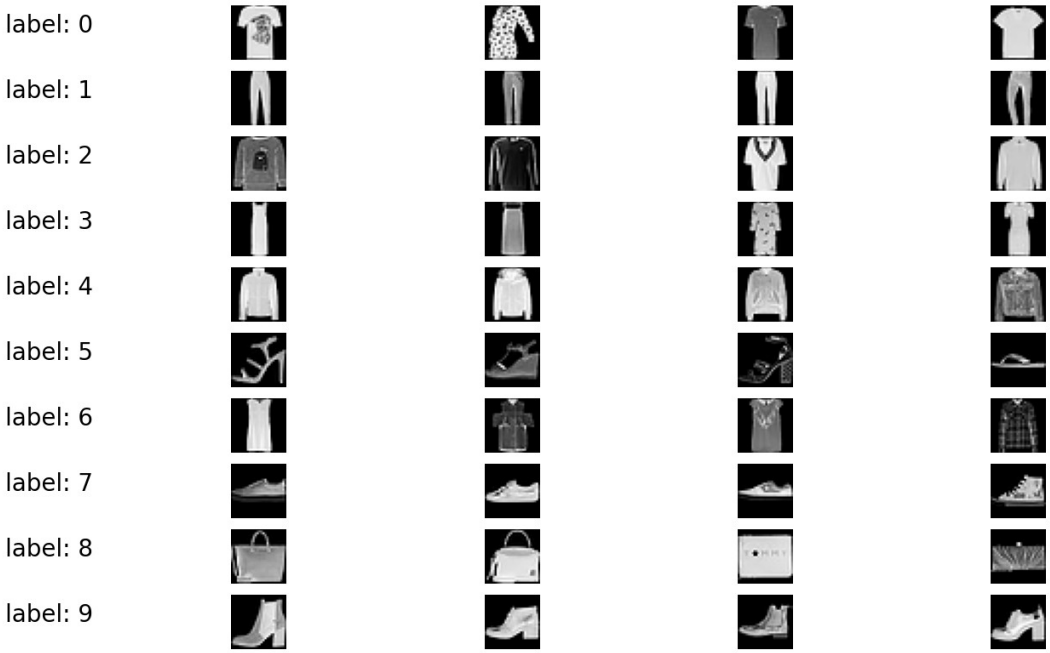
Programming Exercise 2

Ofer Idan, 038467668

Ron Ferens, 037222825

Part 1 - Visualize the Data

The following are 40 (10x4) samples showing 4 different examples from each class in the training dataset:

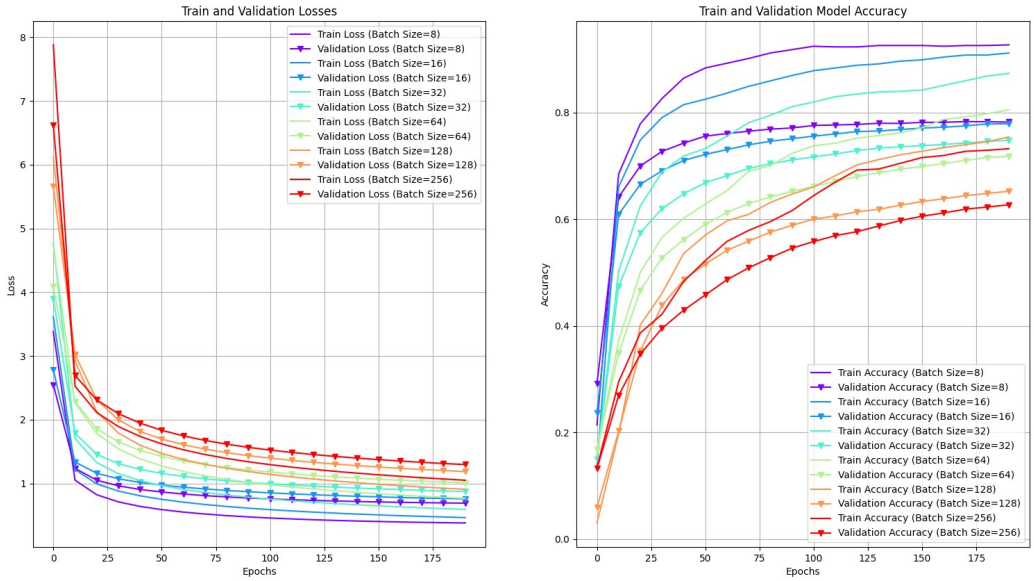


Part 2 - Logistic Regression Classifier

In the following section we will showcase the impact of different configurations:

1. Batch size
2. Learning rate
3. Regularization coefficient

2.1. Batch Size

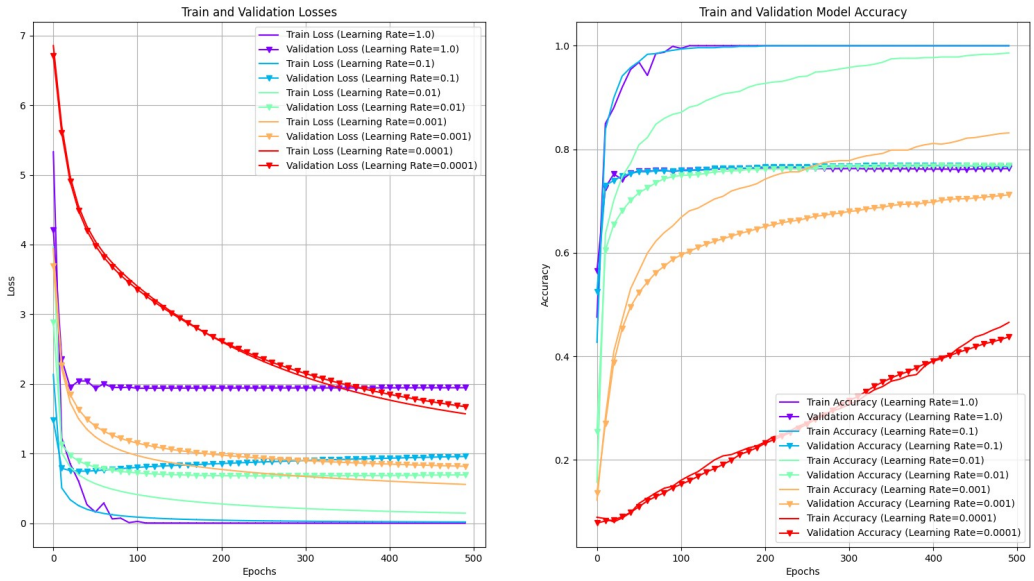


The graphs above shows the impact of changing the batch size used by the logistic regressor.

We have test batched size 8, 16, 32, 64 , 128 and 256.

From the graphs above, we can notice that, as expected, bigger batch size will results in lower overfitting as it acts as a regularization parameter for the model. However, bigger batch size will require notable longer training time.

2.2. Learning Rate



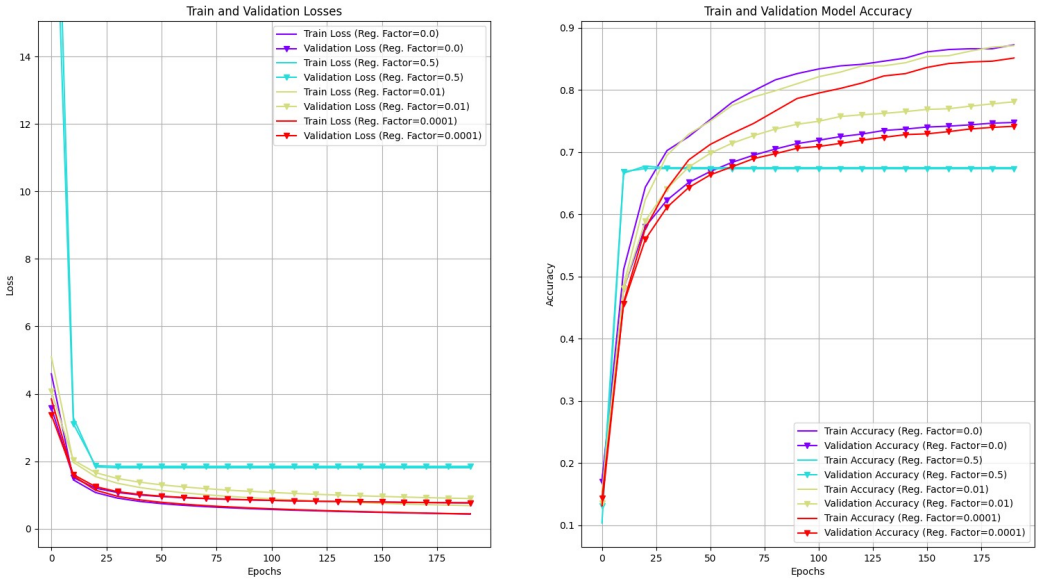
Changes in learning rate used for training a logistic regression model are shown in the graphs above. We have tested learning rate of 1.0, 0.1, 0.01, 0.001 and 0.0001.

When using a too high learning rate (1.0 and 0.1) the model converge very fast, but tend to suffer from higher overfitting – larger gap between the training and the validation accuracy.

On the other hand, when using a very low learning rate (0.0001) both the training and validation accuracy are very similar, but the model's conversion is very slow.

Using a "just-right" learning rate (0.01) results a controlled conversion of the model with acceptable gap between the training and the validation accuracy.

2.3 Regularization Coefficient



The imapct of modifiying the regularization coefficient can be seen in the graphs above. We have tested regularization coefficient of 0, 0.5, 0.01 and 0.0001.

As seen, when applying a high coefficient will reults in a “too-strong” contraint over the model’s training. Although it almost totaly eliminates the overfitting, it also harms the model’s performance and results in a poor accuracy.

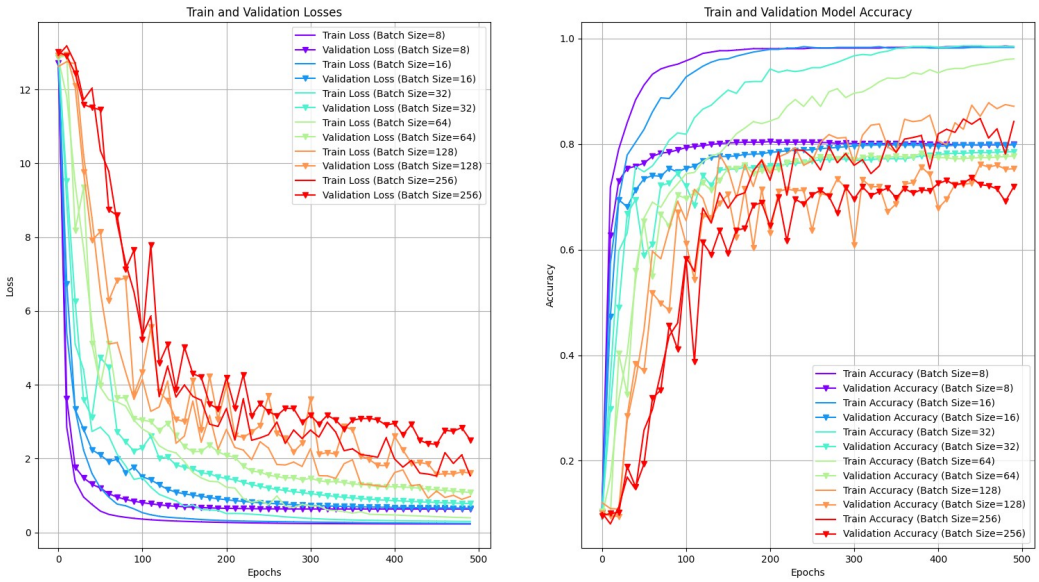
Using a low regularization coefficient or removing it (0 or 0.0001) will result in higher overfitting, compared to a correct value (0.001-0.005) that will better balance the train and validation accuracies.

Part 3 - Neural Network with One Hidden Layer

A comparison of different configurations will be presented in the following sections:

1. Batch size
2. Learning rate
3. Hidden layer dimension
4. Regularization coefficient
5. Selection of activation function
6. Applying Dropout

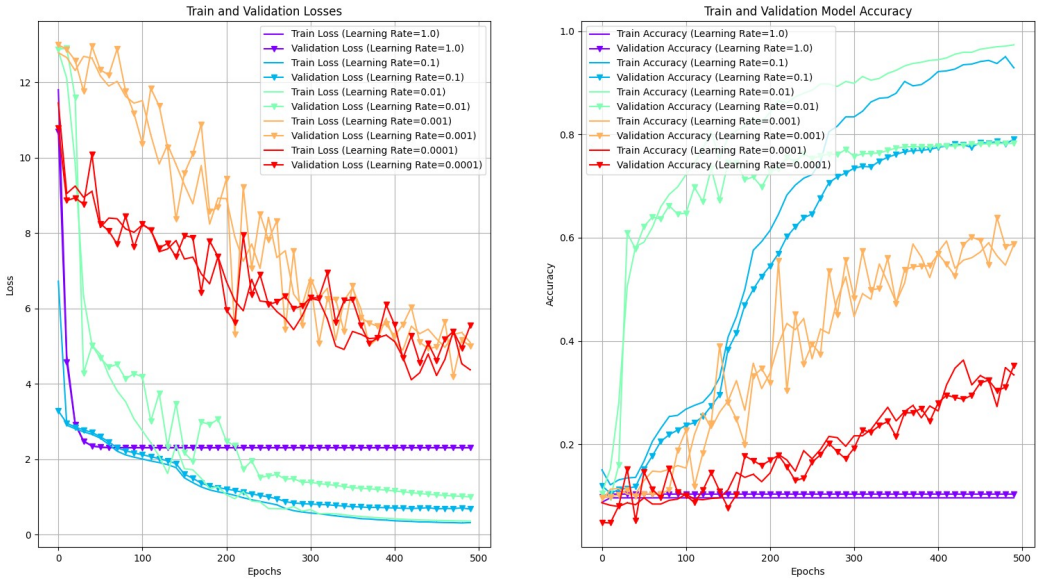
3.1. Batch Size



As you can see from the graphs above, changing the batch size used for training a neural network has a significant impact on its performance. We have test batched size 8, 16, 32, 64 , 128 and 256.

In the case of the neural network, the model converged to quite the same validation accuracy – around 80% correct classification. However, we can note that using larger batch size requires longer training time, but tend to less overfit when comparing the training and validation accuracy.

3.2. Learning Rate



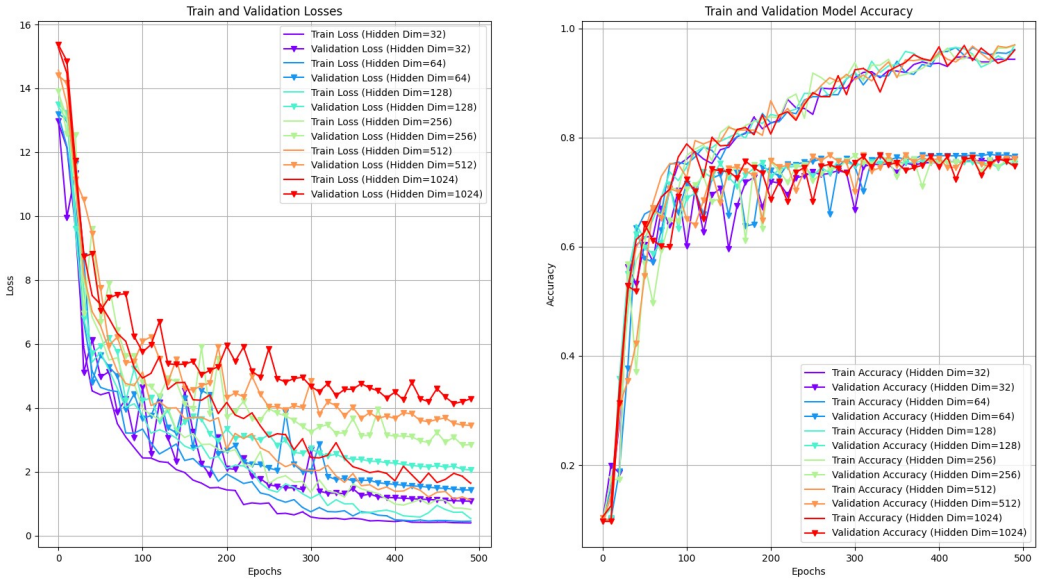
The graphs above illustrate changes in learning rates used for training the neural network model. We have tested learning rate of 1.0, 0.1, 0.01, 0.001 and 0.0001.

As seen, high learning rate (1.0) prevent the model from converging and result in very poor performance – only 10% accuracy for both training and validation.

Based on these experiment, it seems that for the current netowrk, a learning in the range of 0.1-0.01 will probably be a good selection. One can notice that tending more toward 0.1 will result in lower overfitting.

In case a lower learning rate is applied (0.001 and 0.0001) we can see that the model does converges, but in much slower pace and will require very long training time.

3.3. Hidden layer dimension

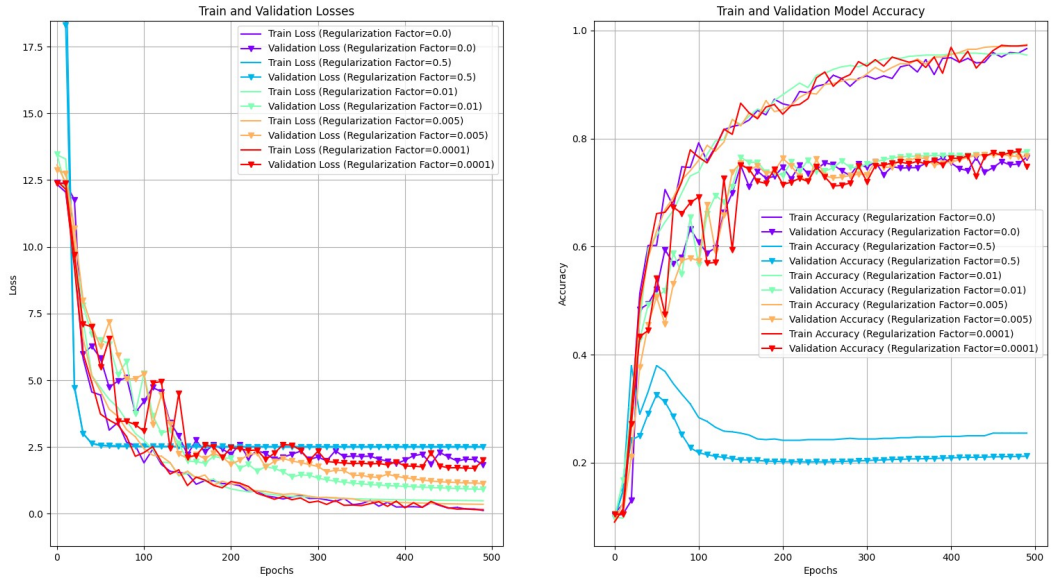


The hiddem dimension hyper parameters determines the size of the neural netwrok hidden layer (size ofthe W_1 weights). We have tested hidden layer of sizes 32, 64, 128, 256, 512 and 1024.

We can note that the performance of each one of these selections results in very similar model performance when training the network for 500 epochs. However, we should note that when setting a bigger hidden layer the network tend for bigger differences between the loss of the training and validation datasets, which might indicate possible overfitting in later phase of the model training or on the test dataset. This is aligned with the fact that choosing a bigger hidden dimension makes the model more complex which can result in higher variance.

3.4. Regularization coefficient

Using a modified regularization factor, here are the results of our experiments:

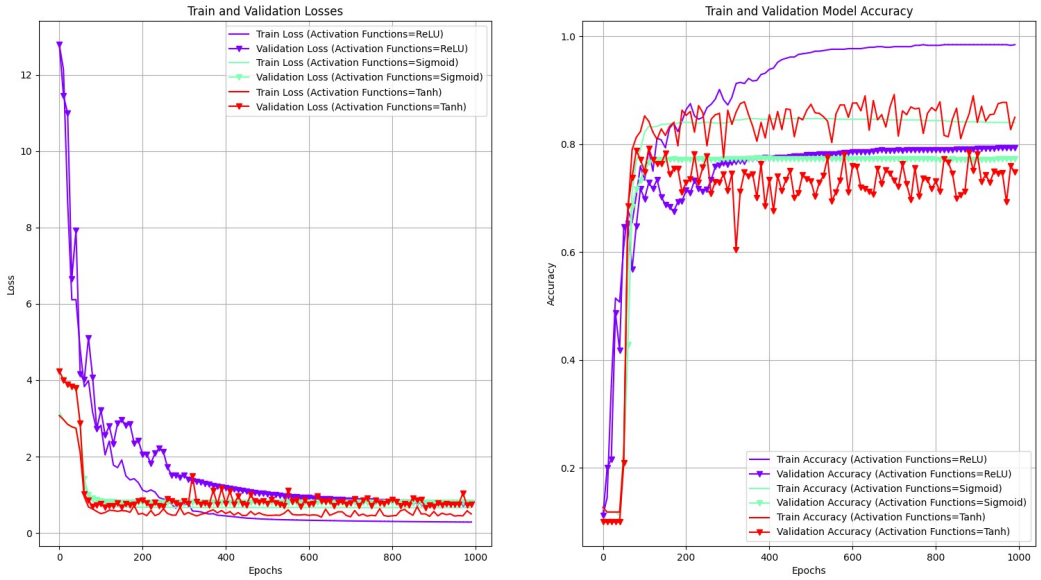


We setting the regularization factor to 0.0, we basicaly eliminate its impact on the loss (and its devirative during the backprobogation), hence we can get the imperssion of the network performance when not using regularized loss. As expected the network tend to higher overfitting compared to training that used regularization factors in the range of 0.01-0.005. When setting the regularization coefficient to even lower value it has very low impact and the model perfomance are pretty similar to the non-regularized one.

On the other size of the scale, when applying a very high coefficient (0.5) we get a very stong regularization impact and the model is not able to converge propery.

3.5. Selection of activation function

We have evaluated the imapct of using three different activation function: ReLU, Sigmoid and Tanh:

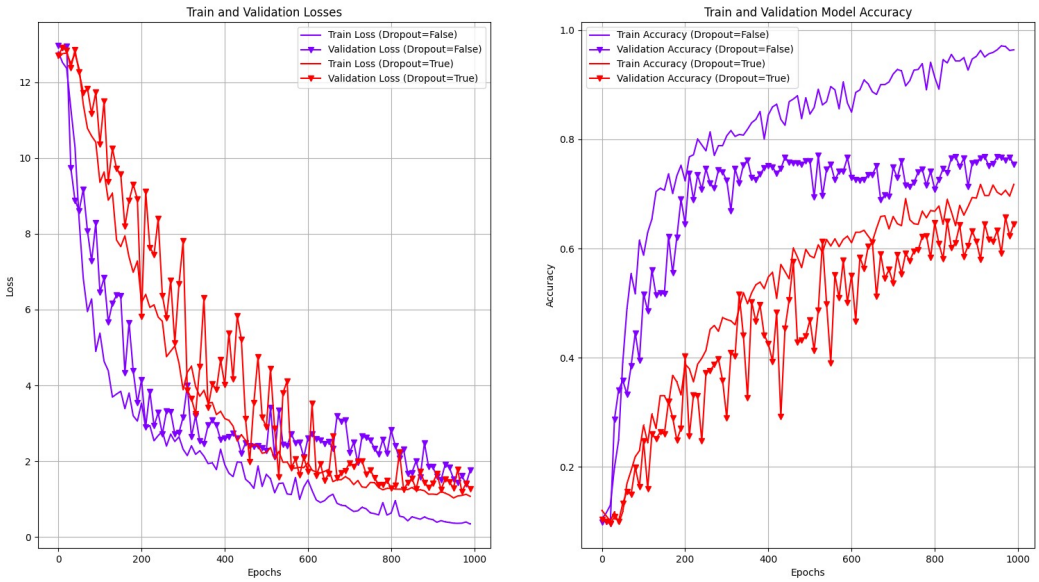


We note the the ReLU activation function results in a better model performance and generates higher classification accuracy.

In addition, we noted that both the Sigmoid and the Tanh activation function required a much higher learning rate in order to perform better. This is another disadvantage when considering which activation funtion to select since higher learning rate might harm the proper convergence of the model.

3.6. Applying Dropout

In our final ablations experiments, we have evaluated the impact of applying Dropout on the model's hidden layer. The following graphs show the comparison of the model performance with and without the Dropout layer:



As expected, since the Dropout layer is designed to handle the model overfitting we can clearly see its impact of the model behavior. When using Dropout layer with p=0.2, the model tend to lower overfitting.

Running the Code

- The entire code is placed within the main.py and conssist four different sections
    - *Part 1 - Visualize the Data*
    - *Part 2 - Logistic Regression Classifier*
    - *Part 3 - NN with a Softmax Activation*
    - *Part 4 - Models Evaluation*
- In order to run a specific part, you should comment out all others