# Fashion MNIST Classification using MLP

**Abtin Mahyar**

## PART II. Implementation

### 1. Data

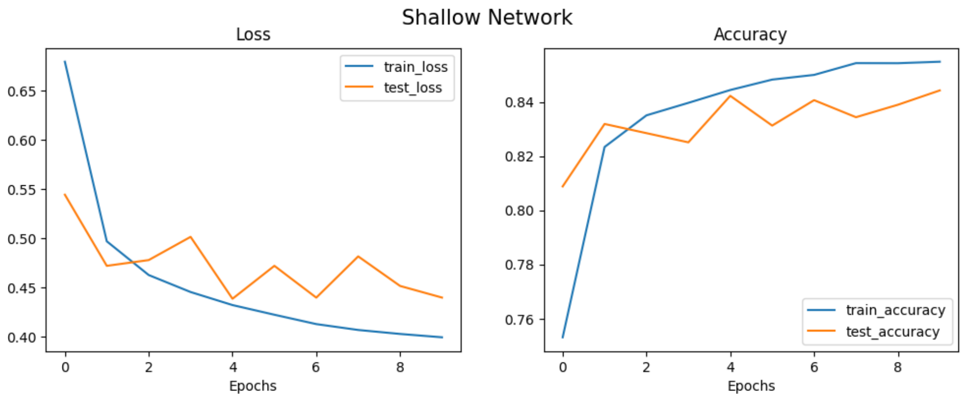
The process of aggregating and making the data ready for fitting into the models consists of following steps:

1. First of all, since this dataset is one of the built-in datasets that Torchvision library provides, I downloaded the data using this library. The data was available in two formats of train and test sets. There were 60,000 records in the training set, and 10,000 records in test set. There were 10 different classes as target labels for the records. The size of each input image is 28 * 28 pixels.
2. No especial transform function was performed on the data, since there was no need to augmentation because of size of the dataset, and the quality of dataset. Only one transformation was used in order to convert raw inputs to Torch tensor for ease of computation in future works.
3. Train and test datasets were passed into their own data loader in order to make the data into batches of size 32. For training phase, I used shuffling for each epoch.

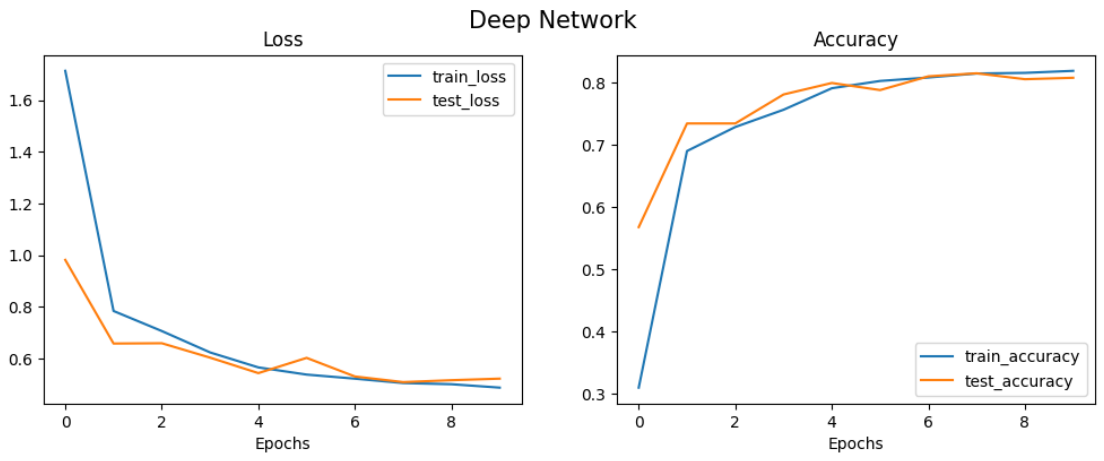
### 2. Expriments

Lots of experiments was performed on the dataset using different architectures of multi-layer perceptron and training techniques which are going to discuss in the following section. I used the cross entropy as the loss function and SGD optimizer with learning rate set to 0.1 (except for the models with regularization) for the defined models. In each hidden layers in these models there are 10 or 50 neurons based on the definition of the model and their performance.

1. Neural Network Depth Effect
   For experimenting this section, I made two different neural networks. First one is the shallow one which consist of 3 linear layers (the last one for classification) with ReLU activation function. Before fitting the data into model, images will convert to a 1-d tensors using flatten layer. The changes in loss and accuracy of the model on both train and test sets are illustrated in the following plot. As it can be seen from the following plot, the slope of decreasing loss is low and test accuracy fluctuates a lot due to simplicity of the model architecture.



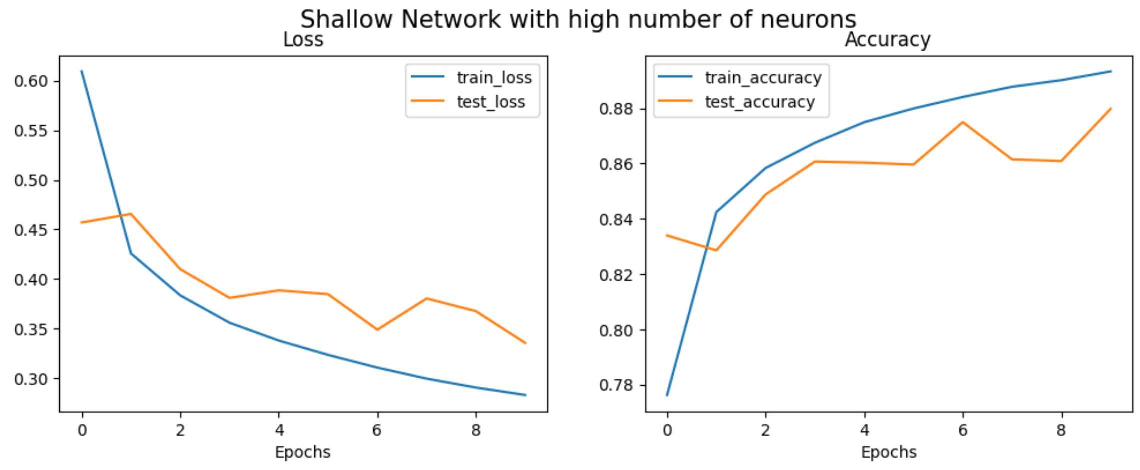
Second model is the deeper one, which consist of 8 linear layers (the last one for classification) with ReLU activation function. Before fitting the data into model, images will convert to a 1-d tensors using flatten layer. The changes in loss and accuracy of the model on both train and test sets are illustrated in the following plot. As it can be seen from the following plot, since the architecture is more complex, network is able to extract more important features from the inputs, as a result, the fluctuation in accuracy decreased and the slope of decreasing loss increased in compare to shallow model.



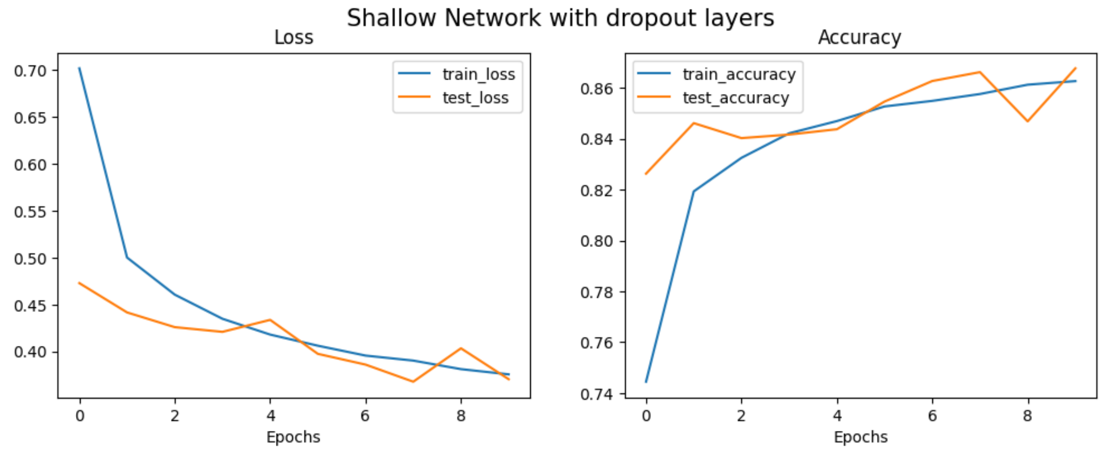
Each model was trained for 10 epochs, and at the end the shallow model has the better accuracy on the test data. The main reason could be the low number of epochs of training or vanishing gradients problem.

2. <u>Effect of number of neurons in hidden layers</u>

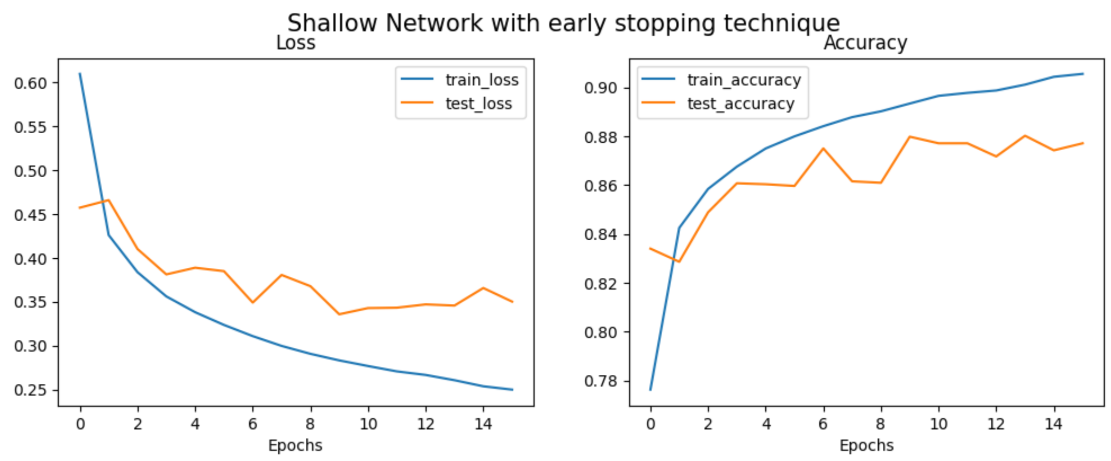
   For this experiment I made a neural network same as the shallow one but in each hidden layer there are 50 neurons instead of 10 neurons which the results are shown in the following plot. As it can be seen, the performance of the model increased significantly, but the fluctuations and low slope are stayed as same as before.



3. <u>Effect of using dropout layers</u>
   For this experiment I made a neural network same as the shallow one but add a dropout layer with 0.2 probability after each activation function except for the final layer which the results are shown in the following plot. As it can be seen, the fluctuations decreased since dropout increase regularization and generalization in our model and loss values are decreased together in different epochs and does not have much difference.



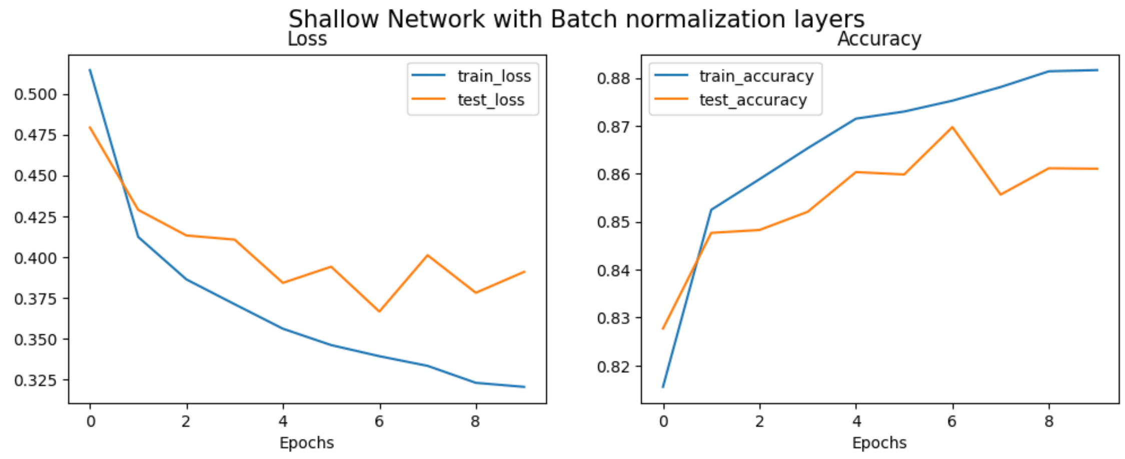
4. <u>Effect of using early stopping</u>
   For this experiment I used a model same as the shallow one but add a stopping criterion in the train function which checks the model performance on test data to decrease at least 0.01 in each epoch, and after 3 epochs of no improvement It will stop the training. Results are shown in the following plot. As it can be concluded from the following plot, the model was starting to overfit to the training data and the stopping criterion has finished the training in order to prevent overfitting. It was trained for 16 epochs.



5. <u>Effect of using Batch Normalization</u>

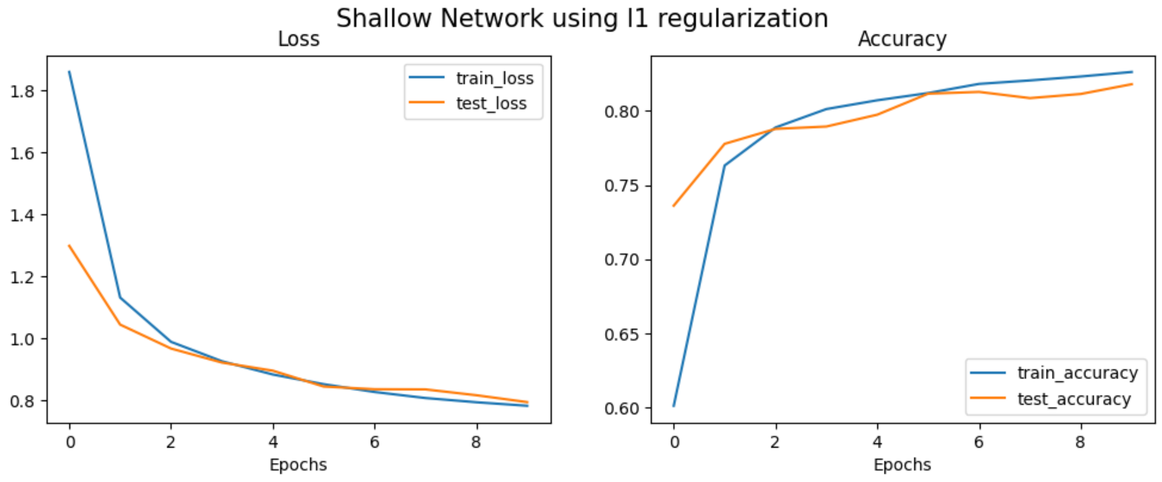
   Batch normalization is a technique for training very deep neural networks that standardizes the inputs to a layer for each mini-batch. This has the effect of stabilizing the learning process and dramatically reducing the number of training epochs required to train deep networks and increase the regularization effect on the model.

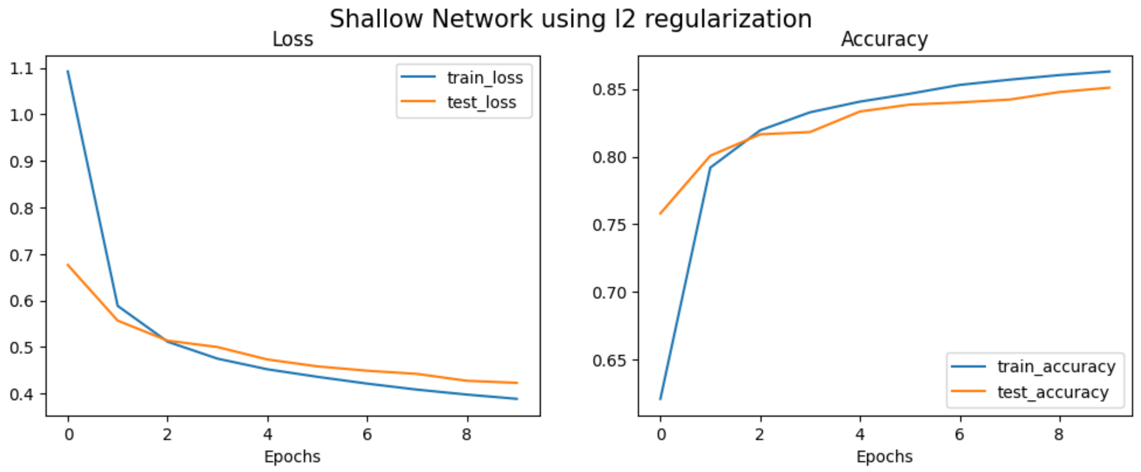
   For this experiment I made a neural network same as the shallow one but add a batch normalization layer after each activation function except for the final layer which the results are shown in the following plot. As it can be seen, this layer has the same performance as using dropout layers. The fluctuations decreased since batch normalization increase regularization and generalization in our model.

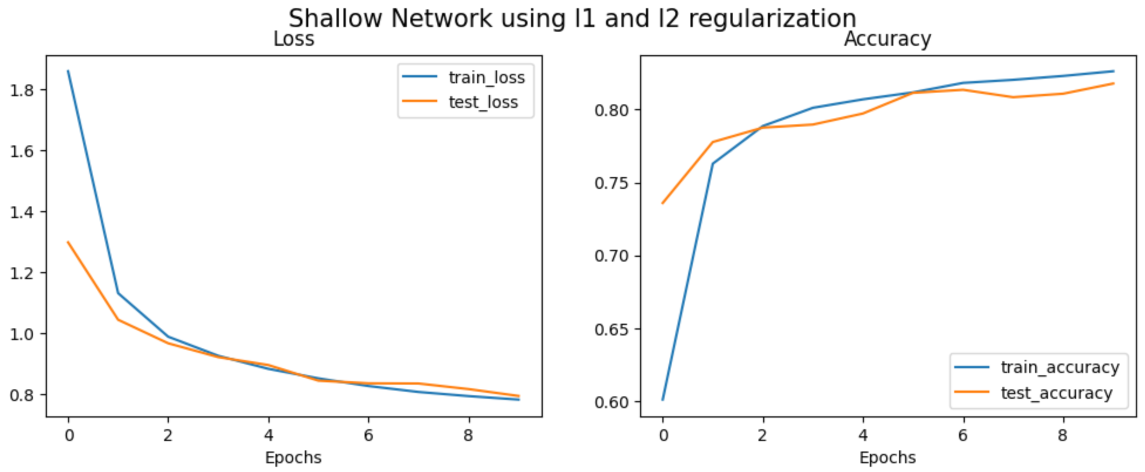


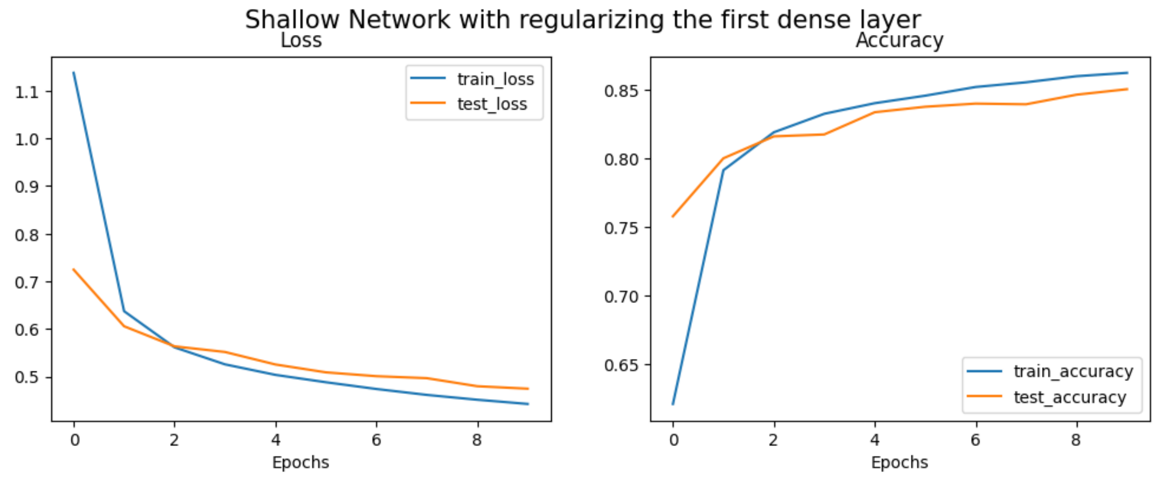
6. <u>Regularization</u>
   For this experiment I made several neural networks same as the shallow one but add a regularization term into the cross-entropy loss function. For the first and second one, I used l1 and l2 regularization which was described in Theory section and these terms to the loss function. For the third model, I used l1 and l2 regularization simultaneously. For the final model I added the l2 norm of the first linear layer in the model to the original loss. which results are shown in the following plots. As it can be seen from the following plots, all of these methods decrease the fluctuations in the plots and have the significant regularization

and generalization effect on the model in a way that loss values are decreased together in different epochs and does not have much difference.

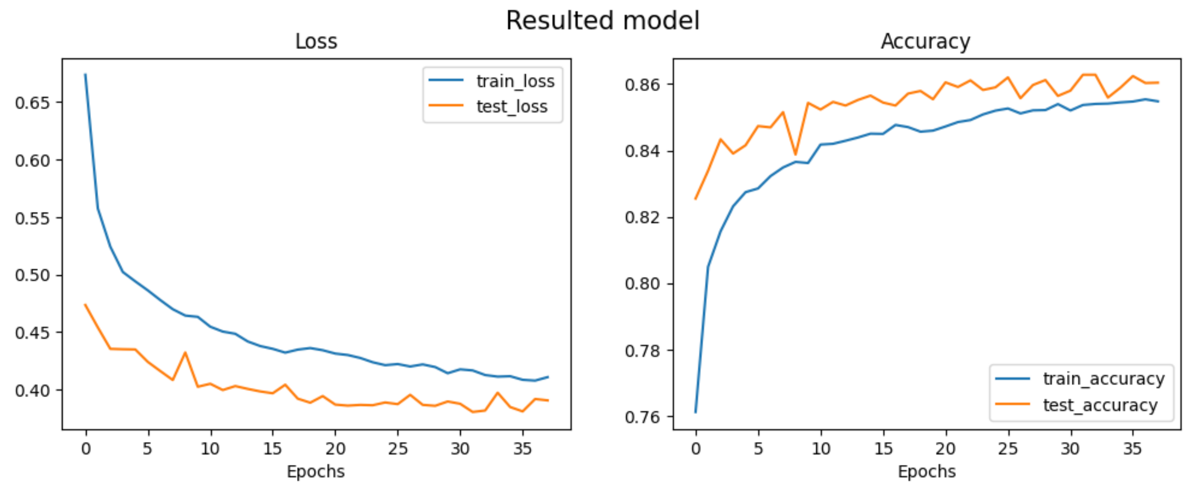






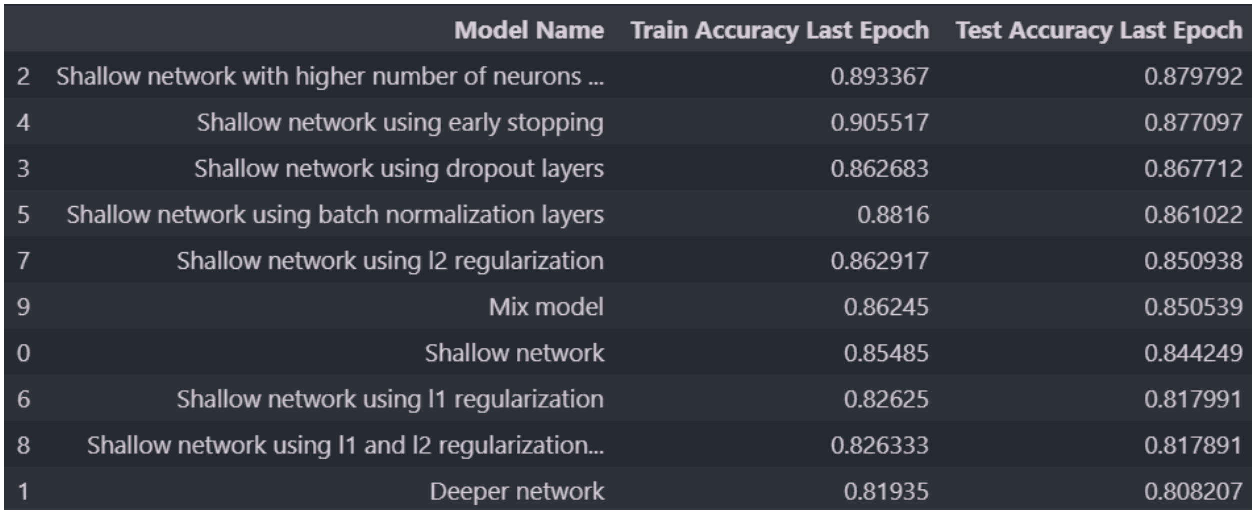


7. <u>Mixing all together</u>
   For this experiment I made a neural network with mix of methods that leads to a good result from the above experiments. Model consists of 4 linear layer (last one for classification) and ReLU activation layers after each one of them (except the last one). Also, after each activation layer, there is a batch normalization and dropout layer respectively in order to prevent overfitting and regularization (however batch normalization is enough, since it does the good regularization effect, but dropout increase training time). Moreover, I used the same stopping criterion which was mentioned above. Model was trained for 38 epochs. results are shown in the following plot. As it can be seen, model is performing good on the data, the decreasing slope of loss values is high and there is not much fluctuation in the values which means the model was regularized pretty well and there is no sign of overfitting. Model is performing better on test data than train data.



## 3. Results

Results of above experiments are aggregated in the following table and sorted based on the performance of the model on test data. I used the performance of the model on train and test data for comparison.



| | Model Name | Train Accuracy Last Epoch | Test Accuracy Last Epoch |
|---|---|---|---|
| 2 | Shallow network with higher number of neurons ... | 0.893367 | 0.879792 |
| 4 | Shallow network using early stopping | 0.905517 | 0.877097 |
| 3 | Shallow network using dropout layers | 0.862683 | 0.867712 |
| 5 | Shallow network using batch normalization layers | 0.8816 | 0.861022 |
| 7 | Shallow network using l2 regularization | 0.862917 | 0.850938 |
| 9 | Mix model | 0.86245 | 0.850539 |
| 0 | Shallow network | 0.85485 | 0.844249 |
| 6 | Shallow network using l1 regularization | 0.82625 | 0.817991 |
| 8 | Shallow network using l1 and l2 regularization... | 0.826333 | 0.817891 |
| 1 | Deeper network | 0.81935 | 0.808207 |

Its worth mentioning that all the models except for the ones that has the stopping criteria have trained for 10 epochs. As it can be concluded from the above table, the shallow network with 3 linear layer and 50 neurons in each hidden layer which was trained for 10 epochs has the best performance among the others; whilst, the deepest model which has 8 linear layer and 10 neurons in its hidden layers has the lowest performance among the others.