ARN - Practical Work 4 – Deep Neural Networks

Auteurs: Jonathan Friedli Valentin Kaelin

Date: 11.05.2022

> ❓ 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)

L'algorithme utilisé est le RMSprop. Voici les paramètres qu'il est possible d'utiliser avec Keras:

```
tf.keras.optimizers.RMSprop(
    learning_rate=0.001, # learning rate
    rho=0.9,             # Facteur d'actualisation pour le gradient historique/à venir
    momentum=0.0,        # momentum
    epsilon=1e-07,       # epsilon
    centered=False,      # normalisé ou pas
    name="RMSprop",
    **kwargs
)
```

L'équation interne utilisée est la suivante:

$$E[g^2]_t = \beta E[g^2]_{t-1} + (1-\beta)\left(\frac{\delta C}{\delta w}\right)^2$$

$$w_t = w_{t-1} - \frac{\eta}{\sqrt{E[g^2]_t}} \frac{\delta C}{\delta w}$$

E[g] — moving average of squared gradients. dC/dw — gradient of the cost function with respect to the weight. n — learning rate. Beta — moving average parameter (good default value — 0.9)

La loss function que nous utilisons est : "categorical_crossentropy".

https://keras.io/api/losses/probabilistic_losses/#categoricalcrossentropy-class

❓ 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.

Shallow network learning from raw data

Modèle choisi: Nous avons juste changé le nombre de neurone de 300 à 242

Number of inputs: `784`

Number of neuron in the hidden layer: `242`

Outputs: `10`

Weight in hidden layer: `784 * 242 + 242 = 189’970`

Weight in output: `10 * 242 + 10 = 2430`

Total Weights: `189’970 + 2430 = 192’400`

Shallow network learning from features

Modèle choisi : Celui de base (**pix_p_cell 4, 8 orientation**)

Number of inputs: `392`

Number of neuron in the hidden layer: `200`

Outputs: `10`

Weight in hidden layer: `392 * 200 + 200 = 78’600`

Weight in output: `200 * 10 + 10 = 2010`

Total Weights: `78’600 + 2010 = 80'610`

CNN

Modèle choisi : Celui de base

Outputs: `10`

`l1 (Conv2D) = 9 * 25 + 9 = 234`

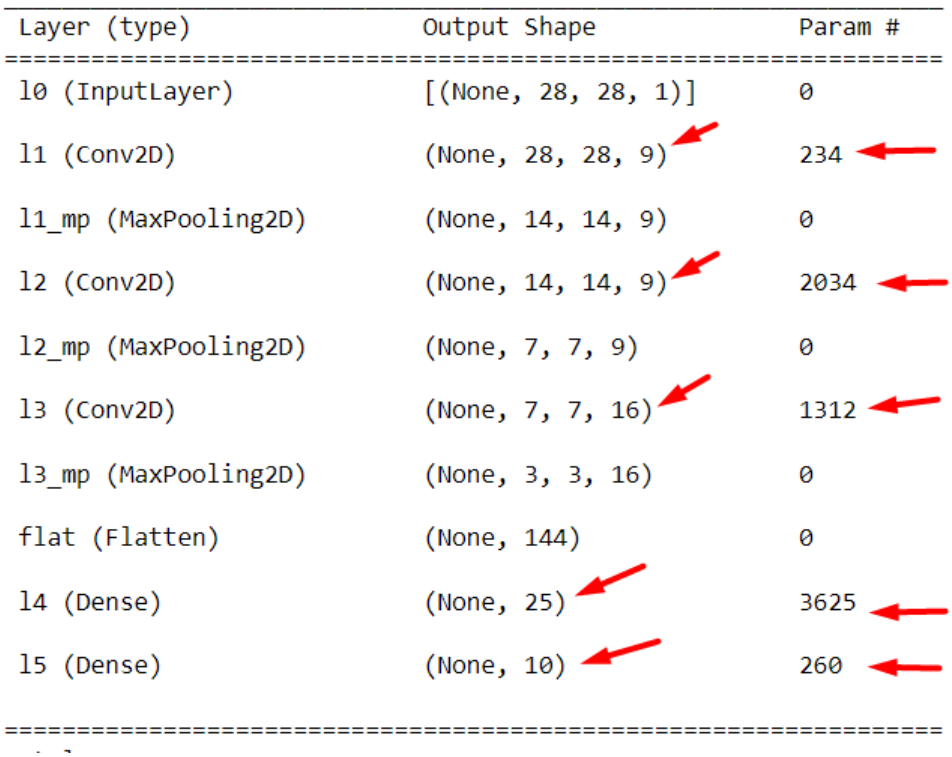
`l2 (Conv2D) = 9 * 225 + 9 = 2034`

`l3 (Conv2D) = 16 * 81 + 16 = 1312`

`l4 (Dense) = 25 * 144 + 25 = 3625`

`l5 (Dense) = 10 * 25 + 10 = 260`

Total Weights: `234 + 2034 + 1312 + 3625 + 260 = 7'465`



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

Fashion MNIST

Modèle choisi : Nous sommes passé de 25 neurones et 10 epoch à 10 neurones et 50 epoch

Outputs: `10`

`l1 (Conv2D) = 9 * 25 + 9 = 234`

`l2 (Conv2D) = 9 * 225 + 9 = 2034`

`l3 (Conv2D) = 16 * 81 + 16 = 1312`

`l4 (Dense) = 10 * 144 + 10 = 1450`

`l5 (Dense) = 10 * 10 + 10 = 110`

Total Weights: `234 + 2034 + 1312 + 1450 + 110 = 5'140`

> ❓ Do the deep neural networks have much more “capacity” (i.e., do they have more weights?) than the shallow ones? explain with one example

Les shallow et deep neural network sont capables d'approximer n'importe quelle fonction. Pour un même niveau de précision, les deep neural network peuvent être beaucoup plus efficaces en termes de calcul et de nombre de paramètres.

Par exemple: dans le cadre ce laboratoire nous avons deux modèles shallow (from raw data et from feature). Ces deux modèles ont respectivement 190’000 et 80’000 de total weight tandis que les deu modèles deep neural network ont des poids totaux n’excédant pas les 10’000. Cela fait entre 8 et 20 fois moins de poids.

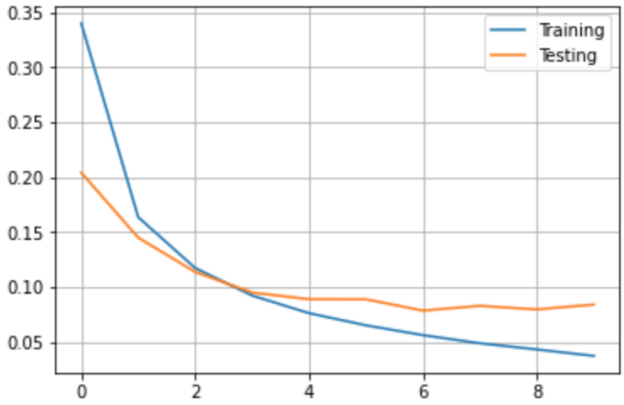
❓ Test every notebook for at least three different meaningful cases (e.g., for the MLP exploiting raw data, test different models varying the number of hidden neurons, for the feature-based model, test pix_p_cell 4 and 7, and number of orientations or number of hidden neurons, for the CNN, try different 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?

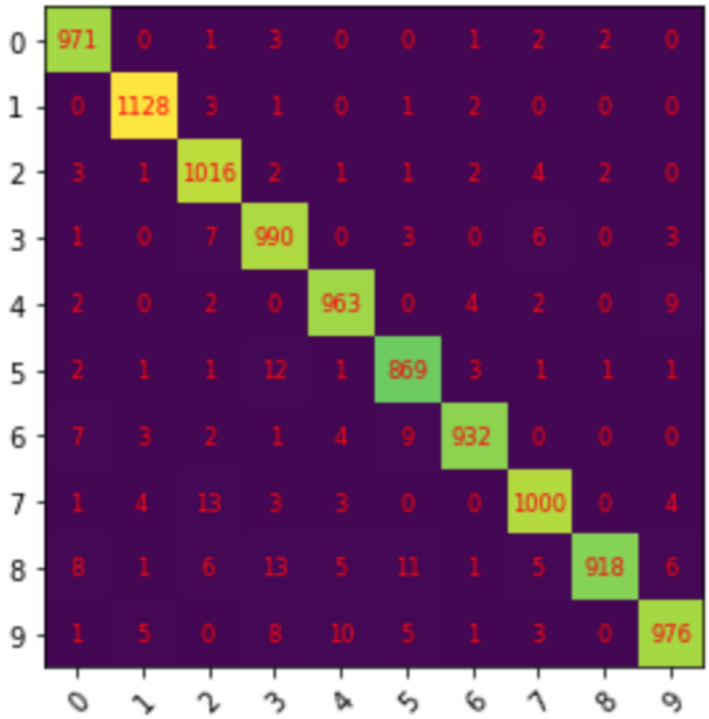
Shallow network learning from raw data

Nous avons essayé de modifier le nombre de neurones dans la couche cachée qui était initialement à 300. Peu importe les valeurs que nous testions, les résultats étaient moins bons que ceux initiaux. Nous avons donc finalement tenté de modifier le nombre de couche cachées (initialement il y en avait une seule).

110 neurones:

```
Test score: 0.08389027416706085
Test accuracy: 0.9763000011444092
```

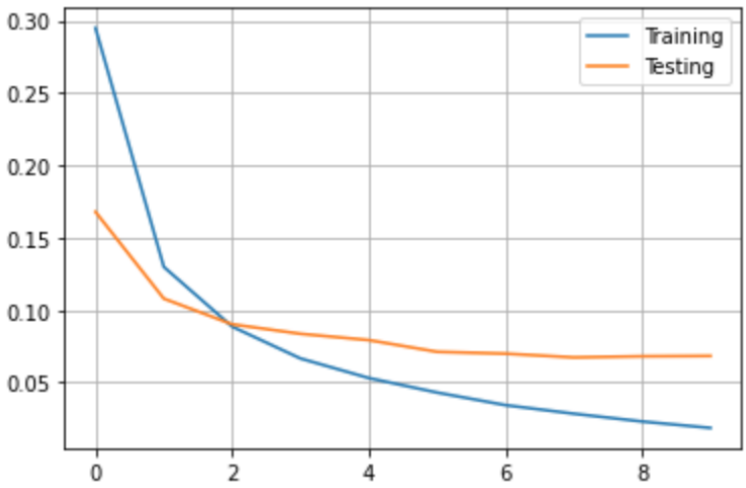


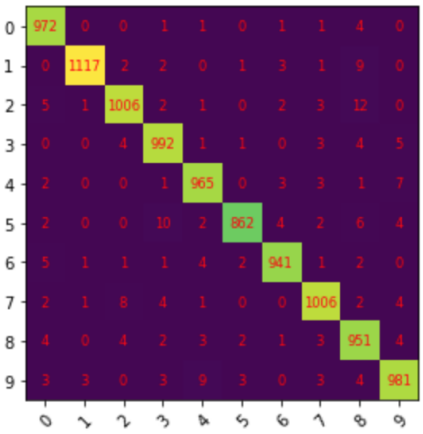


Comme nous avons vu un léger overfitting avec les 300 neurones initiaux, nous avons tenté de réduire drastiquement ce nombre. Nous observons encore de l'overfitting mais les résultats sont maintenant moins biens qu'avant.

242 neurones:

```
Test score: 0.06832381337881088
Test accuracy: 0.9793000221252441
```

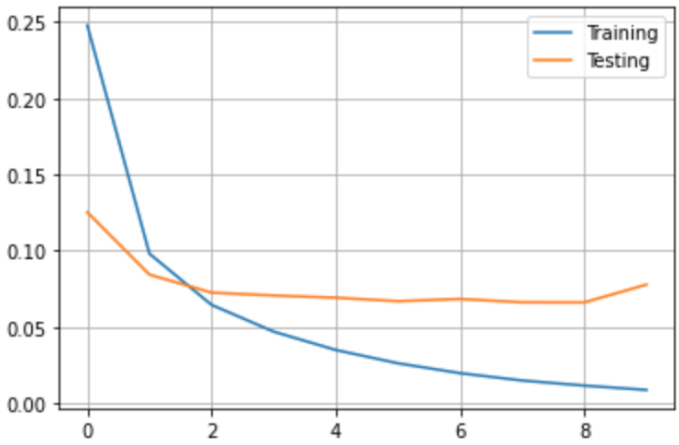


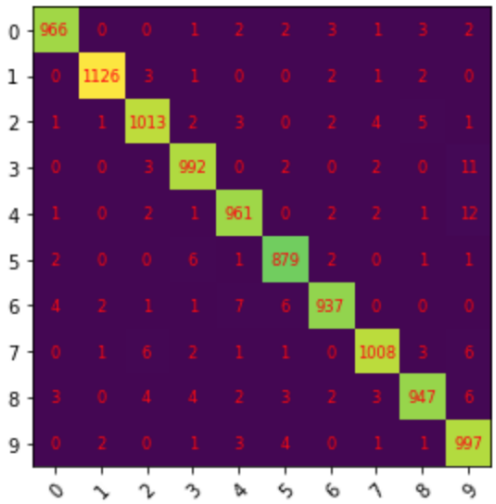


Nous observons de bons résultats, nous n'avons pas réussi à avoir de meilleurs résultats que ceux-ci en modifiant le nombre de neurones.

600 neurones:

```
Test score: 0.07777944952249527
Test accuracy: 0.9825999736785889
```





Comme nous aurions pu l'attendre, augmenter le nombre de neurones augmente encore l'overfitting, ce qui n'est clairement pas bénéfique. De plus nos résultats sont quand même moins bons qu'avec environ 240 neurones.

Globalement, ce premier modèle a de bonne performances mais souffre d'un léger overfitting. Il a plus de mal à classifier les chiffres suivants (avec 242 neurones):

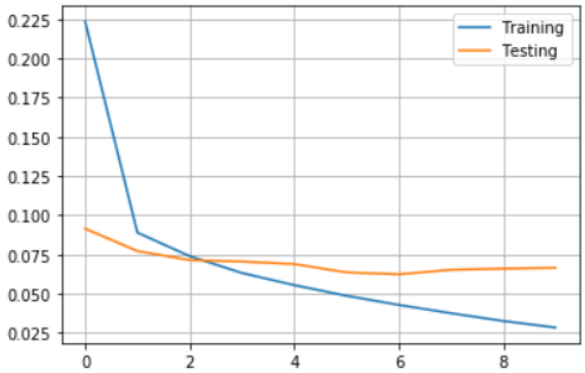
- Confond les 2 en 8
- Confond les 5 en 3

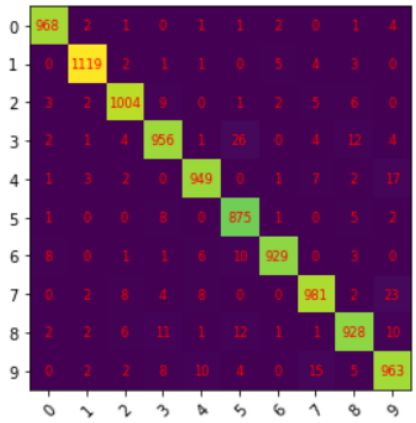
Les moins bon chiffres sont le 5 et le 6, ce qui est normal car leur ressemblance est plus élevée. Le chiffre 1 par exemple qui est bien spécifique est mieux classifié.

Shallow network learning from features

pix_p_cell 4, 8 orientation et 200 hidden neuron :

```
Test score: 0.06650333051643101
Test accuracy: 0.9801
```

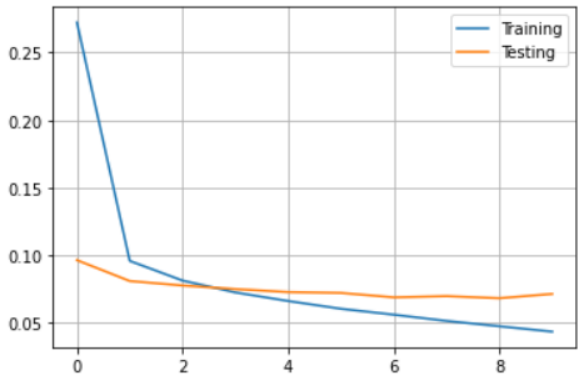


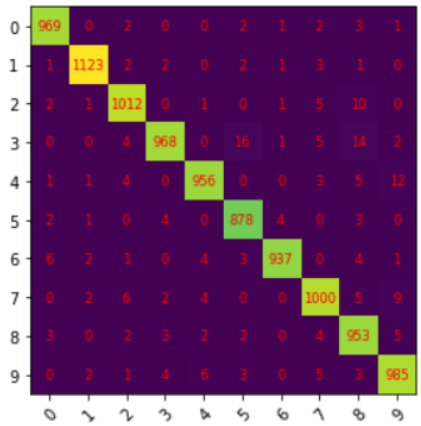


Il s'agit des paramètres que nous avons reçu par défaut. Le résultat est un peu overfitté mais l'échelle est très zoomée donc l'overfitting est très léger.

pix_p_cell 4, 8 orientation et 100 hidden neuron :

```
Test score: 0.07144611328840256
Test accuracy: 0.9781000018119812
```

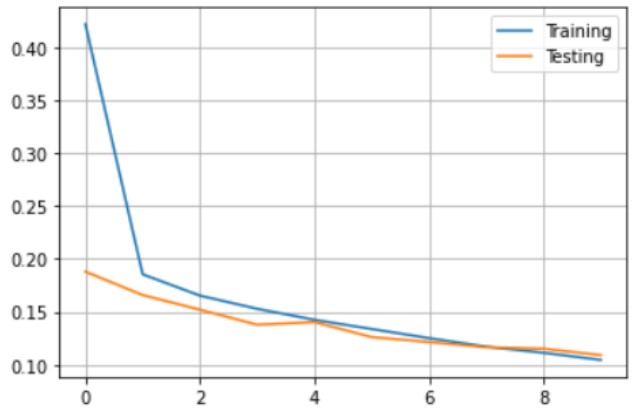


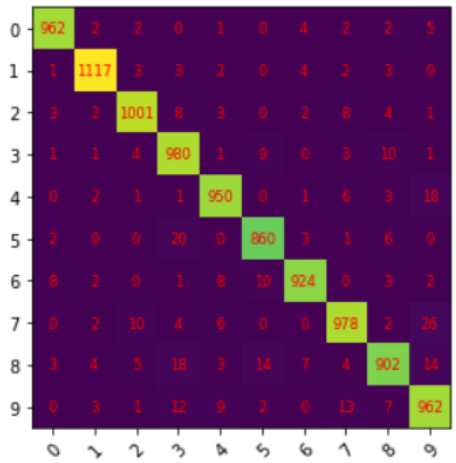


En diminuant juste le nombre de neurones, nous remarquons qu'il y a un peu moins d'overfitting mais le résultat ne change pas vraiment.

pix_p_cell 7, 8 orientation et 200 hidden neuron :

```
Test score: 0.10905688256025314
Test accuracy: 0.9635999798774719
```

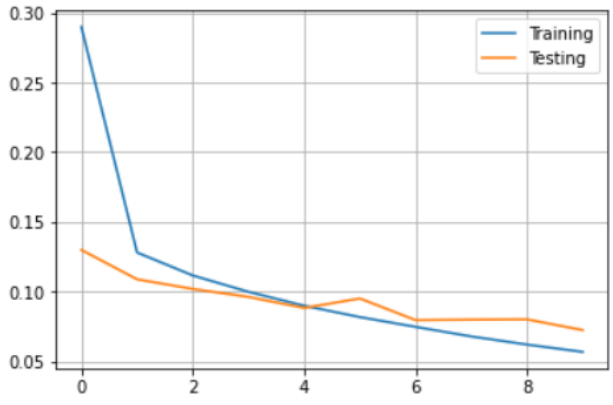


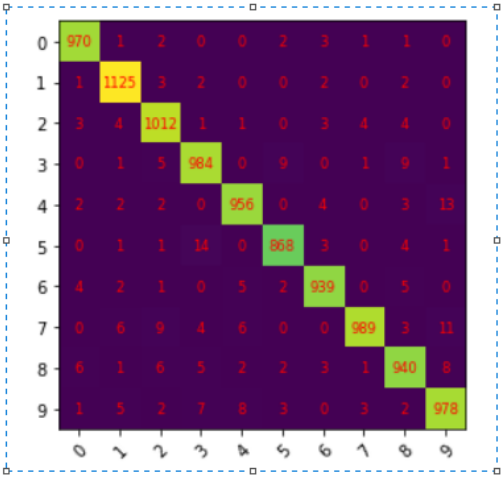


Dans ce deuxième cas, l'accuracy et le score de la loss function sont moins bons que l'expérience précédente mais nous n'avons plus d'overfitting.

pix_p_cell 4, 4 orientation et 200 hidden neuron :

```
Test score: 0.0723189115524292
Test accuracy: 0.9761000275611877
```





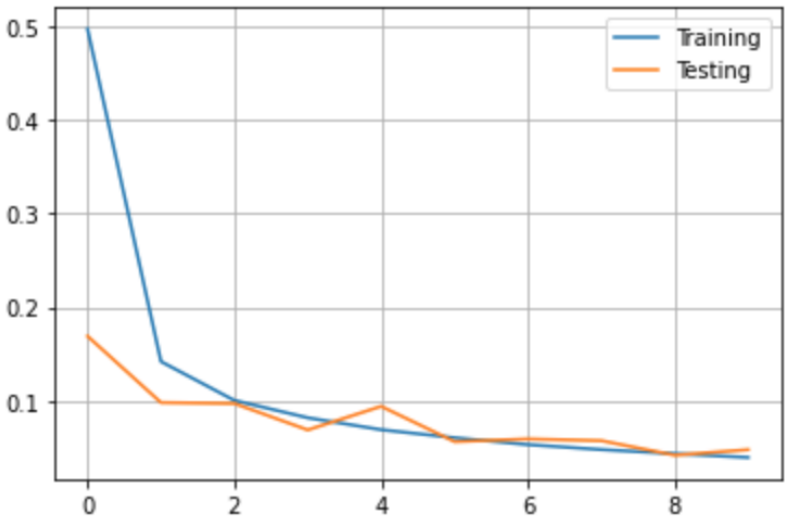
Les résultats sont un peu mieux que l'expérience précédente

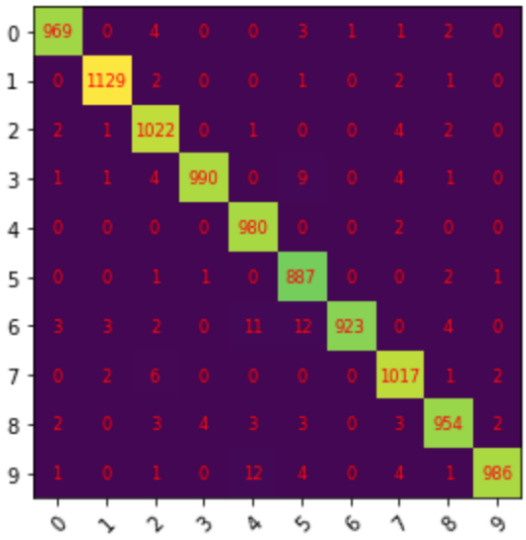
Le meilleur modèle est donc celui avec les paramètres de bases : 8 orientations, 4 pix_p_cell et 200 hidden layer. Sur ce modèle, nous voyons qu'il a le plus de mal avec 5 et 6, ce qui est assez cohérent car ces deux chiffres se ressemblent passablement.

CNN

25 neurones (par défaut):

```
Test score: 0.04756416752934456
Test accuracy: 0.9857000112533569
```

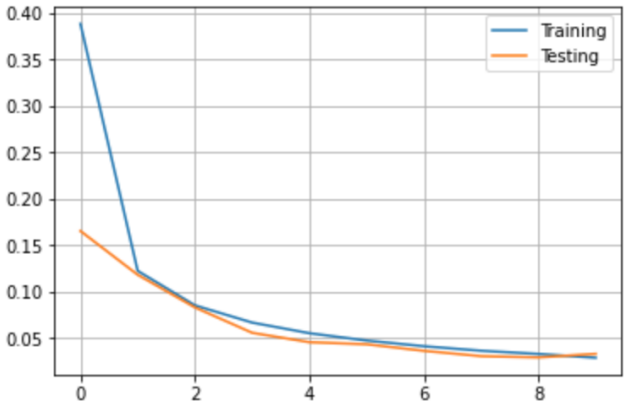


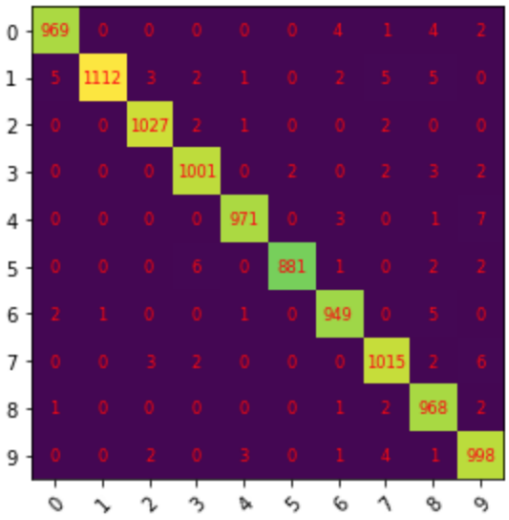


Les résultats sont vraiment bons et il ne semble pas y avoir d'overfitting.

100 neurones:

```
Test score: 0.03296331316232681
Test accuracy: 0.9890999794006348
```

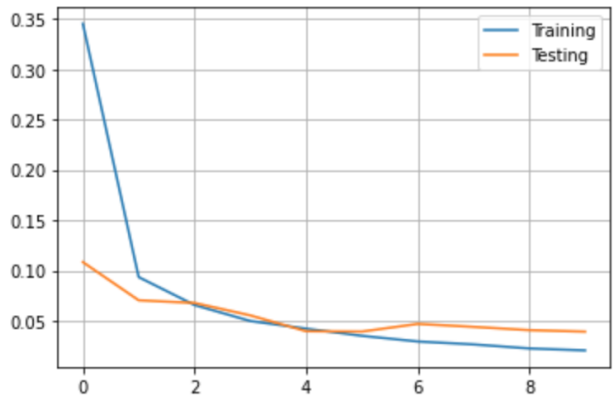


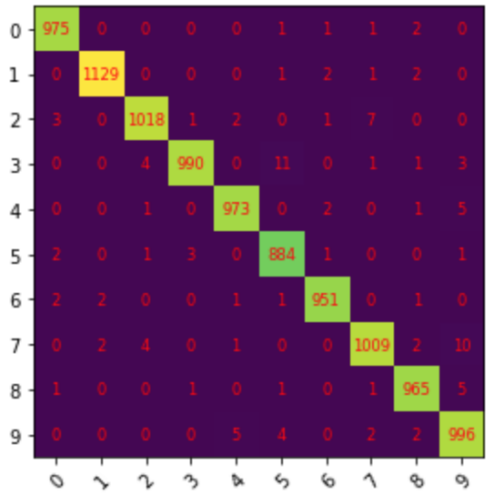


Les résultats sont encore mieux (pas de manière significative cependant) et il ne semble toujours pas y avoir d'overfitting.

400 **neurones:**

```
Test score: 0.039131008088588715
Test accuracy: 0.9890000224113464
```





Avec beaucoup plus de neurones, les résultats ne sont pas meilleurs. Nous choisissons donc 100 neurones comme résultats définitifs.

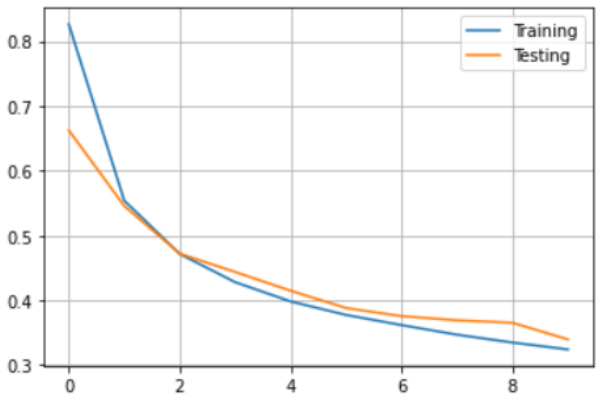
Pour conclure, nous observons que le modèle, comme les modèles précédants, a plus de mal avec les chiffres 5 et 6.

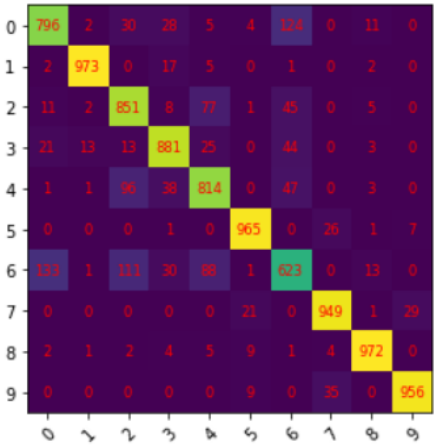
Fashion MNIST

❓ 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?

25 neurones (par défaut):

```
Test score: 0.3395996689796448
Test accuracy: 0.878000020980835
```

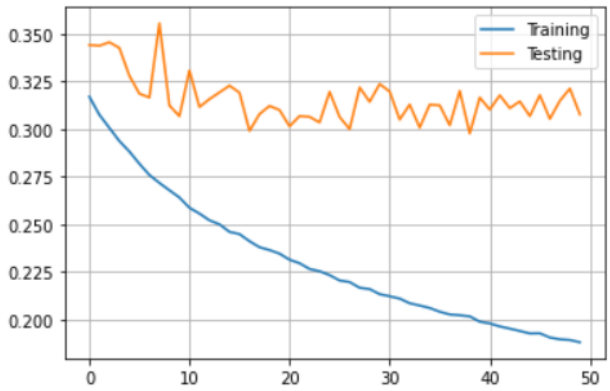


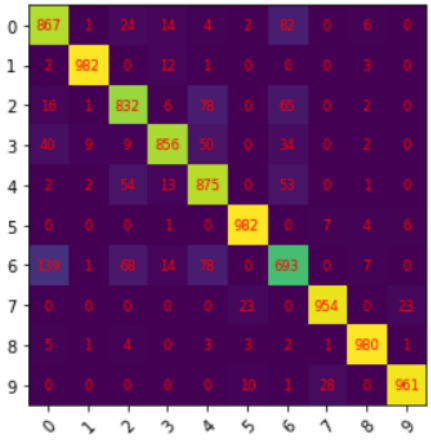


Nous remarquons que le graphe n'est clairement pas terminé, le modèle est toujours en train d'apprendre. Il faut donc augmenter le nombre d'epochs (malgré le temps d'exécution très long! Environ 20 secondes par epoch).

25 neurones mais avec 50 epochs:

```
Test score: 0.30765479803085327
Test accuracy: 0.8981999754905701
```

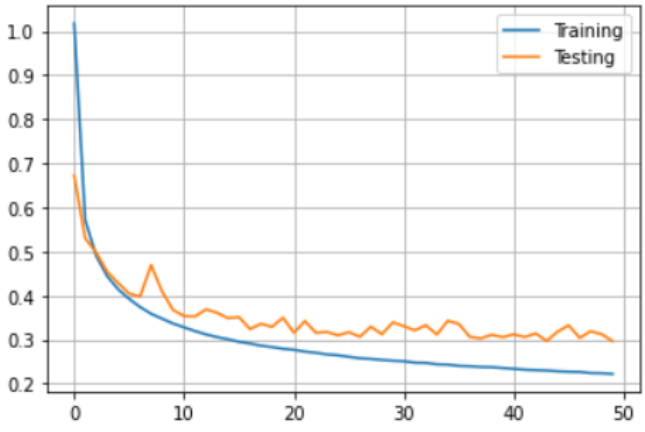


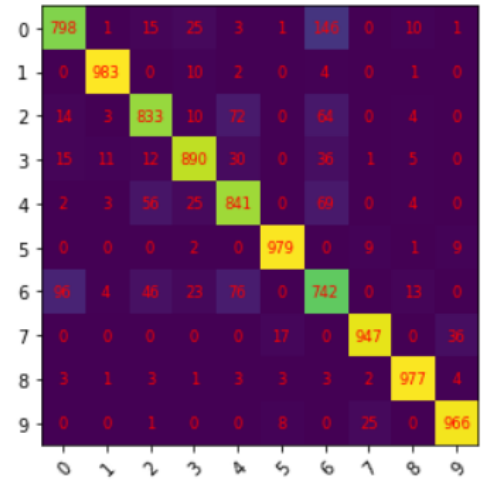


Avec bien plus d'epochs, nous remarquons bien que le modèle est vraiment overfitté. Nous allons donc essayer en le simplifiant.

10 neurones (toujours 50 epochs):

```
Test score: 0.2967931926250458
Test accuracy: 0.8956000208854675
```





Ici nous sommes toujours overfitté. Nous voyons que le training set est encore en train d'apprendre alors que le testing set n'apprend que très peu, voir commence à stagner.

C'est pourquoi nous prenons ces paramètres dans notre modèle final.

Accuracy : 0.896

```
F-score de la classe 0: 0.833
F-score de la classe 1: 0.981
F-score de la classe 2: 0.836
F-score de la classe 3: 0.886
F-score de la classe 4: 0.834
F-score de la classe 5: 0.970
F-score de la classe 6: 0.718
F-score de la classe 7: 0.946
F-score de la classe 8: 0.973
F-score de la classe 9: 0.960
```

Conclusion MNIST

Le modèle ayant les meilleurs résultats pour MNIST est donc le dernier. Dans ce dernier, nous voyons que le test score est de 0.29 tandis que de base nous étions à 0.34.

L' accuracy n'augmente pas autant nous passons de 0.87 à 0.89. Ceci dit, même si nous sommes un peu overfitté dans ce modèle final, nous en sommes plutôt satisfait.

Nous voyons que la classe n°6 est de loin celle qui est le plus dur à prédire. En effet, cette dernière est très souvent confondue avec la classe n° 0. Une autre paire de classe facilement confondable est la paire n° 2 et la n° 4 mais c'est un confusion plus légère que entre la n° 6 et la n° 0.

Ici : https://github.com/zalandoresearch/fashion-mnist

Nous pouvons voir à quoi correspondent les différentes classes. Cela nous permet de voir que la classe 0 est un Tshirt/top tandis que la 6 est un shirt. Cela explique pourquoi il y a une confusion aussi grande entre les deux classes.

La classe 2 est le pullover et la 4 le coat, il est plus difficile d'expliquer cette confusion. Cependant même s'il y a moins de ressemblance entre le pullover et le coat, il y a également une bien plus petite confusion.

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