ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ
ATHENS UNIVERSITY OF ECONOMICS AND BUSINESS

Anastopoulos Konstantinos, Tsaklidou Sofia

Fashion item recognition

Supervisor: Dr. Malakasiotis

2022

The code of the assignment can be found here:
MLP:https://colab.research.google.com/drive/1yMBRXTy7l46tqW35pxtrWSpoi8xeCv1h?usp=sharing
CNN: https://colab.research.google.com/drive/1_qhWs1rRbqkwCLk-y3Ic7E-K-a30CBUd?usp=sharing

The objective of this project build a deep learning model that recognizes the fashion item using 2 different architectures, one with MLPs and one with CNNs. The data used for the project is the Fashion-MNIST, found here. The dataset is also available from Tensorflow and Keras.

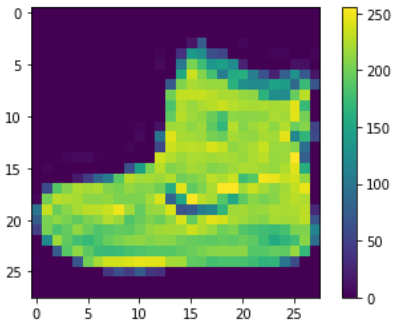
Fashion-MNIST is a Zalando article image dataset that includes a training set of 60,000 samples and a test set of 10,000 examples. Each sample is a 28x28 grayscale picture with a label from 10 classes. Zalando intends Fashion-MNIST to serve as a direct drop-in replacement for the original MNIST dataset for benchmarking machine learning algorithms. It shares the same image size and structure of training and testing splits.

1 Exploratory Data Analysis

Here is a brief analysis of the exploratory data to get a sense of what the data mean.

```
Fashion MNIST train -  rows: 60000  columns: 28
Fashion MNIST test -  rows: 10000  columns: 28
```

We observe that, as already mentioned above, there exist 60000 training examples and 10000 test examples. Each image is 28 pixels in height and 28 pixels in width, for a total of 784 pixels in total.



Inspecting the image of the training set we can conclude that the pixel values fall in the range of 0 to 255.

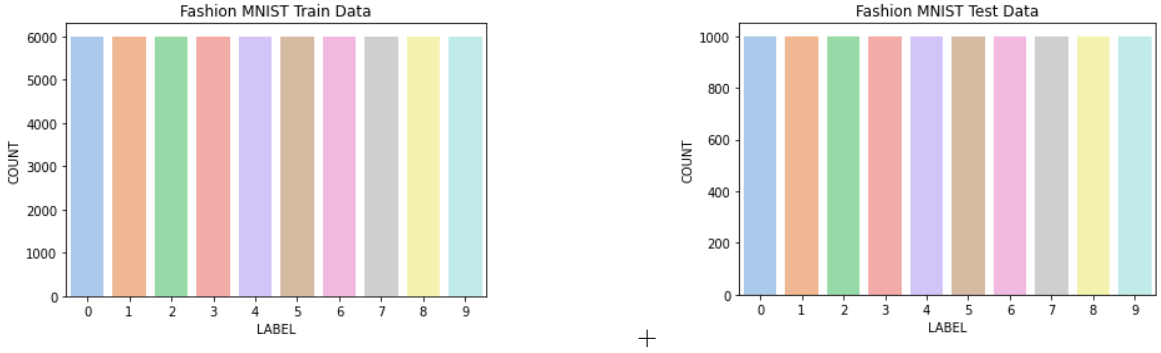
This means that doing the preprocessing we will scale these values to a range of 0 to 1 before feeding them to the neural network model..

We already mentioned that there are 10 classes in this dataset, represented as unique integers, which are shown below:

- T-shirt/top
- Trouser
- Pullover
- Dress
- Coat
- Sandal
- Shirt
- Ankle boot

Then, it is important also to check the balance of the data in both train and test data set.
As a next step, we will plot some images to get an idea.

Figure 1: Dataset images and their class labels



So in both cases, train and test datasets, we observe that the distribution of classes are uniformly distributed. Thus, the dataset does not suffer from a class imbalance.

Data Preprocessing

As a first step we are going to convert train and the test datasets to from integers to floats. Then, we will reshape our data -the images- in 3 dimensions. In the sequence, we normalize the data, in other words we divide the values by 255. It's important that the training set and the testing set be preprocessed in the same way.
In addition, we mentioned that there are 10 classes -labels- which means that we use a one hot encoding for the class element of each sample, transforming the integer into a 10 element binary vector with a 1 for the index of the class value.

Architectural Choices

The scope of this assignment is to focus on 2 different architectures, one with MLPs and one with CNNs.

2 MLP

A Multilayer Perceptron (MLP) is a class of feedforward artificial neural network (ANN). In order to find which is the best MLP architecture that fits our problem we divided it into two approaches based

on the number of hidden layers,we used 2 and 3 hidden layers respectively, each model has and then we tuned some of its parameters. For the hidden layers ReLu was used as an activation function and Softmax was used in our output layers since we have a multiclass classification problem. Also, Dropout of 0.2 after each hiddel layers was used. The parameters that we tuned were:

- Batch Size where we tried 128, 256 and 512.
- Optimizer, Adam and SGD.
- Various Learning Rates for each optimizer.

We used early stopping with patience of 10 in both Hyperparameter tuning and in our final model training.

Model 1 using 2 hidden layers

After testing all of our parameters we had the below results:

Optimizer	Learning_Rate	Batch_Size	Train_Loss	Val_Loss	Test_Loss	Train_Accuracy	Val_Accuracy	Test_Accuracy
Adam	0.0010	512	0.171049	0.310474	0.307544	0.934296	0.897167	0.8951
Adam	0.0010	256	0.161893	0.338903	0.336052	0.937574	0.894000	0.8937
Adam	0.0010	128	0.159727	0.338039	0.340994	0.938944	0.897500	0.8929
Adam	0.0001	128	0.170257	0.300050	0.315331	0.937944	0.896167	0.8929
SGD	0.1000	128	0.194522	0.297791	0.313302	0.926704	0.893500	0.8913
Adam	0.0001	512	0.192273	0.293183	0.312733	0.930333	0.895667	0.8903
Adam	0.0001	256	0.177528	0.296477	0.311040	0.934444	0.894833	0.8901
SGD	0.1000	512	0.195646	0.298253	0.313399	0.928037	0.893833	0.8896
SGD	0.1000	256	0.228646	0.304168	0.316641	0.916222	0.884833	0.8879
SGD	0.0100	128	0.274412	0.306335	0.332232	0.901648	0.885167	0.8801
SGD	0.0100	256	0.326920	0.331225	0.360505	0.883204	0.876667	0.8706
SGD	0.0100	512	0.384726	0.367260	0.392202	0.863926	0.866167	0.8577

Based on that we selected the model with Batch Size = 512 and Adam with learning rate 0.001 as our optimizer.
We trained our best model with the following architecture:

```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 Input (InputLayer)          [(None, 784)]             0

 Hidden-1 (Dense)            (None, 256)               200960

 Dropout-1 (Dropout)         (None, 256)               0

 Hidden-2 (Dense)            (None, 256)               65792

 Dropout-2 (Dropout)         (None, 256)               0

 Output (Dense)              (None, 10)                2570

=================================================================
Total params: 269,322
Trainable params: 269,322
Non-trainable params: 0
_________________________________________________________________

Test Loss      : 0.31527
Test Accuracy  : 0.89010
```

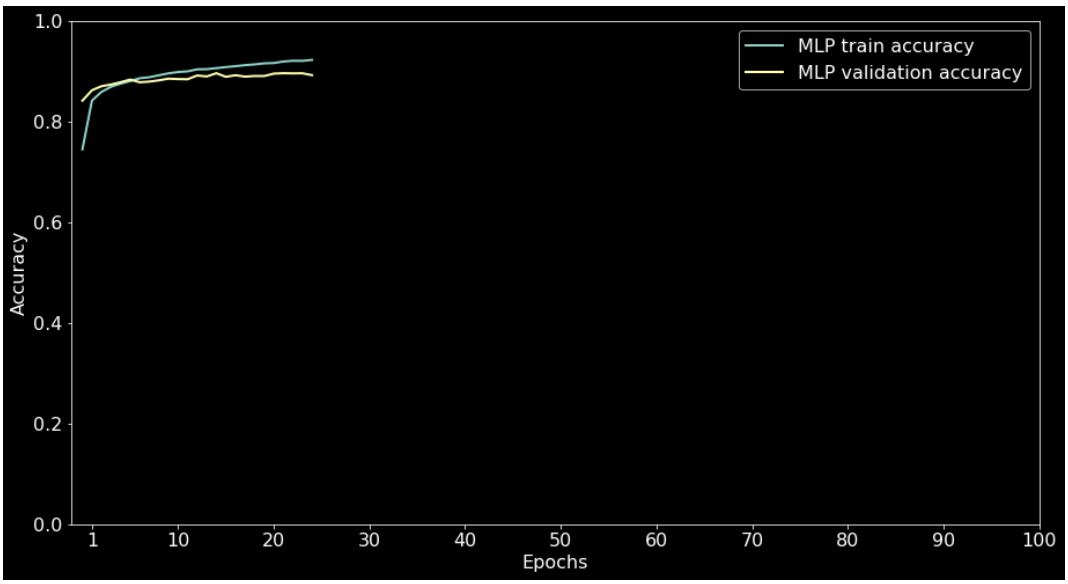


Figure 2:

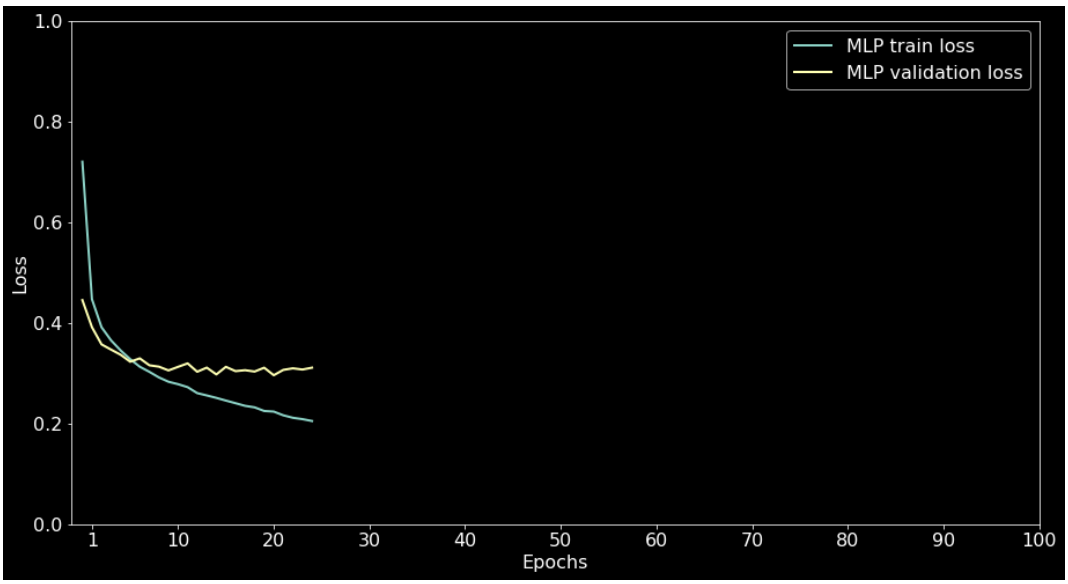


Figure 3:

Model 2 using 3 hidden layers

After testing all of our parameters we had the below results:

Optimizer	Learning_Rate	Batch_Size	Train_Loss	Val_Loss	Test_Loss	Train_Accuracy	Val_Accuracy	Test_Accuracy
Adam	0.0001	128	0.122518	0.354204	0.353139	0.952944	0.897833	0.8966
Adam	0.0010	512	0.184589	0.317770	0.322340	0.930963	0.893833	0.8940
Adam	0.0010	128	0.199642	0.319874	0.318674	0.924148	0.893833	0.8932
Adam	0.0001	256	0.164193	0.324472	0.328093	0.939241	0.894167	0.8911
SGD	0.1000	128	0.188838	0.332262	0.315868	0.928389	0.893000	0.8908
Adam	0.0001	512	0.191818	0.307590	0.321397	0.928444	0.893333	0.8906
SGD	0.1000	256	0.184645	0.341030	0.322307	0.929500	0.891333	0.8905
Adam	0.0010	256	0.179636	0.339120	0.346552	0.932093	0.891000	0.8893
SGD	0.1000	512	0.210672	0.339953	0.320931	0.920889	0.887333	0.8873
SGD	0.0100	128	0.248173	0.308503	0.328898	0.908574	0.890333	0.8861
SGD	0.0100	256	0.312624	0.321045	0.347149	0.887852	0.881333	0.8751
SGD	0.0100	512	0.374955	0.354171	0.381499	0.866593	0.870333	0.8641

Based on that we selected the model with Batch Size = 128 and Adam with learning rate 0.0001 as our optimizer.
We trained our best model with the following architecture:

```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 Input (InputLayer)          [(None, 784)]             0

 Hidden-1 (Dense)            (None, 256)               200960

 Dropout-1 (Dropout)         (None, 256)               0

 Hidden-2 (Dense)            (None, 256)               65792

 Dropout-2 (Dropout)         (None, 256)               0

 Hidden-3 (Dense)            (None, 256)               65792

 Dropout-3 (Dropout)         (None, 256)               0

 Output (Dense)              (None, 10)                2570

=================================================================
Total params: 335,114
Trainable params: 335,114
Non-trainable params: 0
_________________________________________________________________

Test Loss      : 0.32351
Test Accuracy  : 0.89220
```

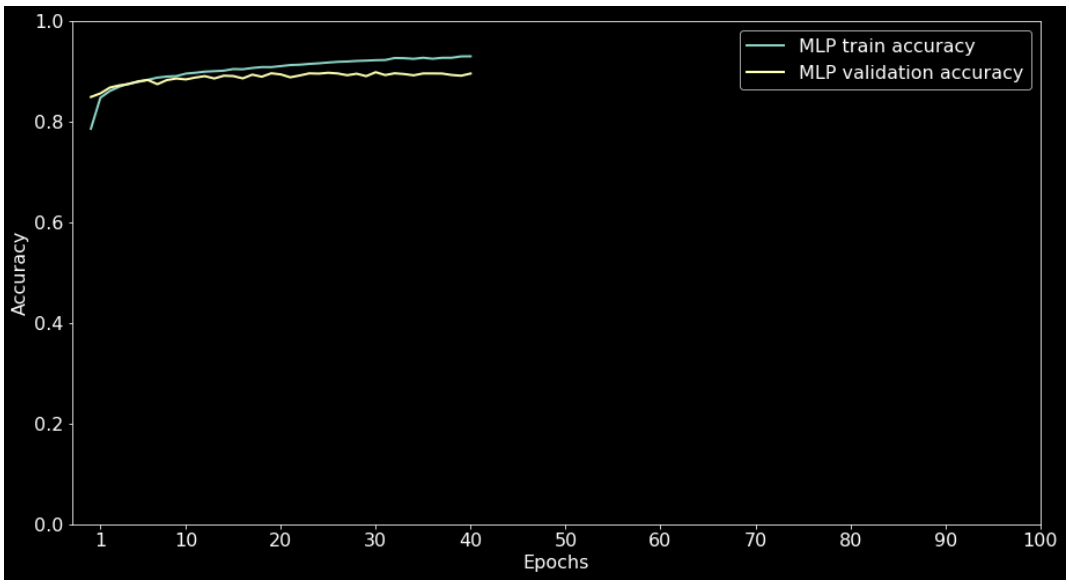


Figure 4:

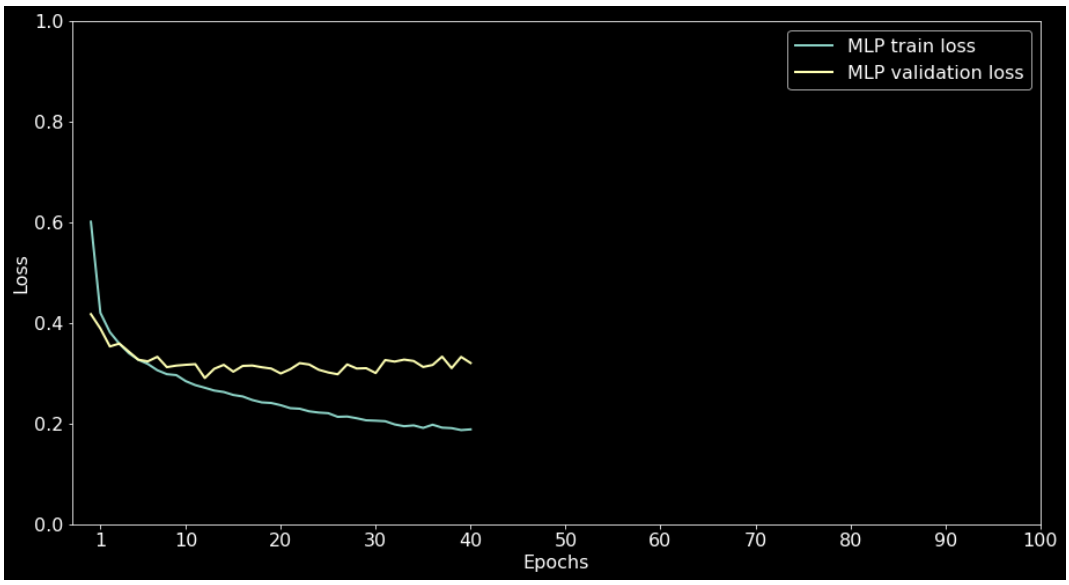


Figure 5:

Given the results of our two best models we would probably choose the one with the 2 hidden layers due to the smaller loss in our Test set. Accuracy on both was relatively close.

3 CNN

A convolutional neural network (CNN) is a type of artificial neural network used in image recognition and processing that is specifically designed to process pixel data.
Having implemented the MLP architecture we will proceed with the creation of 3 CNNs with the following architectures / characteristics:

Model 1 using SGD optimizer without dropout layers

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 26, 26, 32)        320

 max_pooling2d (MaxPooling2D  (None, 13, 13, 32)       0
 )

 conv2d_1 (Conv2D)           (None, 11, 11, 64)        18496

 max_pooling2d_1 (MaxPooling  (None, 5, 5, 64)         0
 2D)

 conv2d_2 (Conv2D)           (None, 3, 3, 128)         73856

 flatten (Flatten)           (None, 1152)              0

 dense (Dense)               (None, 128)               147584

 dense_1 (Dense)             (None, 10)                1290

=================================================================
Total params: 241,546
Trainable params: 241,546
Non-trainable params: 0
_________________________________________________________________

Test loss: 0.34615159034729004
Test accuracy: 0.9125000238418579
```

To begin with, regarding model 1 we have used Early Stopping in order to stop training when a monitored metric has stopped improving.
Next, we defined a baseline convolutional neural network model for the problem using the SGD optimizer without dropout layers. So we start with a convolutional layer with a small filter size (3,3) and a modest number of filters (32) followed by a max pooling layer. Then we add a second convolutional layer with a small filter size (3,3) and a modest number of filters (64) followed by a max pooling layer. Then, we add the last convolutional layer with a small filter size (3,3) and a modest number of filters (128) followed by flatten layer, which basically converts matrics to array, and flatten the image in order to proceed to the classification block.
The Dence layer does classification and it is deep neural network. Last but not least, we add an output layer which has softmax activation.
Since we are using a use softmax activation in output layer, we have used *categorical_crossentropy* as a fit optimizer.

We observe that we got *Test accuracy: 0.9125*, which is not bad, but we are going to improve our model adding dropout layers.

Model 2 using SGD optimizer with dropout layers

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 26, 26, 32)        320

 max_pooling2d (MaxPooling2D  (None, 13, 13, 32)       0
```

```
 )

 dropout (Dropout)           (None, 13, 13, 32)        0

 conv2d_1 (Conv2D)           (None, 11, 11, 64)        18496

 max_pooling2d_1 (MaxPooling  (None, 5, 5, 64)         0
 2D)

 dropout_1 (Dropout)         (None, 5, 5, 64)          0

 conv2d_2 (Conv2D)           (None, 3, 3, 128)         73856

 dropout_2 (Dropout)         (None, 3, 3, 128)         0

 flatten (Flatten)           (None, 1152)              0

 dense (Dense)               (None, 128)               147584

 dropout_3 (Dropout)         (None, 128)               0

 dense_1 (Dense)             (None, 10)                1290

=================================================================
Total params: 241,546
Trainable params: 241,546
Non-trainable params: 0
_________________________________________________________________

Test loss: 0.24628889560699463
Test accuracy: 0.9106000065803528
```

Regarding the second model, we used the same architecture with model 1 with the difference of adding dropouts layers. Here we observe a slight decrease when it comes to *Test accuracy* which in this case is 0.9106.

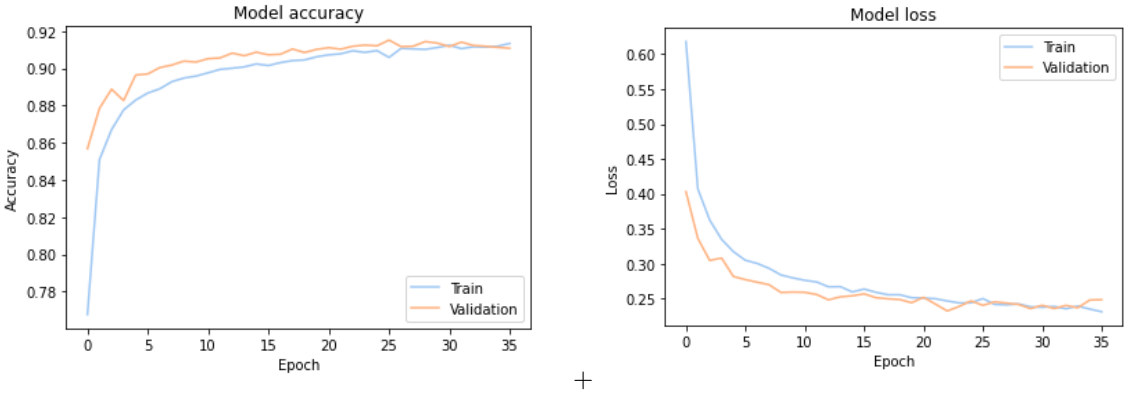
As next step, we are going to try an other optimizer, Adam instead of SGD, having the dropout layers in order to chech if there will be any improvement in the model.

Model 3 using Adam optimizer with dropout layers

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 26, 26, 32)        320

 max_pooling2d (MaxPooling2D  (None, 13, 13, 32)       0
 )

 dropout (Dropout)           (None, 13, 13, 32)        0

 conv2d_1 (Conv2D)           (None, 11, 11, 64)        18496

 max_pooling2d_1 (MaxPooling  (None, 5, 5, 64)         0
 2D)
```

```
dropout_1 (Dropout)          (None, 5, 5, 64)          0

conv2d_2 (Conv2D)            (None, 3, 3, 128)         73856

dropout_2 (Dropout)          (None, 3, 3, 128)         0

flatten (Flatten)            (None, 1152)              0

dense (Dense)                (None, 128)               147584

dropout_3 (Dropout)          (None, 128)               0

dense_1 (Dense)              (None, 10)                1290

=================================================================
Total params: 241,546
Trainable params: 241,546
Non-trainable params: 0
_________________________________________________________________

Test loss: 0.24042096734046936
Test accuracy: 0.9151999950408936
```

We observe that using the last model with Adam optimiser adding dropout layers to the model we got the best results (Test loss 0.24 and Test accuracy 0.9152). This means that we are going to continue with some predictios in order to see how the model works.



Predictions

At this point, we are going to plot the first 30 test images, their predicted labels, and the true labels. The color of the correct predictions are in blue whereas the incorrect predictions are in red.
Overall

Figure 6: Plot of 30 first test images, their predicted labels, and the true labels.