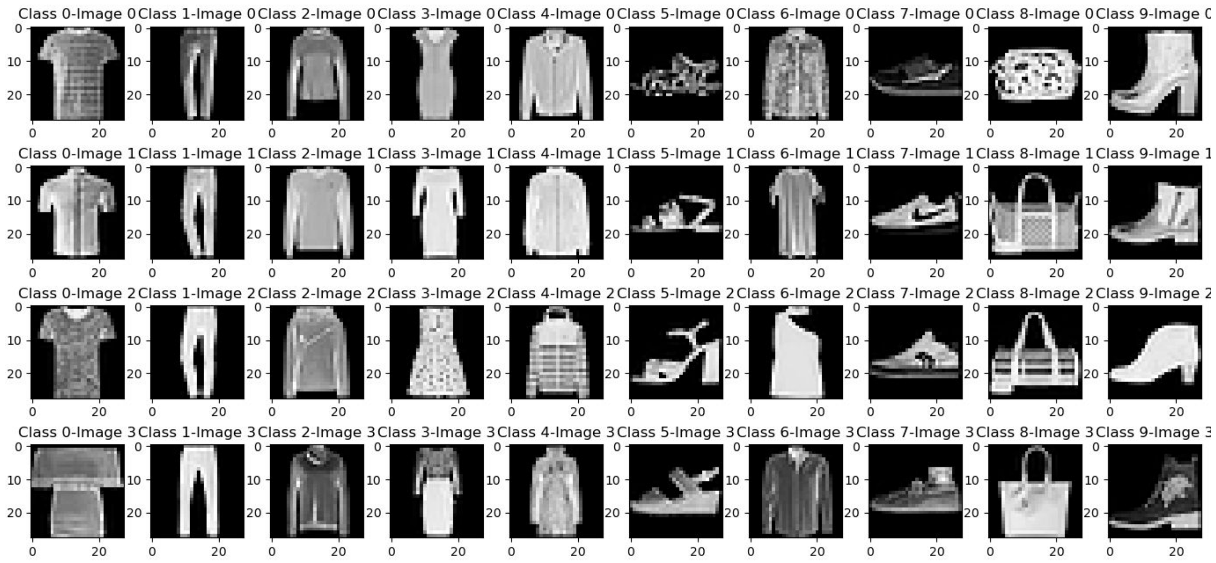
Coby Penso. 208254128

Part 1 - Visualize the data:

Visualizing the data, for every label visualize 4 images:



Part 2 - Logistic Regression Classifier:

The model implemented in the **Ex2_LRClassifier.py**
I have tested many options of hyper parameters, using a grid search - for every combination of hyperparameters train the model for 500 epochs and save to a file the accuracy achieved.

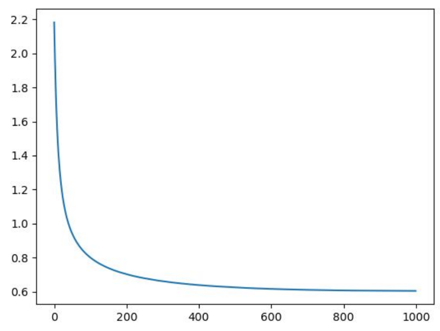
Steps:

1. Grid search of combinations of hyperparameters
2. Save to a CSV file the table of results from step 1
3. training the best model found in step 1
4. Plot the accuracy and loss progress

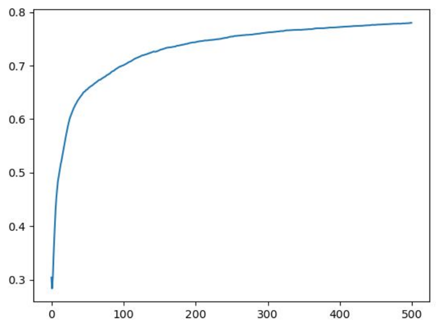
Hyperparameters results in the file - **LRC_Summary.csv:**
main insights: for lower Lambda coeff (small regularization) the results are better for training and validation images, probably because of the nature of the dataset the fear of overfitting is less valid in our case.
Also, bigger batch size leads for better results, i.e bigger batch size makes the training loss more accurate and robust.
In addition, lower learning rates reach more stably to the minimum and lead for better results.

The results progress with the best model chosen from step 1:

Loss:



Accuracy:



Part 3a - Deriving the Gradients of a NN with a Softmax Activation

I have calculated gradients of the NN with a softmax activation and ReLU as a non linearity. Those calculations used in the back propagation in the training phase

Part 3b - Neural Network with One Hidden Layer

The model implemented in the **Ex2_NN.py**
I have tested many options of hyper parameters, using a grid search - for every combination of hyperparameters train the model for 500 epochs and save to a file the accuracy achieved.

Steps:

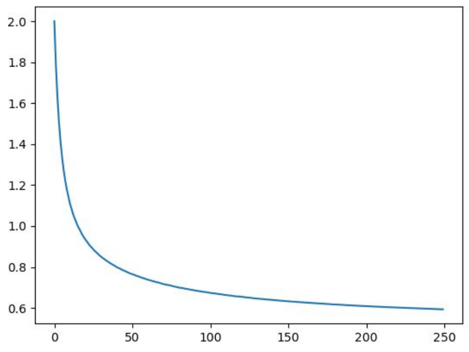
1. Grid search of combinations of hyperparameters
2. Save to a CSV file the table of results from step 1
3. training the best model found in step 1
4. Plot the accuracy and loss progress

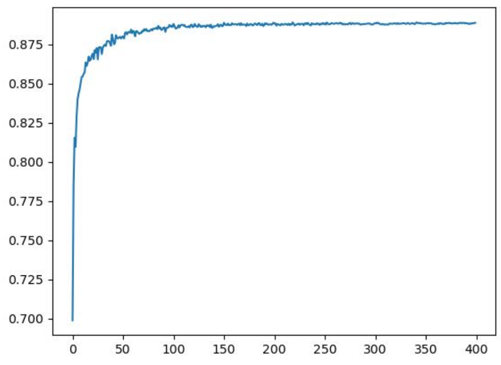
Hyperparameters results in the file - **NN_Summary.csv:**
main insights:

- For a bigger hidden layer the results improve, that's because the expression power for the same amount of training is bigger.
- Smaller lambda achieve better results.
- Bigger batch sizes achieve better results.
- Decaying learning rate achieve more stable results

The results progress with the best model chosen from step 1:

Loss: **Accuracy:**





Predictions:

For part 2 and part 3b, the best model has been chosen, and predictions have been made to the test data.
The predictions saved in the CSV files:
lr_pred.csv & **NN_pred.csv**

Also few options of activation functions examined:

- ReLU
- Tanh
- Sigmoid

Main Insights:

- For every activation function there was a different best hyperparameter, i.e in order to find the real best hyperparams for best accuracy the activation function should be considered as one.
- The best activation function for my grid search was ReLU.

Usage:

In order to use the models trained and predict the labels on new test data, the following step should apply:

1. The scripts should be of “Predict” mode. this mode is the default mode, so no active action should be taken before run
2. The model saved into a file - `"LRC_model.pkl" or "NN_model.pkl"` It is important that this file would be in the same dir as the script to be run.
3. The predictions will be saved to a file - `"LRC_test.csv" or "NN_test.csv"`
4. Env requirements:
    a. Numpy
    b. pandas
    c. argparse
    d. matplotlib
    e. pickle(for model save/upload to/from file)