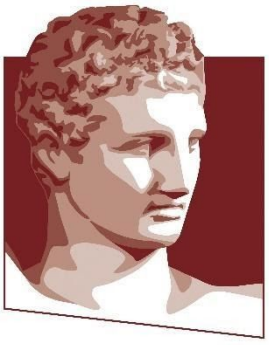


Deep Learning

Project 1 - MLPs and CNN for image classification

Prodromos Malakasiotis - professor (rullller@aueb.gr)

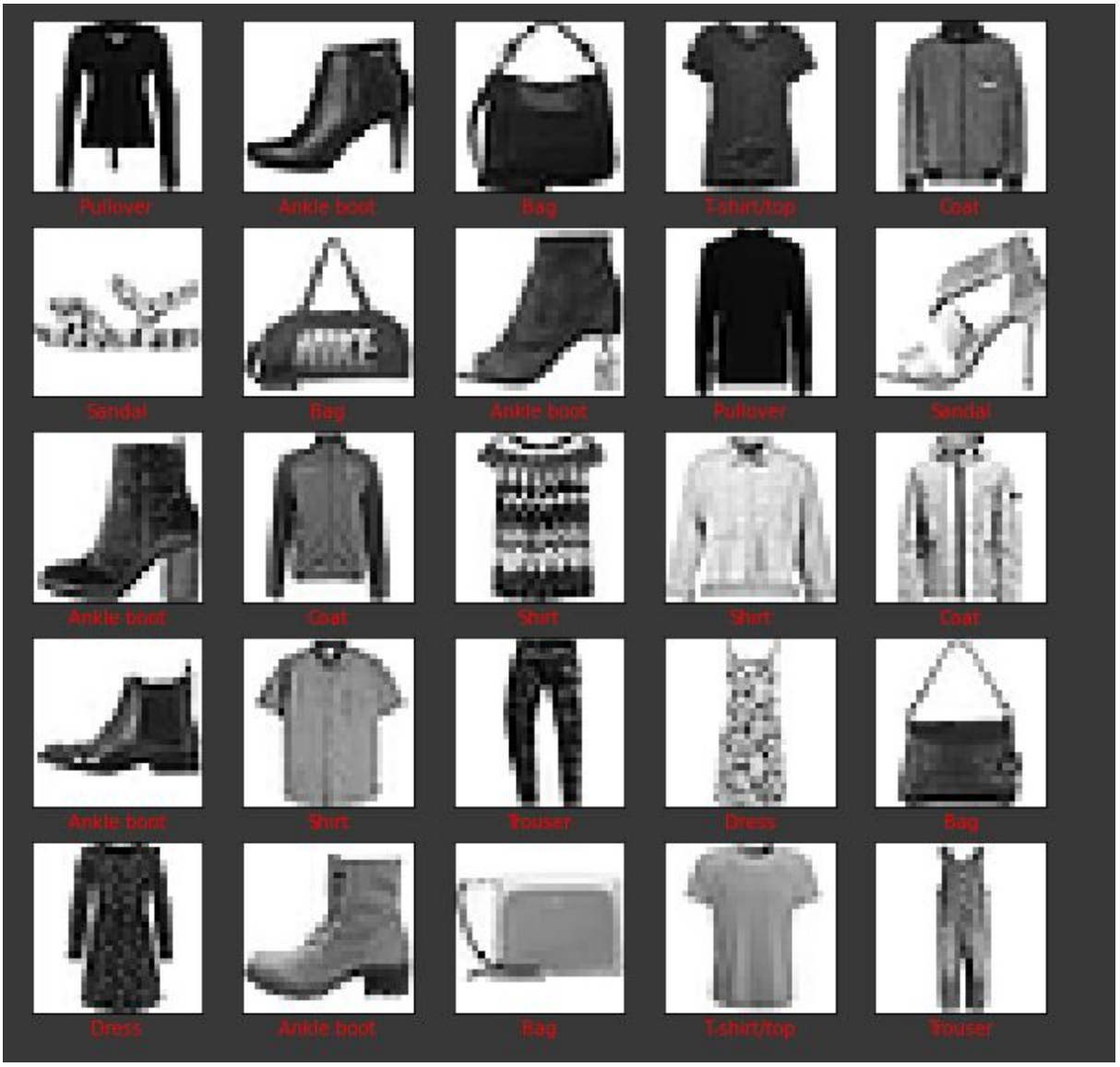
Panagiotis Rallis
ID:P3351816

This project describes some deep learning architectures which can be used to recognize and classify a fashion item to the appropriate category, using the fashion mnist dataset.

There are 10 different categories:

0 T-shirt/top
1 Trouser
2 Pullover
3 Dress
4 Coat
5 Sandal
6 Shirt
7 Sneaker
8 Bag
9 Ankle boot

and the images are depicted in greyscale colour.



Fashion item recognition using Multilayer Perceptron(MLP)

Our problem is related with the recognition of a fashion item using an image. Therefore we deal with a classification problem which only one result can be correct, among 10 categories. In order to conclude in one result, taking the observation with the higher probability, we can use a RELU activation function on hidden layers, to take values > 0 and a softmax activation function for the output, in order to transform this values between 0 and 1 as probabilities. Furthermore, while only one result can be correct, we choose the categorical cross entropy as the loss function.

A simple logistic regression without hidden layer achieves a decent result:

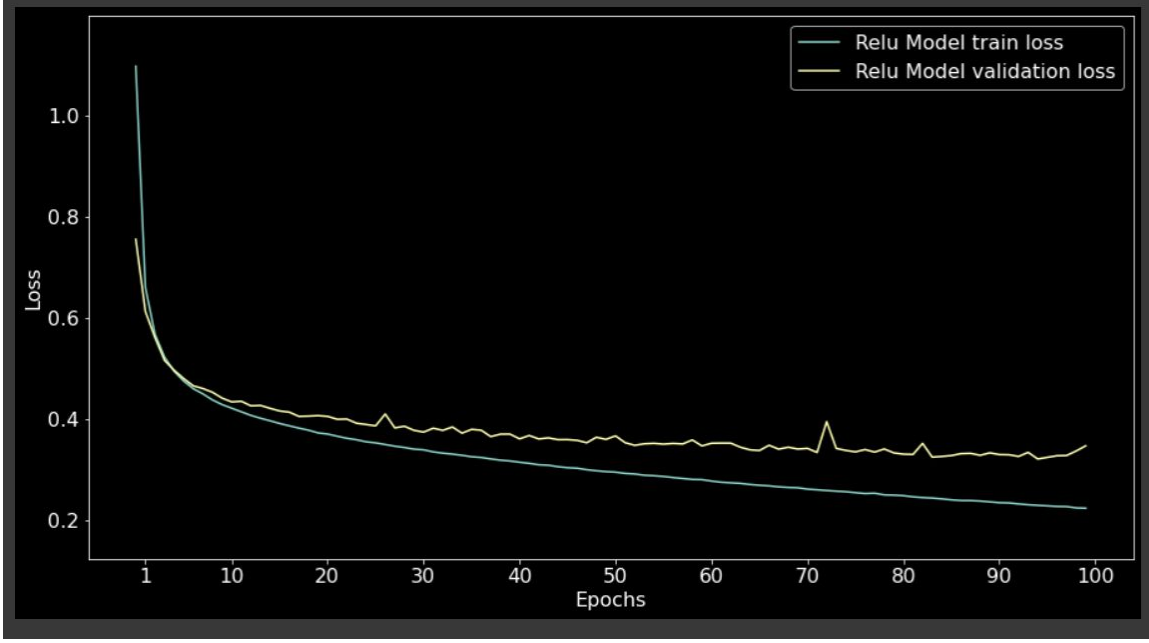
```
Train accuracy     : 0.83913
Validation accuracy: 0.83417
Test accuracy      : 0.82520
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
flatten (Flatten)            (None, 784)               0
_________________________________________________________________
dense (Dense)                (None, 10)                7850
=================================================================
Total params: 7,850
Trainable params: 7,850
Non-trainable params: 0
_________________________________________________________________
Finished training.
------------------
313/313 [==============================] - 0s 1ms/step - loss: 0.5134 - accuracy: 0.8252
```

Performing manual tuning of various different MLPs, the below model achieves a more accurate score at validation data, in 100 epochs.

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
flatten (Flatten)            (None, 784)               0
_________________________________________________________________
dense (Dense)                (None, 256)               200960
_________________________________________________________________
dense_1 (Dense)              (None, 128)               32896
_________________________________________________________________
dense_2 (Dense)              (None, 10)                1290
=================================================================
Total params: 235,146
Trainable params: 235,146
Non-trainable params: 0
_________________________________________________________________
Finished training.
------------------
313/313 [==============================] - 0s 1ms/step - loss: 0.3541 - accuracy: 0.8740
```

```
Train loss categorical crossentropy: 0.22373
Validation loss categorical crossentropy: 0.34704
Test loss categorical crossentropy: 0.35405
```



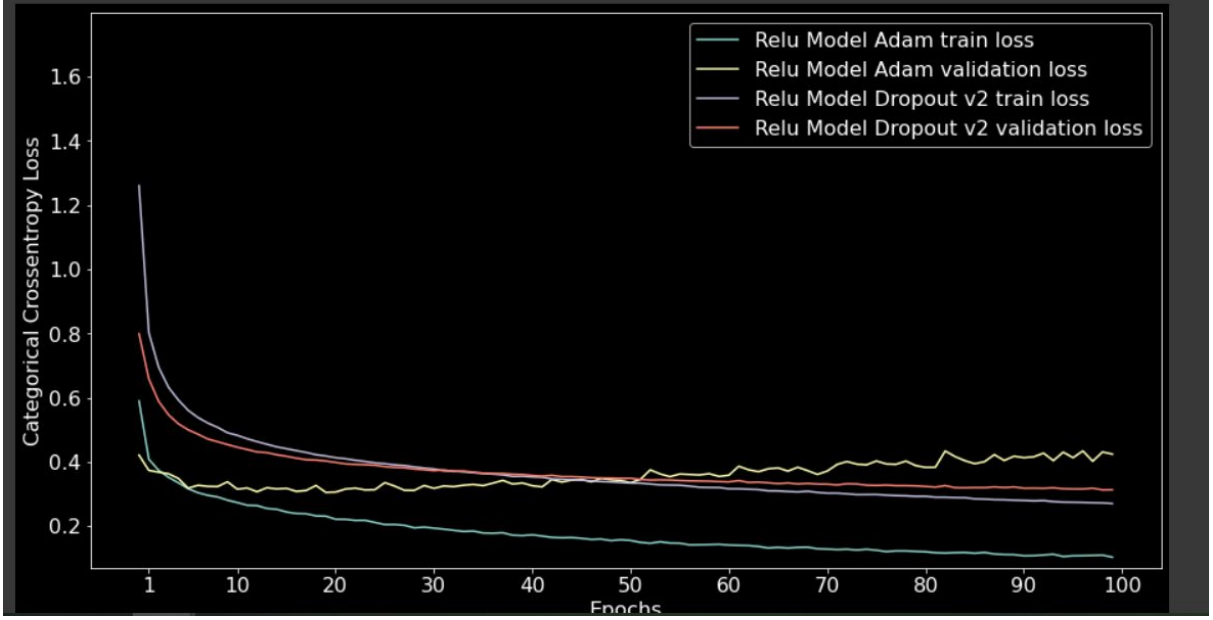

```
Train accuracy     : 0.92006
Validation accuracy: 0.88000
Test accuracy      : 0.87400
```

As we can observe from the above plots, while the epochs increases, the distance between train and validation loss is increased. This is a sign of overfitting which we have to deal with using a dropout method. Also, there is already achieved a good learning rate in both models.

Applying an adaptive learning rate method for faster converge, the loss decreases faster but there is a significant increase of overfitting.

As aforementioned above, a **dropout method** is applied in order to decrease overfitting. Below it is represented an MLP of 2 hidden layers using dropout and **SGD optimizer,** in comparison with the same MLP using adaptive learning method (**Adam Optimizer)** and dropout method.

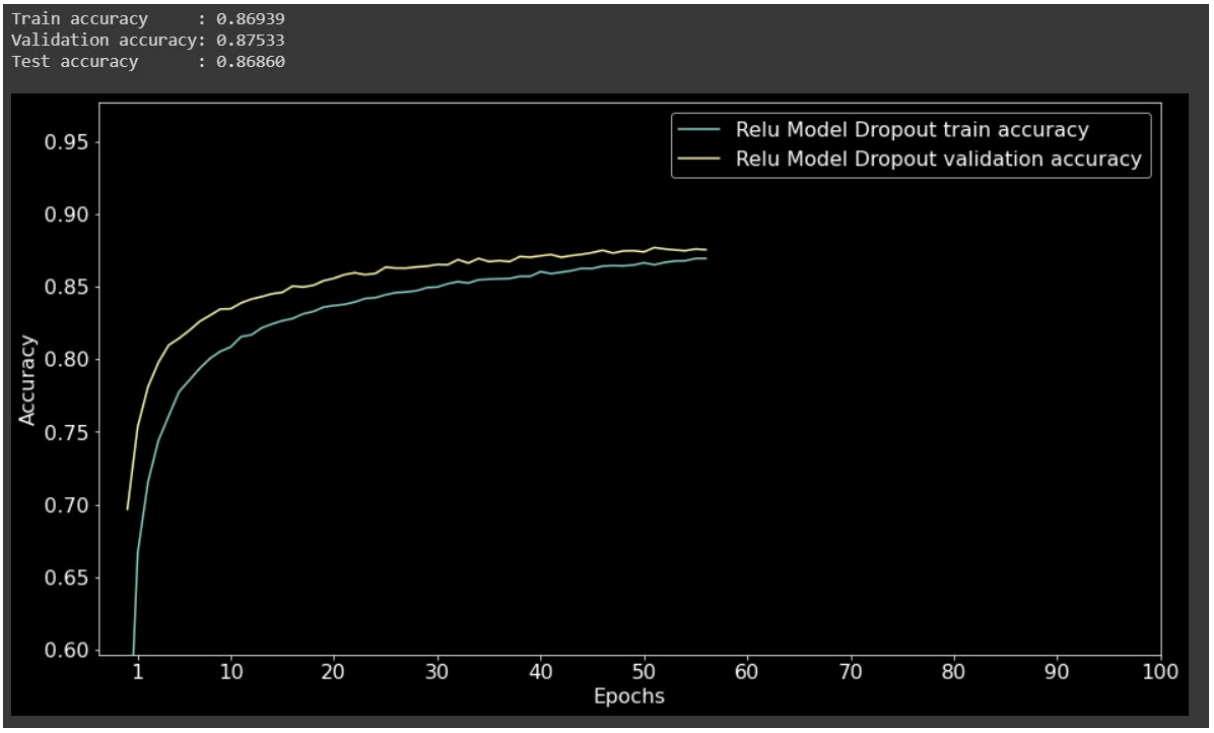
```
Train loss categorical crossentropy: 0.10312
Validation loss categorical crossentropy: 0.42403
Test loss categorical crossentropy: 0.45762
---
Train loss categorical crossentropy: 0.27017
Validation loss categorical crossentropy: 0.31309
Test loss categorical crossentropy: 0.32250
```




```
Train accuracy     : 0.95970
Validation accuracy: 0.90000
Test accuracy      : 0.89340
---
Train accuracy     : 0.90243
Validation accuracy: 0.88833
Test accuracy      : 0.88590
```

It is clear that using only the dropout method with SGD optimizer, the train and validation loss are close enough. Therefore the overfitting is decreased. Also the accuracy of model using only dropout is increased to 0.8859 in test data. However, the model using dropout and Adam overfits in training data and the validation loss increases during the epochs which is something really unwelcome. To conclude, when it comes to Multilayer Perceptrons, we choose the model with dropout and SGD optimizer as the most suitable model with a fair score and without overfitting signs.

Finally, in order to avoid unnecessary runs to epochs, without any change of validation accuracy, we apply an early stopping method using Dropout and SGD optimizer with patient 5 achieving the below results in epoch 57.



Convolutional Neural Networks(CNN)

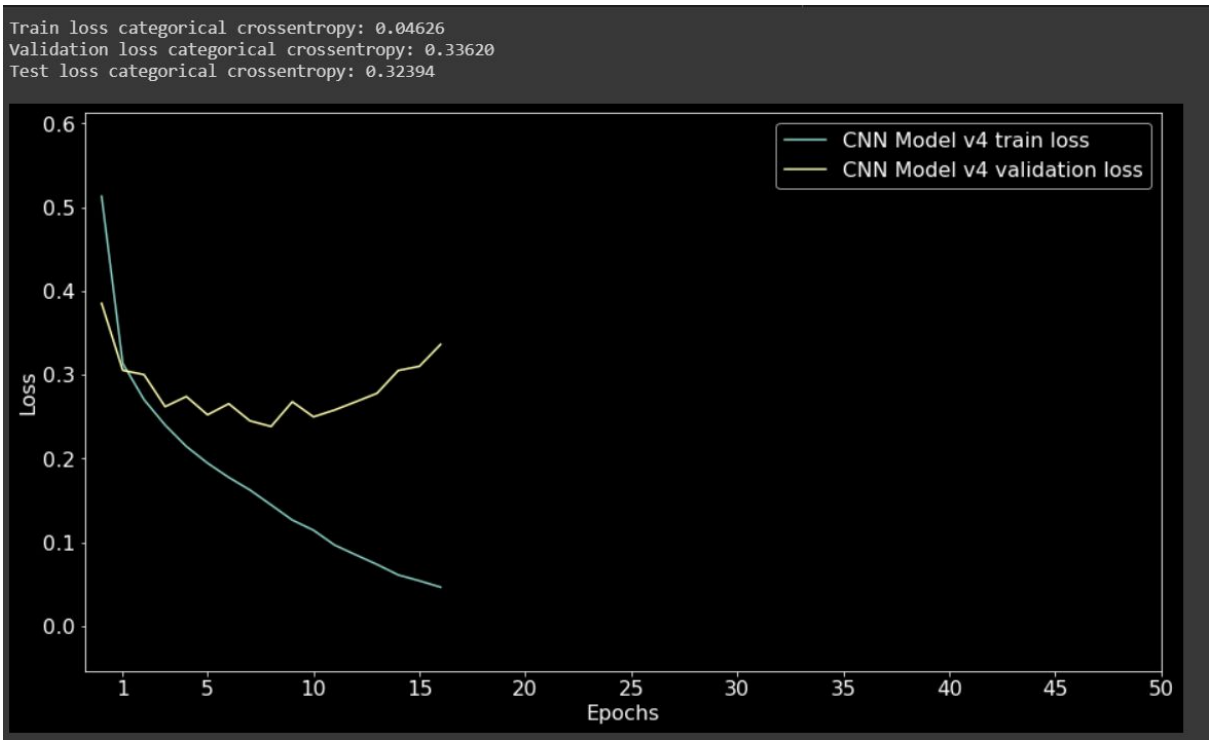
The next implementations use Convolutional Neural Networks architecture. Firstly, we have to reshape our data with the appropriate dimensions in order to feed the CNN. As we use grayscale images of size 28x28, the data have to be reshaped to size 28x28x1 which the 1 is related with the number of different colours.

The approach of convolution activation functions and output function will be the same as the previous MLPs(relu function/ softmax).There were performed various tests using dropout and early stopping with patient 5. Each model contains pooling layers of size (2,2) with stride 1.

model	Filter Dimension	Conv. Layers	Kernel Filter	Opti mizer	Validation Loss	Validation Accuracy	Test Accuracy
1st CNN model	8	2	2	Adam	0.31902	0.88833	0.8881
1st CNN model v2	8	2	2	SGD	0.40656	0.85483	0.8552
2nd CNN model	32	2	2	Adam	0.29056	0.9075	0.9093
3rd CNN model	32	2	3	Adam	0.27192	0.91133	0.9099
4rth CNN with MLP	32	2	3	Adam	0.3362	0.91533	0.9205
5th CNN with MLP	32	3	3	Adam	0.34504	0.9125	0.9192
6th CNN with MLP	64	2	3	Adam	0.40664	0.91983	0.9161

As we can observe, the best score at test data was 0.9205 and achieved with the 4rth CNN which is a CNN architecture with an MLP hidden layer.

However, looking at the loss plot of 4rth CNN, there is a clear sign of overfit in train data.

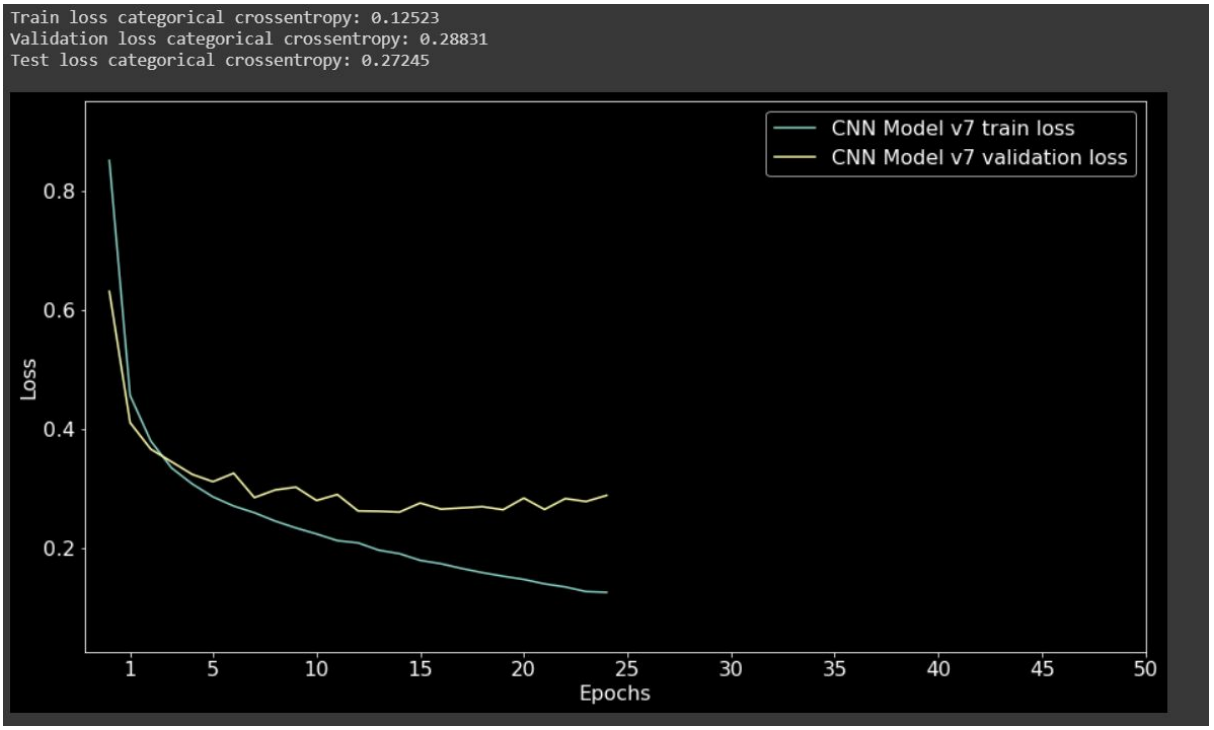


In order to decrease overfitting, we can try two different approaches. We can use SGD optimizer and another early stopping criteria like val_loss with patient 5.

As we can observe at the above plot, the validation loss stop decreasing after the 10th epoch. This means that we will achieve a slightly lower accuracy, but this is preferable than an overfitting model.

7th CNN with MLP	32	2	3	SGD	0.28831	0.9075	0.913
8th CNN with MLP early stop loss	32	2	3	SGD	0.26416	0.91267	0.9136

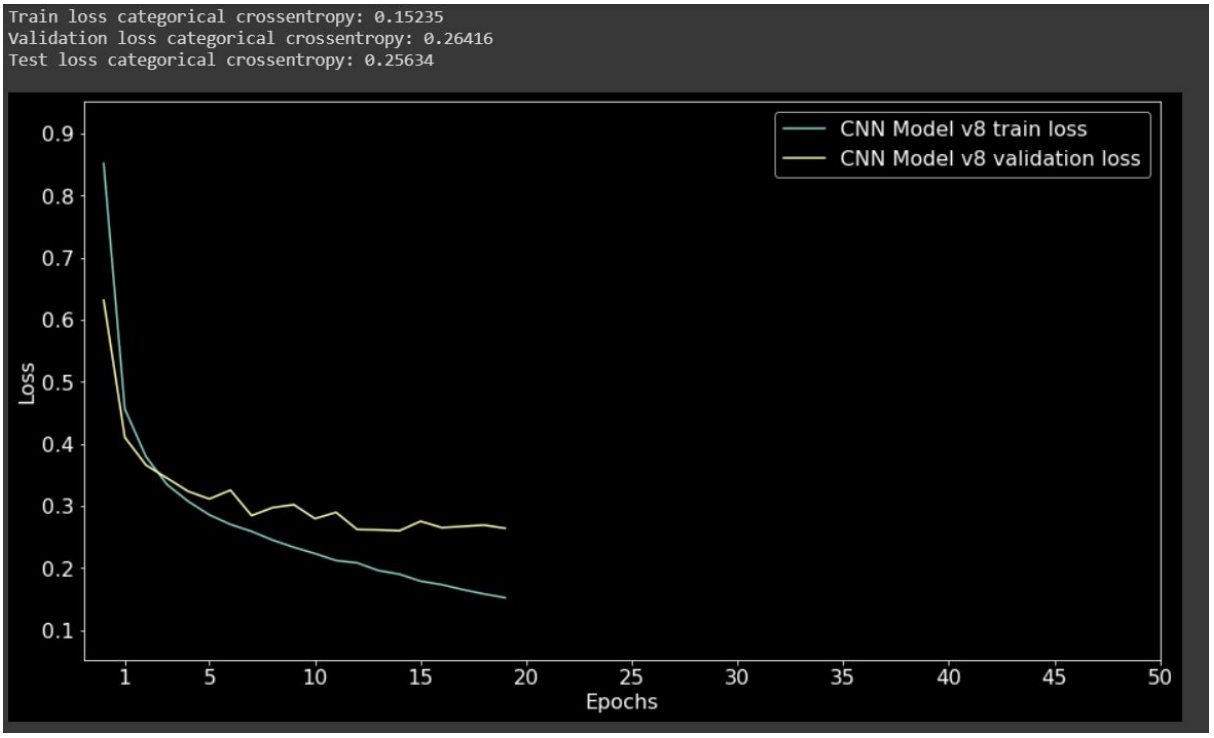
7th model using SGD with momentum 0.9




```
Train accuracy      : 0.95350
Validation accuracy: 0.90750
Test accuracy       : 0.91300
```

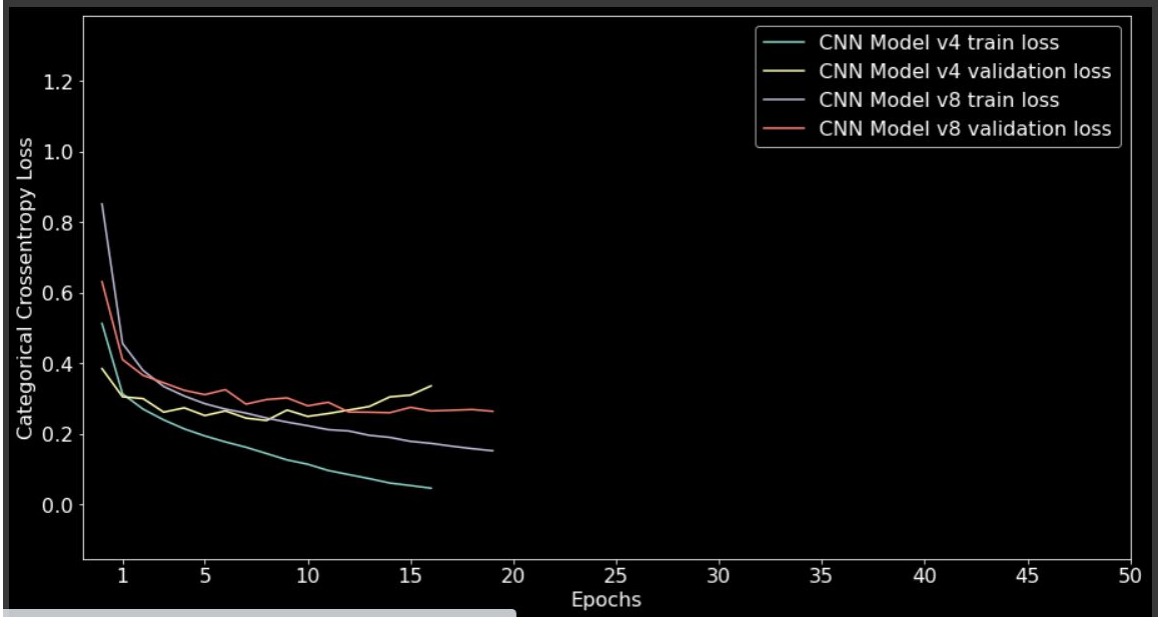
The accuracy was decreased from 0.9192 to 0.913 but the overfitting decreased too.

8th model using SGD with momentum 0.9 and early stopping on validation loss with patient 5

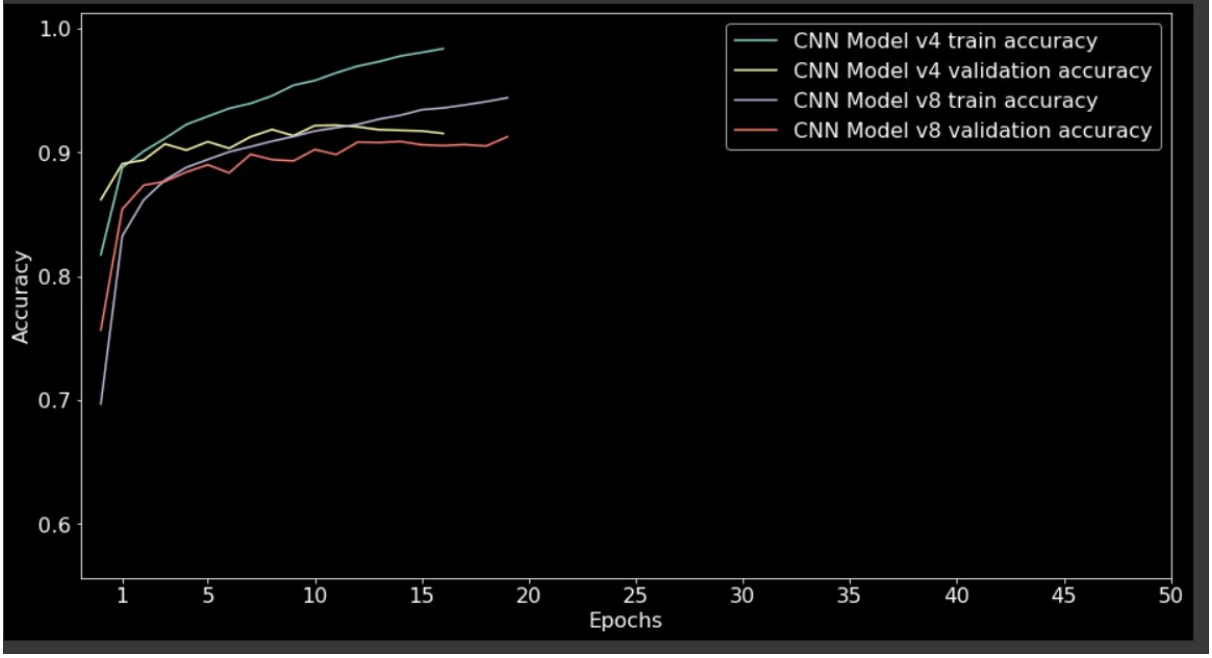


As we can observe, the model v8 perform an accuracy score on test data about 0.9136 and the model v4 a score equal to 0.9205.
However it seems that v8 performs less overfitting than v4 in train data and that is the reason **we choose v8 for our prediction**.

```
Train loss categorical crossentropy: 0.04626
Validation loss categorical crossentropy: 0.33620
Test loss categorical crossentropy: 0.32394
---
Train loss categorical crossentropy: 0.15235
Validation loss categorical crossentropy: 0.26416
Test loss categorical crossentropy: 0.25634
```



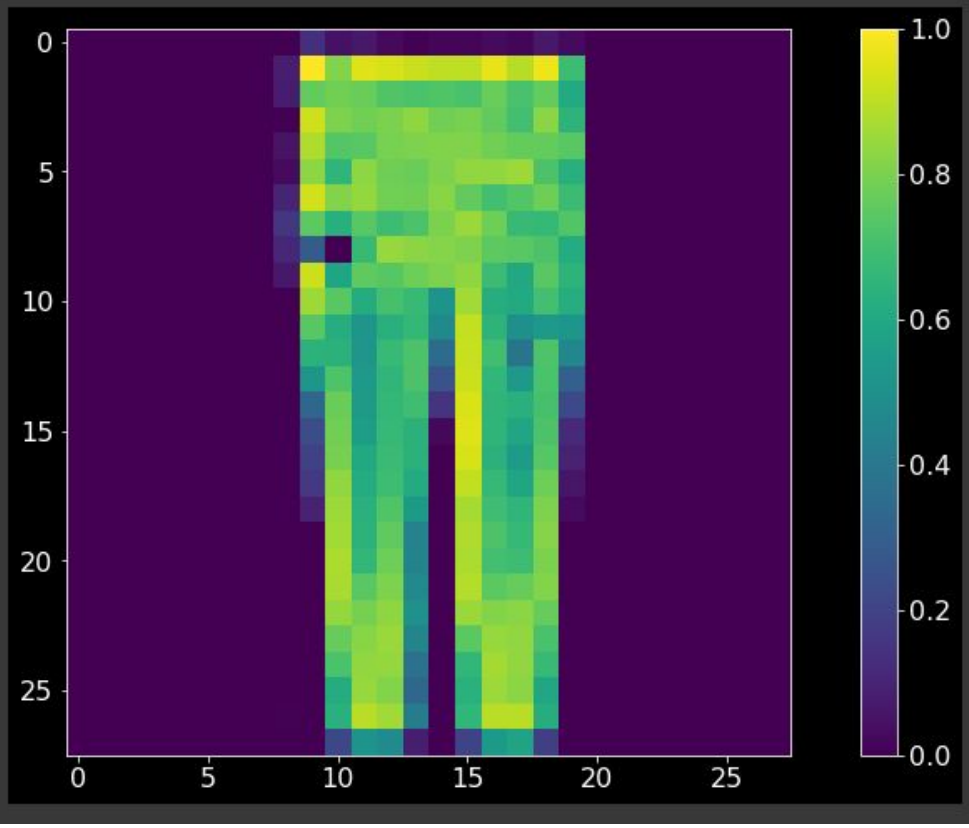
```
Train accuracy     : 0.98372
Validation accuracy: 0.91533
Test accuracy      : 0.92050
---
Train accuracy     : 0.94413
Validation accuracy: 0.91267
Test accuracy      : 0.91360
```



Prediction

Apply CNN_model_v8 on a random test image.

```
[116] plt.figure()
      plt.imshow(x_test_images[200])
      plt.colorbar()
      plt.grid(False)
      plt.show()
```




```
[114] y_test_labels[200]

      1
```

1 Trouser

```
result = cnn_model_v8.predict_proba(predict_image)
result.argmax(axis=-1)

array([1])
```