HAUTE ÉCOLE
D'INGÉNIERIE ET DE GESTION
DU CANTON DE VAUD
www.heig-vd.ch

Enzo Allemano
Yohann Paulus

13/05/2022

Perez Uribe Andres
Izadmehr Yasaman

Apprentissage par Réseaux de Neurones artificiels

Rapport laboratoire 4:
Deep Neural Networks

Question 1 : What is the learning algorithm being used to optimize the weights of the neural networks? What are the parameters (arguments) being used by that algorithm? What cost function is being used ? please, give the equation(s)

Question 2 : Model complexity: for each experiment (shallow network learning from raw data, shallow network learning from features, CNN, and Fashion MNIST), select a neural network topology and describe the inputs, indicate how many are they, and how many outputs. Compute the number of weights of each model (e.g., how many weights between the input and the hidden layer, how many weights between each pair of layers, biases, etc..) and explain how do you get to the total number of weights.

Question 3 : Do the deep neural networks have much more "capacity" (i.e., do they have more weights?) than the shallow ones? explain with one example

Question 4 : Test every notebook for at least three different meaningful cases (e.g., for the MLP exploiting raw data, test different models varying the number of hidden neurons, for the feature-based model, test pix_p_cell 4 and 7, and number of orientations or number of hidden neurons, for the CNN, try different number of neurons in the feed-forward part) describe the model and present the performance of the system (e.g., plot of the evolution of the error, final evaluation scores and confusion matrices). Comment the differences in results. Are there particular digits that are frequently confused?

Question 5 : Train a CNN to solve the MNIST Fashion problem, present your evolution of the errors during training and perform a test. Present a confusion matrix, accuracy, F-score and discuss your results. Are there particular fashion categories that are frequently confused?

Question 1

L'algorithme utilisé est « RMSprop » avec tous les paramètres par défauts .

Il utilise les paramètres suivants :

- Learning rate
- Rho : Facteur d'actualisation pour le gradient à venir
- Momentum
- Epsilon : une petite constante pour la stabilité numérique
- Centered
- Name

La fonction de coût est « categorical_crossentropy ». L'équation qui correspond est la suivante :

$$\text{Loss} = -\frac{1}{\substack{\text{output} \\ \text{size}}} \sum_{i=1}^{\substack{\text{output} \\ \text{size}}} y_i \cdot \log \hat{y}_i + (1 - y_i) \cdot \log (1 - \hat{y}_i)$$

Question 2

MLP_from_raw_data

```
Model: "sequential_3"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_6 (Dense)             (None, 300)               235500

 dropout_1 (Dropout)         (None, 300)               0

 dense_7 (Dense)             (None, 10)                3010

=================================================================
Total params: 238,510
Trainable params: 238,510
Non-trainable params: 0
_________________________________________________________________
```

Nombre d'entrée : 768 (image_size*image_size)

Une couche dense de 300 neurones.

Un Dropout de 0.5 qui montre que la moitié des neurones ne sont pas utilisés.

Une couche de sortie avec 10 sorties. (une pour chaque digits)

$dense\_6 = nombre\ d'entrée * nombre\ de\ neurones + biais = 768 * 300 + 300 = 235'500$

$dense\_7 = nombre\ de\ sortie\ * nombre\ de\ neurones + biais = 300 * 10 + 10 = 3'010$

Pour finir on a :

$Total\ params = dense\_6 + dense\_7 = 238'510$

MLP_from_HOG

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 200)               78600

 dense_1 (Dense)             (None, 10)                2010

=================================================================
Total params: 80,610
Trainable params: 80,610
Non-trainable params: 0
_________________________________________________________________
```

Nombre d'entrée : 392 = hog_size

Une couche dense de 200 neurones.

Une couche de sortie avec 10 sorties. (une pour chaque digits)

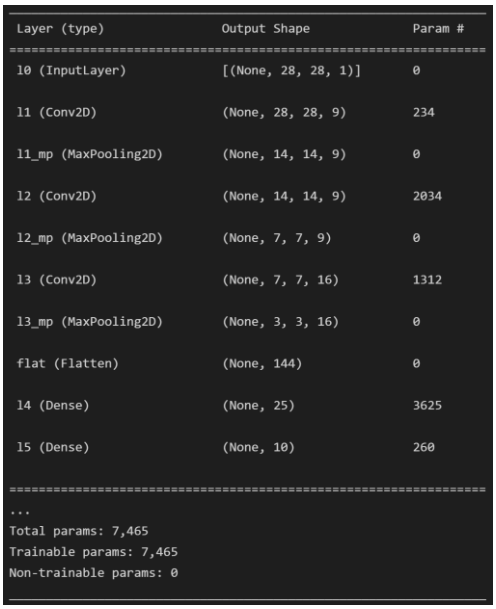
$$dense = nombre\ d'entrée * nombre\ de\ neurones + biais = 200 * 392 + 300 = 78600$$

$$dense\_1 = nombre\ de\ sortie\ * nombre\ de\ neurones + biais = 200 * 10 + 10 = 2'010$$

Pour finir on a :

$$Total\ params = dense + dense\_1 = 80'610$$

CNN



Layer (type)	Output Shape	Param #
l0 (InputLayer)	[(None, 28, 28, 1)]	0
l1 (Conv2D)	(None, 28, 28, 9)	234
l1_mp (MaxPooling2D)	(None, 14, 14, 9)	0
l2 (Conv2D)	(None, 14, 14, 9)	2034
l2_mp (MaxPooling2D)	(None, 7, 7, 9)	0
l3 (Conv2D)	(None, 7, 7, 16)	1312
l3_mp (MaxPooling2D)	(None, 3, 3, 16)	0
flat (Flatten)	(None, 144)	0
l4 (Dense)	(None, 25)	3625
l5 (Dense)	(None, 10)	260

...
Total params: 7,465
Trainable params: 7,465
Non-trainable params: 0

La cellule 4 est très utile pour comprendre l'architecture de CNN.

L1 = 234

L2=2'034

L3=1'312

L4=3'625

L5=260

$Total\ params = L1 + L2 + L3 + L4 + L5 = 7'465$

Fashion MNIST

Les résultats sont exactement les mêmes que pour CNN. C'est dû au fait que l'architecture ne change pas.

Question 3

Les « deep neural networks » on plus de capacité que les « shallow neural networks ». En théorie les « shallow neural networks » peuvent atteindre des performances équivalentes aux « deep neural networks » mais en pratique ce n'est généralement pas le cas.

Dans les explications possibles nous avons plusieurs théories :

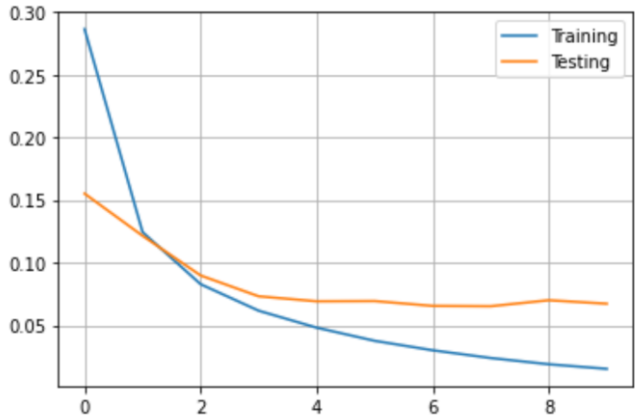
- Premièrement, les « shallow neural networks » nécessite peut-être plus de neurones que les « deep neural networks ».
- Deuxièmement, peut-être que les « shallow neural networks » nécessitent différent algorithme d'entrainement que ceux utiliser actuellement ou dernièrement, peut-être que cette architecture ne correspond pas aux problèmes que nous essayons de résoudre en général.

Question 4

MLP_from_raw_data

Résultat avec les paramètres de base

Nombre de neurones	Dropout
300	-

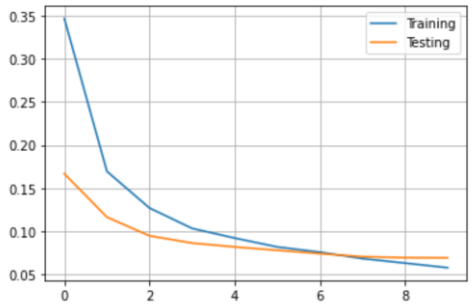


Test score: 0.06988484412431717

Test accuracy: 0.9811999797821045

Résultat avec les paramètres suivants

Nombre de neurones	Dropout
200	0.3



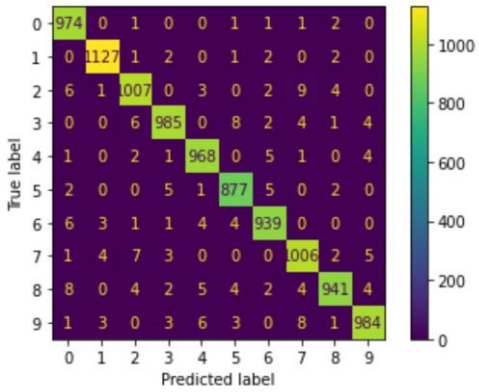
Test score: 0.06868607550859451

Test accuracy: 0.9800999760627747

F1 Score :

0	1	2	3	4
0.98433552	0.991641	0.97719554	0.97912525	0.98324022

5	6	7	8	9
0.97988827	0.98016701	0.97622513	0.97563504	0.97910448

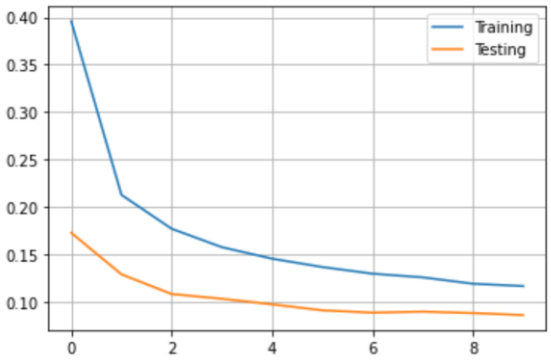


Le résultat est du graphe pour les tests sont meilleures. On peut notamment voir que la matrice de confusion présente plutôt de bon résultat

---

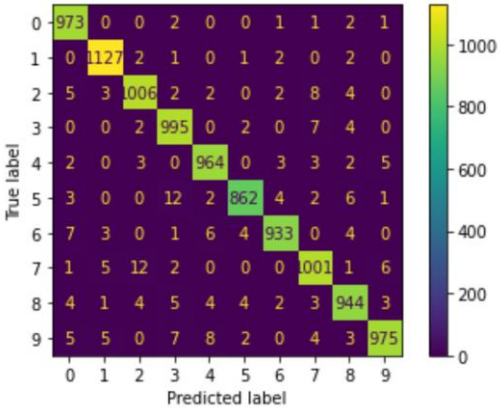
Résultat avec les paramètres suivants

Nombre de neurones	Dropout
500	0.7



Test score: 0.0855076014995575

Test accuracy: 0.9779999852180481



F1 Score :

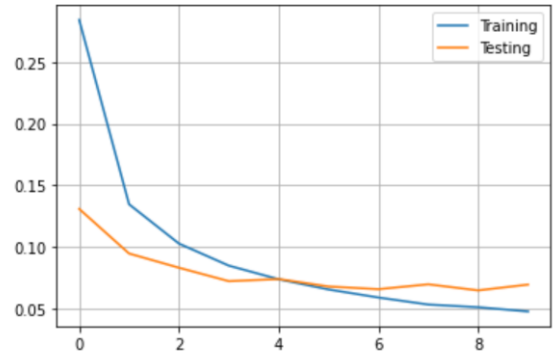
0	1	2	3	4
0.98282828	0.98903028	0.97622513	0.97692685	0.9796748

5	6	7	8	9
0.97566497	0.97952756	0.97326203	0.97019527	0.975

Le test score présente un résultat un petit mieux. Cependant, on n'atteint pas assez rapidement un point de convergence. On a une sorte d'underfiting.

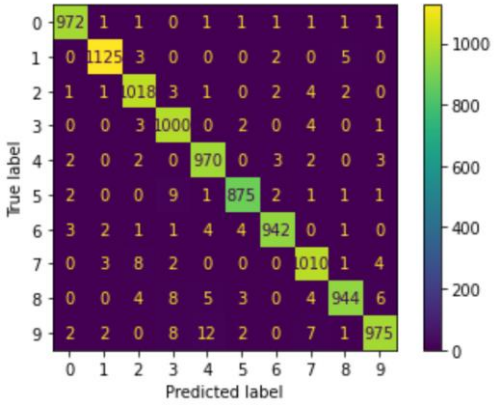
Résultat avec les paramètres suivants

Nombre de neurones	Dropout
900	0.5



Test score: 0.0693397969007492

Test accuracy: 0.9830999970436096



F1 Score :

0	1	2	3	4
0.99082569	0.99162627	0.98262548	0.97991181	0.98178138

5	6	7	8	9
0.98369871	0.98638743	0.98010674	0.97823834	0.975

Ici on remarque une très bonne accuracy. On peut notamment relever que le digits 5 à un score de true positive relativement bas (875).

Concernant le graphe, celui-ci est un peu moins constant que les autres essaies avec d'autre paramètre. Cependant, les résultats restent très bons !

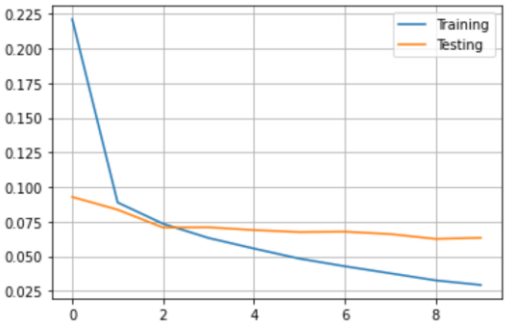
MLP_from_raw_data

Pour les prochaines séries de tests, les paramètre ont été laissé à leur valeur par défaut.

Nombre de neurones	Dropout
300	-

Résultat avec les paramètres de base

Pix_p_cell	N_orientations
8	4

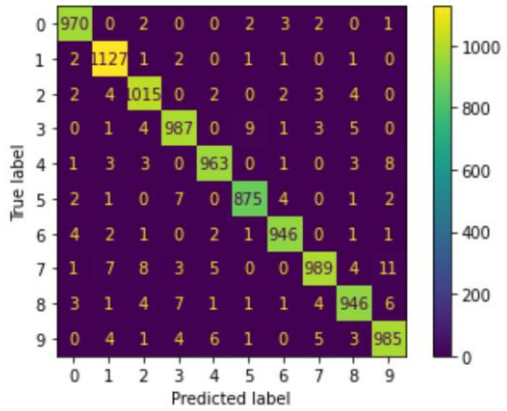


Test score: 0.06325872242450714

Test accuracy: 0.9803000092506409

F1 Score :

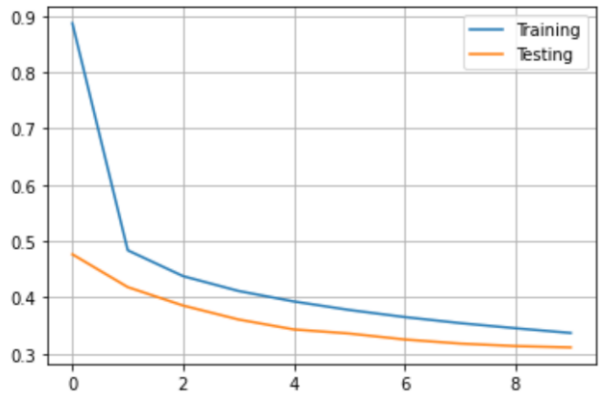
0	1	2	3	4
0.98727735	0.98643326	0.9802028	0.97722772	0.98215196



5	6	7	8	9
0.98204265	0.98695879	0.97246804	0.97425335	0.97380129

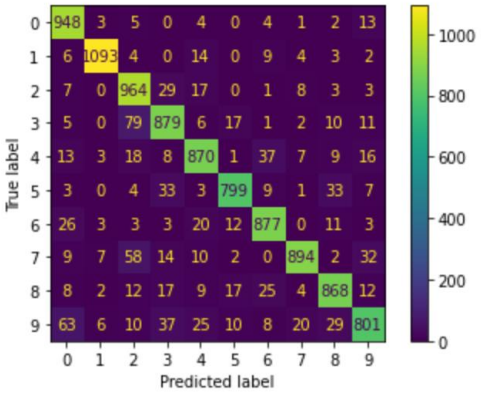
Résultat avec les paramètres suivants

Pix_p_cell	N_orientations
10	14



Test score: 0.31146347522735596

Test accuracy: 0.8992999792098999

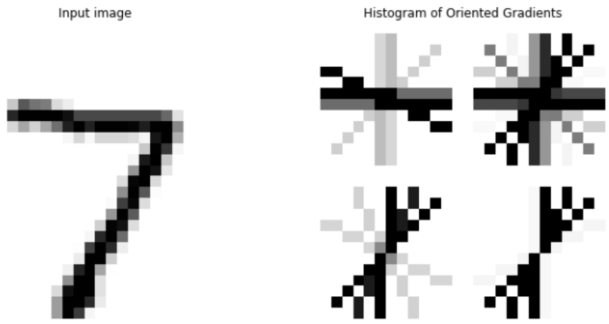


F1 Score :

0	1	2	3	4
0.91682785	0.97069272	0.88076747	0.86600985	0.8877551

5	6	7	8	9
0.91314286	0.90927942	0.90807517	0.89300412	0.83918282

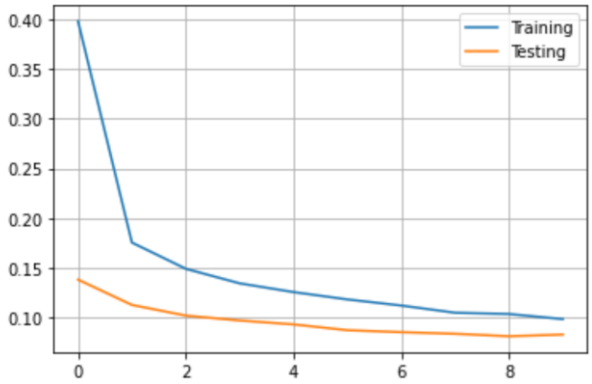
Nos résultat ici sont extrêmement mauvais. Notre accuracy n'est vraiment pas très bonne. Le graphe montre de l'underfiting.



L'histogramme de la descente de Gradient est également mauvais.

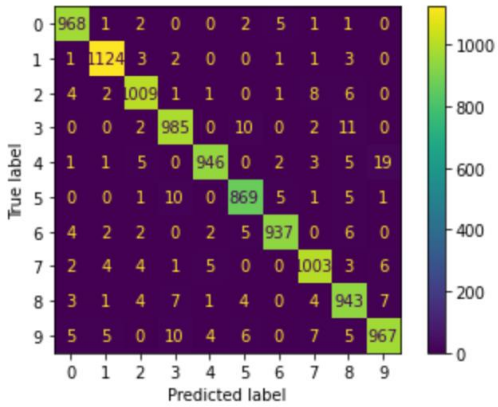
Résultat avec les paramètres suivants

Pix_p_cell	N_orientations	Dropout
4	4	0.5



Test score: 0.08238252252340317

Test accuracy: 0.9750999808311462

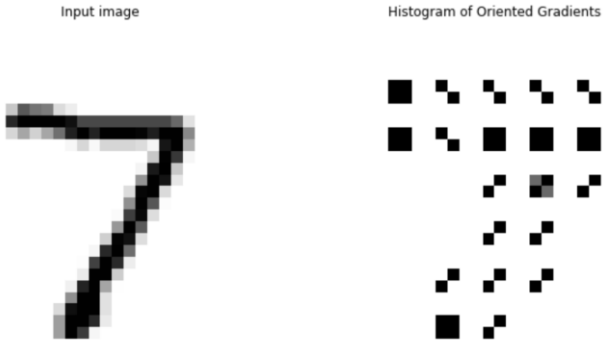


F1 Score :

0	1	2	3	4
0.98373984	0.98813187	0.97771318	0.97235933	0.97475528

5	6	7	8	9
0.97203579	0.98166579	0.97473275	0.96126402	0.96266799

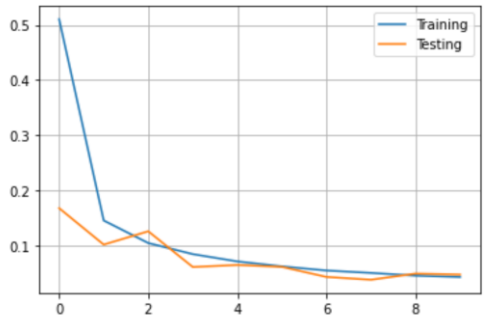
Ce modèle-ci est beaucoup plus attirant que le précédent. L'accuracy est notamment relativement très bonne, même si on peut remarquer un petit peu d'underfiting.



L'histogramme de la descente de Gradient est quant à lui aussi très bon.

CNN

Résultat avec les paramètres de bases

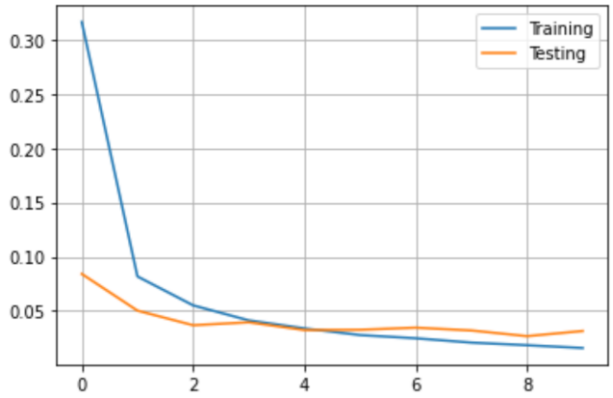


F1 Score :

0	1	2	3	4
0.98715973	0.99603699	0.98039216	0.97949219	0.98765432

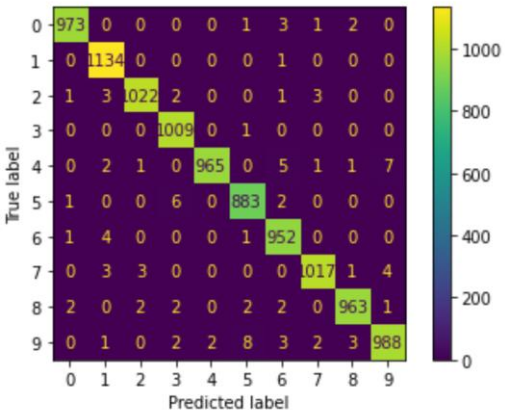
5	6	7	8	9
0.98181818	0.98523207	0.98547919	0.97516472	0.98203593

Résultat avec les valeurs des filtres doublé pour chaque couche



Test score: 0.03138962388038635

Test accuracy: 0.9905999898910522



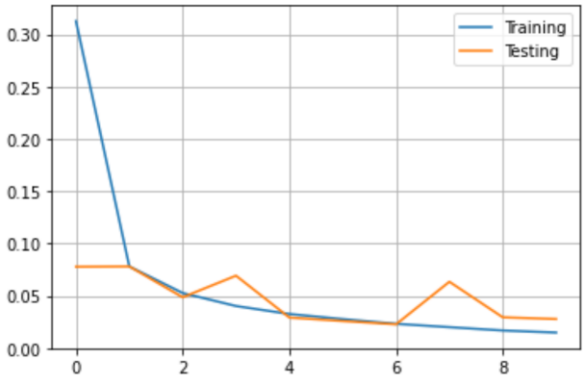
F1 Score :

0	1	2	3	4
0.9938713	0.99386503	0.99223301	0.99359921	0.99025141

5	6	7	8	9
0.98769575	0.98806435	0.99122807	0.99074074	0.98357392

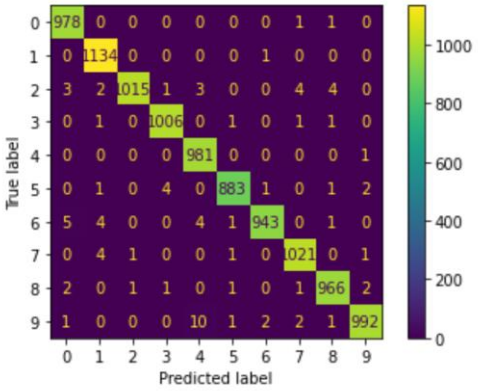
On peut constater de très légères améliorations de résultat par rapport à la version avec les paramètres de base.

Résultat avec les valeurs des filtres doublé pour chaque couche et une augmentation de dense unit



Test score: 0.02767784334719181

Test accuracy: 0.9919000267982483



F1 Score :

0	**1**	**2**	**3**	**4**
0.99339766	0.99430075	0.99072718	0.9950544	0.99090909

5	**6**	**7**	**8**	**9**
0.99213483	0.99002625	0.99222546	0.99127758	0.98854011

Ici on peut constater un graphe relativement étrange avec deux piques. Encore une fois, on obtient de bons résultats. La meilleure conclusion que l'on peut tirer est que ce sont les filtres qui exercent une grande influence sur notre réseau de neurones. En effet, même en diminuant le paramètre de dense ça ne change pas énormément le résultat. On remarque simplement, une légère diminution de l'accuracy.

On tirer une petite conclusion sur le digits 5. D'après les outputs, ce digits est le plus difficile à idientifer. En effet, il est souvent confondu avec le digits 3.

Question 5

Pour cette partie de test du laboratoire, nous avons décidé de garder le maximum de paramètre identique au MLP qui permet de résoudre le problème du dataset avec les digits.

Nous avons donc notamment choisi de garder l'algorithme RMSporp comme algorithme d'apprentissage.

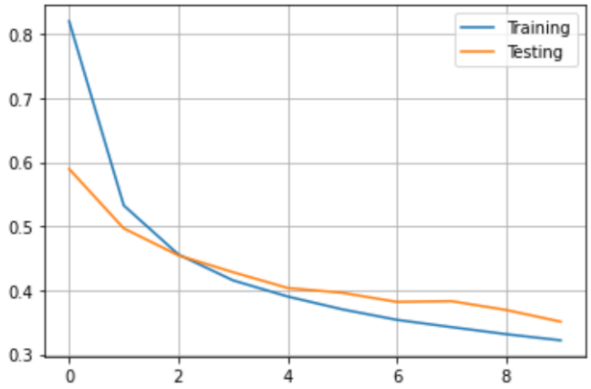
Le dataset sur lequel nous allons entrainer notre modèle est le suivant :

```
from keras.datasets import fashion_mnist
```

La structure de ses données ressemble à ça :

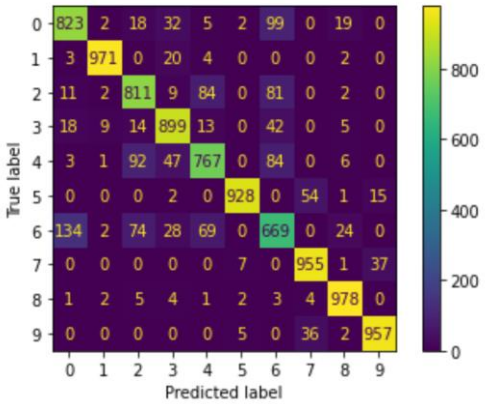
Label	Description
0	T-shirt/top
1	Trouser
2	Pullover
3	Dress
4	Coat
5	Sandal
6	Shirt
7	Sneaker
8	Bag
9	Ankle boot

Un premier entrainement avec les paramètres de bases nous donne le résultat suivant :



Test score: 0.3510725200176239

Test accuracy: 0.8758000135421753



F1 Score :

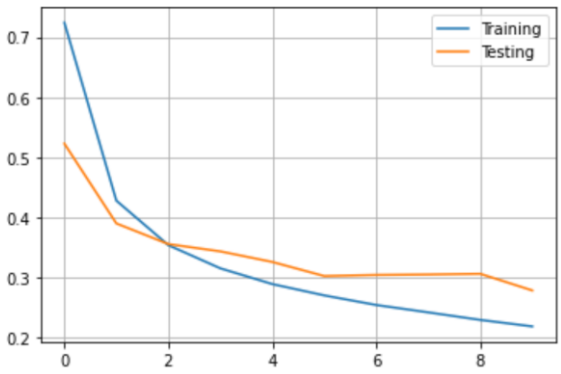
0	**1**	**2**	**3**	**4**
0.82589062	0.97637004	0.80536246	0.88094072	0.78950077

5	**6**	**7**	**8**	**9**
0.95473251	0.67644085	0.93216203	0.95882353	0.95271279

Première impression : les résultats ne sont pas vraiment très bons… On remarque notamment que la classe 6 (qui correspond Shirt) a beaucoup de mal à être classé. Elle rencontre particulièrement des difficulté à être classé avec la classe 1 (qui correspond à Tshirt/top)

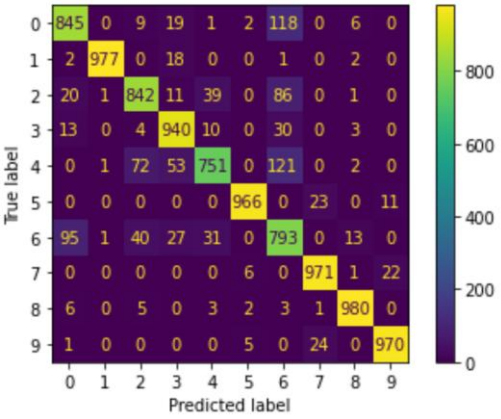
On pourrait imaginer modifier le learning rate pour obtenir de meilleurs résultats, sans oublier le fait que s'il est trop bas cela pourrait perturber le reste de nos prédictions.

Maintenant on va essayer de modifier certains paramètre dans le but d'augmenter la précision de nos prédictions :



Test score: 0.27906954288482666

Test accuracy: 0.8998000025749207



F1 Score :

0	1	2	3	4
0.84274809	0.98275862	0.84833659	0.89229296	0.82116402

5	6	7	8	9
0.97693079	0.73048601	0.95972153	0.97993982	0.96407186

Voici un des meilleurs résultats que nous avons pu obtenir. Très légèrement meilleure que le précédent mais meilleure quand même. Une amélioration, même minime reste très précieuse !