Fintech HW3

This homework aims to create a convolution neural network to realise a classification of the fashion MNIST dataset. It consists into recognizing the outfit type of each image and labelling them *["Top", "Trouser", "Pullover", "Dress", "Coat", "Sandal", "Shirt", "Sneaker", "Bag", "Ankle boot"]*. I will use the library KERAS to create the CNN and evaluate the results.

Question 1

In this question, it is needed to create convolution network for image recognition. In a second time, I test the effect of different settings (filter size, stride size and kernel size) using a Gridsearch.

The structure of my network is the following:
I have 2 identical layers made of:

- 1 convolution layer: Conv2D(filter_size,strides = stride_size, kernel_size= kernel_size, activation= 'relu')
- 1 maxpooling: MaxPool2D(pool_size=2, strides=2)
- 1 dropout layer: Dropout(0.3)

And at the end :

- 1 Flatten() (to get a 1 dimensional output)
- 1 simple layer : Dense(neuronsHL, activation= 'relu')
- 1 output layer : Dense(10, activation= 'softmax')

To test the influence of parameters, I achieve a grid search for different values. I chose the following grid:

```
neuronsHL = [64]
filter_size = [12, 36, 48]
stride_size = [1, 2]
kernel_size = [3, 4]
```

Then, I compute it on Google colab to have a result faster and plot the results :

```
Thanks to the optimisation we got this optimal results :
Best score in search :  0.857400000011921
Parameters used : {'epochs': 25, 'filter_size': 48, 'kernel_size': 3, 'neuronsHL': 64, 'stride_size': 1}
```

*Figure 1: **Grid search results** on Google colab*

According to the results, we can conclude that the result is better with:

- *A big filter sizes*
- *A small kernel*
- *A small stride*

Question 2

With the parameters that I obtained, I trace the **learning and accuracy curves**. (filter sizes = 48, kernel size = 3, stride = 1)

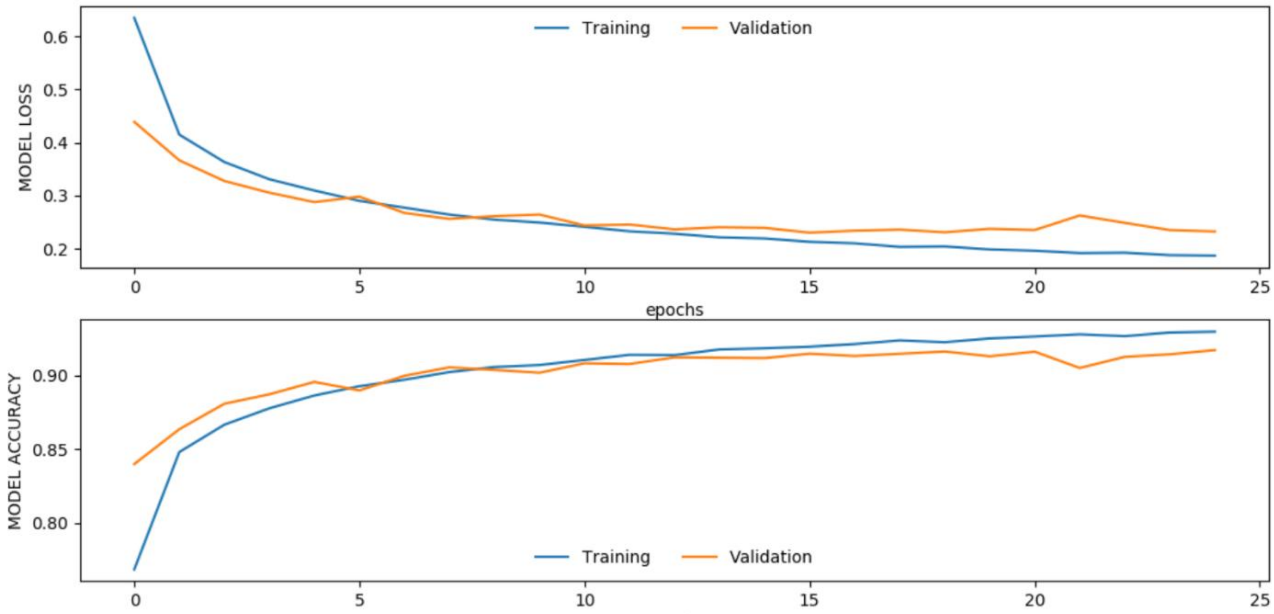

*Figure 2: **Learning curve** and **accuracy curve***

Question 3

In this question, I chose a random sample (*figure 3*) and apply it to the first convolution layer of the model. I obtain the following result (*figure 4*): we get 48 images of size 26x26. The reduction of the size is because of the convolution which apply locally a kernel of 3x3 to create one output value.

It is noticeable that all the 48 images are different. This is because the kernels used for each image are all different. According to the KERAS Documentation, the kernel weights are chosen randomly, and this confirms our observations. Intuitively, this this layer gives us different "versions" of the input sample.

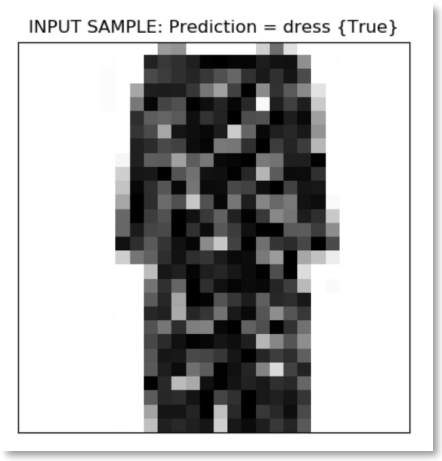

Figure 3: Sample randomly chosen

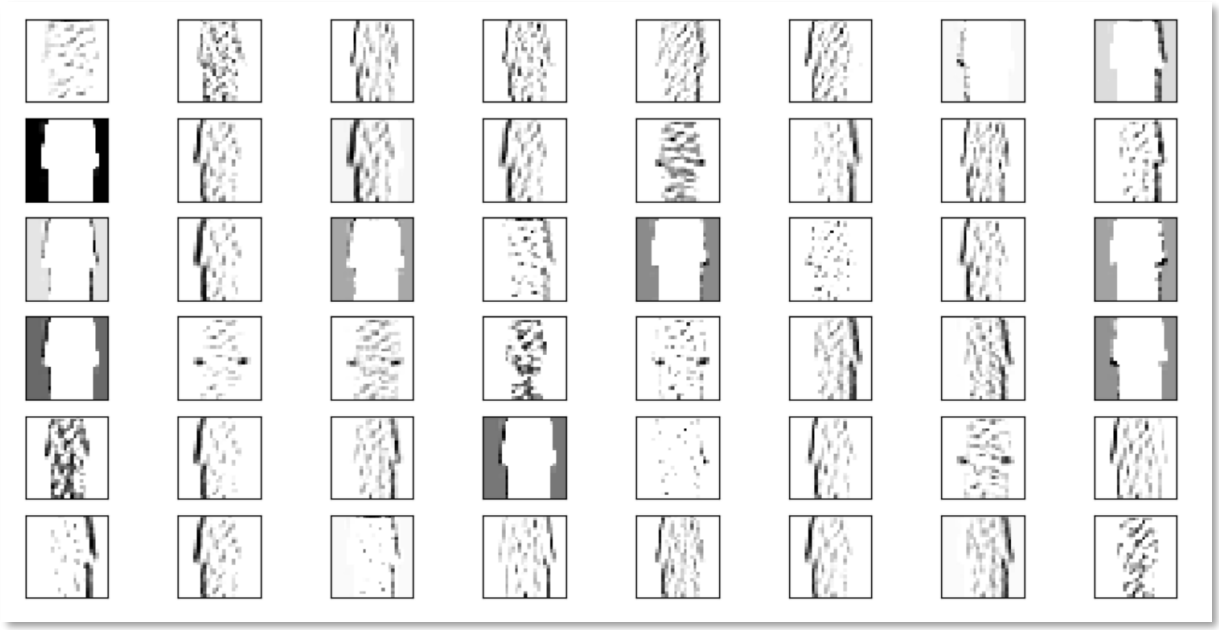

Figure 4: Activation images corresponding to the samples

Question 4

The purpose of this question is to classify 16 sample chosen randomly in testX. I then plot the samples and write the corresponding Labels obtained thanks to the model (*figure 5*). If it is green, the prediction is correct and if red, it is incorrect.

After different tests, I observe that my model is fairly accurate. It may give wrong results in case the object shape seems to belong to several classes; like the elements of the classes *Top*, *Pull-over*, *Coat* and *shirt.* The *figure 6* gives us the precisions of the model class by class. We observe that these 4 have a lower precision than the others. This phenomenon can be explained because the elements of these categories are visually. Indeed, they all have two sleeves and occupy the same space on the image which makes the classification harder.

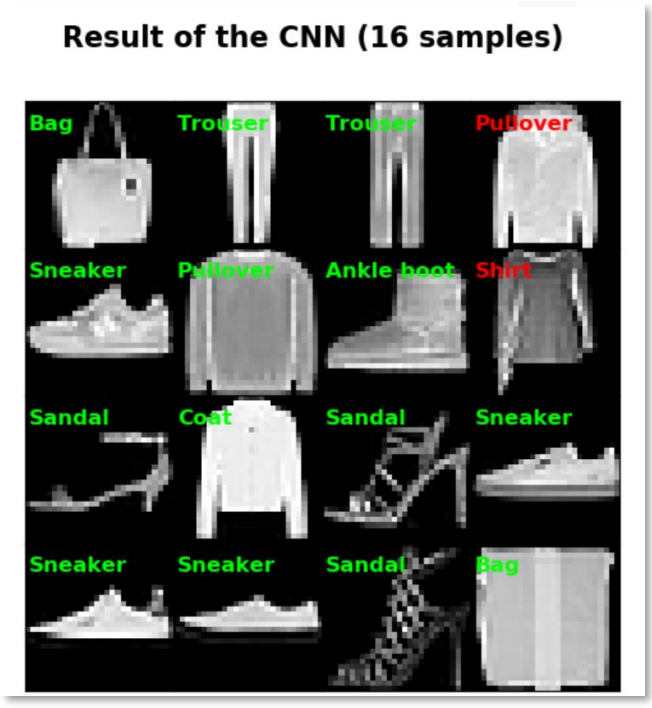

Figure 5: Plot of the classification results

CONCLUSION ON PRECISION:

	precision	recall	f1-score	support
Top	0.86	0.87	0.86	1402
Trouser	0.99	0.98	0.99	1394
Pullover	0.87	0.88	0.87	1385
Dress	0.93	0.91	0.92	1406
Coat	0.86	0.86	0.86	1420
Sandal	0.99	0.97	0.98	1418
Shirt	0.76	0.75	0.75	1388
Sneaker	0.95	0.98	0.96	1363
Bag	0.98	0.98	0.98	1417
Ankle boot	0.97	0.97	0.97	1406
accuracy			0.92	13999
macro avg	0.92	0.92	0.92	13999
weighted avg	0.92	0.92	0.92	13999

Figure 6: Precision of the results by class