HEIG-VD — ARN

Laboratoire 4 – Rapport

[redacted]

30 octobre 2023

1 Algorithmes, paramètres et *cost functions*

What is the learning algorithm being used to optimize the weights of the neural networks ?

Question 1

L'algorithme d'apprentissage utilisé pour optimiser les poids des différents réseaux de neurones est `RMSprop`.

What are the parameters (arguments) being used by that algorithm ?

Question 2

Les paramètres disponibles sont présentés dans le listing 1, entre autres :

- η `learning_rate`
- ρ `rho`, facteur de la moyenne mobile
- `momentum`
- ε `epsilon`, valeur considérée comme "très petite"
- `centered`, normalisation des gradients

Nous utiliserons dans ce laboratoire les paramètres par défaut ($\eta = 0.001$, $\rho = 0.9$). À l'exception de la partie d'entrainement d'un modèle pour le Fashion MNIST, où nous utiliserons un learning rate η variable.

Les équations de cet algorithme sont les suivantes sont données dans l'ensemble d'équations 1.

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

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

Ensemble d'équations 1 – Calcul des poids par RMSprop : $E\left[g^2\right]$ est la moyenne mobile des gradients carrés ; $\partial C/\partial w$ est le gradient de la fonction de coût par rapport au poids ; η est le learning rate ; ρ est le facteur de la moyenne mobile

What cost function is being used ?

Question 3

La fonction de coût utilisée est une fonction classique nommée `categorical_crossentropy`. L'équation est donnée dans l'ensemble d'équations 2.

Listing 1 – Paramètres disponibles pour l'algorithme `RMSprop`

```
keras.optimizers.RMSprop(
    learning_rate=0.001,            rho=0.9,
    momentum=0.0,                   epsilon=1e-07,
    centered=False,                 weight_decay=None,
    clipnorm=None,                  clipvalue=None,
    global_clipnorm=None,           use_ema=False,
    ema_momentum=0.99,              ema_overwrite_frequency=100,
)
```

$$L_i = -\sum_j t_{i,j} \log\left(p_{i,j}\right) \tag{3}$$

Ensemble d'équations 2 – Calcul du coût : p sont les prédictions ; t sont les cibles ; i représente le point de données ; j représente la classe

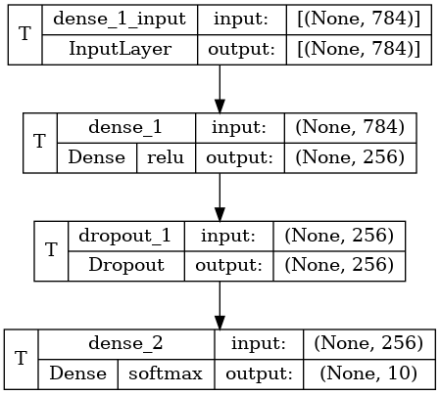


Figure 1 – Modèle utilisé pour la classification par MLP des données MNIST brutes

2 Topologies, entrées, sorties et poids pour chaque expérience

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 4

Note : Les données utilisées dans les premières expériences proviennent de la base de données MNIST (Modified National Institute of Standards and Technology). Elle contient des images de 28 pixels de côté, en noir et blanc, de chiffres (de 0 à 9) écrits à la main. La dernière expérience utilise, elle, la base de données Fashion MNIST, qui contient des images de vêtements de 10 catégories différentes. Ces images sont également au format 28 par 28 pixels et en noir et blanc.

2.1 Shallow network learning from raw data

Dans cette expérience, les images ont été traitées pixel par pixel, offrant un total de 784 pixels à traiter en entrée. Concernant les sorties, les chiffres sont classés en 10 catégories (0 à 9), chacune correspondant à une sortie dans les réseaux de neurones de nos expériences.

Le modèle que nous avons utilisé est présenté dans la figure 1. On notera que nous avons décidé d'ajouter un *dropout*, car le modèle présentait des signes d'overfitting ; cette décision sera expliquée plus en détail dans la sous-section 4.1.

Le réseau de neurones possède donc les caractéristiques suivantes :

- Nombre d'entrées : 784
- Nombre de neurones sur la couche cachée : 256
- Nombre de sorties : 10
- Nombre de poids dans la couche cachée [1] : $784 \cdot 256 + 256 = 200960$
- Nombre de poids en sortie : $10 \cdot 256 + 10 = 2570$
- Nombre total de poids : $200960 + 2570 = 203530$

1. Le nombre de poids d'une couche *dense* classique est donné par la formule #poids = #entrées · #unités + #biais. Dans notre configuration, chaque unité de la couche reçoit un biais, ainsi : #poids = #entrées · #unités + #unités

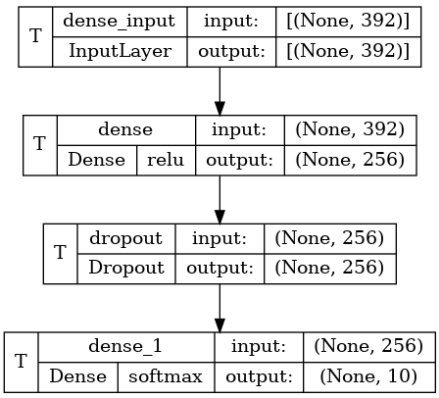


FIGURE 2 – Modèle utilisé pour la classification par MLP des données MNIST organisées par feature

2.2 Shallow network learning from features

Dans cette expérience, les images sont traitées par bloc de pixels. Nous avons décidé de garder les blocs de 4 pixels et les 8 orientations, comme paramétré au départ. L'image sera donc divisée en un total de 392 blocs différents à traiter en entrée. Concernant les sorties, les chiffres devront toujours être classés dans une des 10 catégories (0 à 9).

Le modèle que nous avons utilisé est présenté dans la figure 2. Comme pour le modèle précédent, nous avons également ajouté un *dropout*.

Le réseau de neurones possède donc les caractéristiques topologiques suivantes :

- Nombre d'entrées : 392
- Nombre de neurones sur la couche cachée : 256
- Nombre de sorties : 10
- Nombre de poids dans la couche cachée : $392 \cdot 256 + 256 = 100608$
- Nombre de poids en sortie : $10 \cdot 256 + 256 = 2570$
- Nombre total de poids : $100608 + 2570 = 103178$

2.3 CNN

Dans cette expérience, nous devons ajuster le modèle convolutif fourni pour classifier les images du dataset MNIST. La couche d'entrée prendra donc les mêmes données que l'expérience sur les données brutes, en sous-section 2.1.

Pour cette expérience, nous avons fait la décision de simplifier la topologie suggérée. En effet, après quelques expériences détaillées, présentées dans la sous-section 4.3, nous avons pu observer que le modèle obtenait un meilleur résultat avec une couche convolutive en moins. La taille des images d'entrées étant petite (28 par 28 pixels), la taille des fenêtres de convolution ont aussi été réduites à 3 par 3 pixels, contrairement au 5 par 5 initialement donné. Nous avons aussi enlevé la couche dense après les couches convolutives, pour ne garder qu'une couche pour la sortie.

Le modèle final que nous avons utilisé est disponible dans la figure 3. On y retrouve deux couches convolutives possédant, respectivement, 32 et 64 filtres. Chaque couche convolutive reçoit une couche de *pooling*, divisant ainsi les dimensions de l'espace d'entrée par deux à chaque couche.

Le réseau de neurones possède donc les caractéristiques topologiques suivantes :

- Dimensions de la couche d'entrée : $(28, 28, 1)$
- Couches convolutives :
  - Première couche [2] : $32 \cdot 3 \cdot 3 \cdot 1 + 32 = 320$ poids. Dimensions : $(28, 28, 32)$; après pooling : $(14, 14, 32)$
  - Seconde couche : $64 \cdot 3 \cdot 3 \cdot 32 + 64 = 18496$ poids. Dimensions : $(14, 14, 64)$; après pooling : $(7, 7, 64)$
- Aplanissement [3] : $7 \cdot 7 \cdot 64 = 3136$
- Couche dense : $10 \cdot 3136 + 10 = 31370$ poids
- Total des poids : $320 + 18496 + 31370 = 50186$

2. Le nombre de poids d'une couche convolutive à deux dimensions est donné par la formule #poids = #unités · #`kernel_size[0]` · #`kernel_size[1]` · #channels + #biais. Dans notre configuration, nous utilisons une fenêtre de convolution carrée et chaque unité reçoit un biais, ainsi : #poids = #unités · #`kernel_size` · #`kernel_size` · #channels + #unités. Les channels correspondent aux nombres d'unités de la dernière couche, représenté en troisième dimension.

3. La formule de l'aplanissement est une simple multiplication des dimensions de la couche précédente

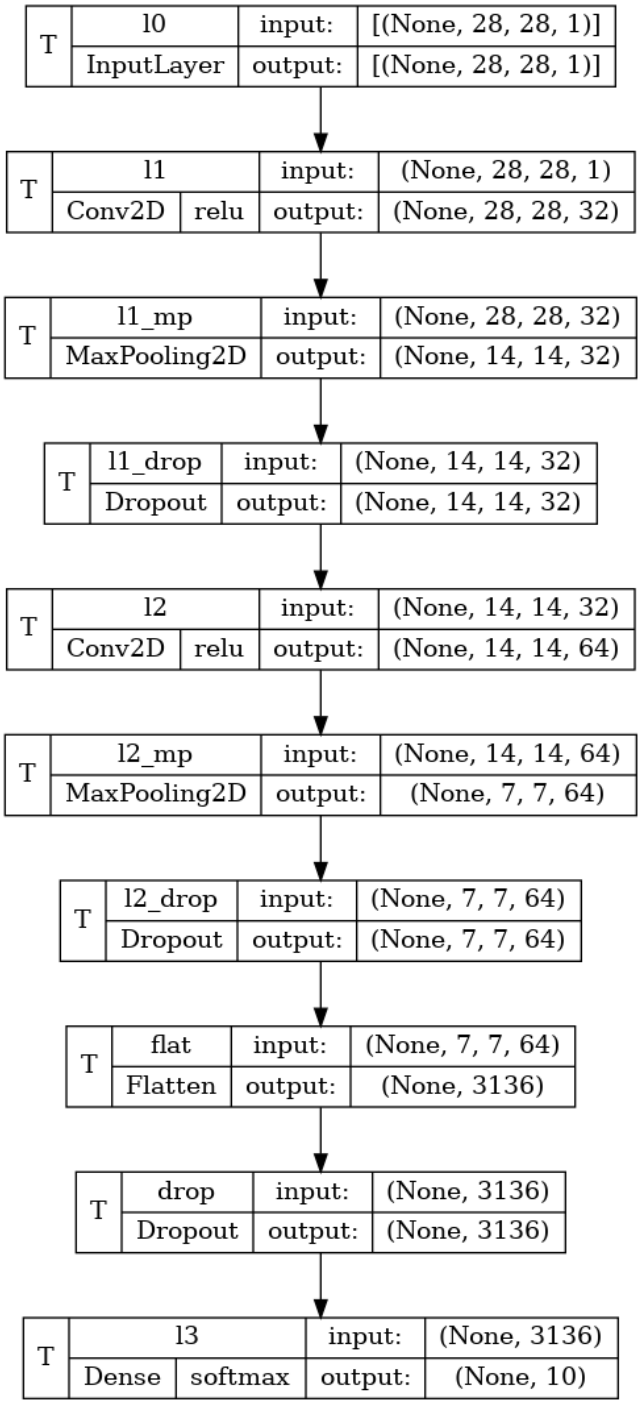


FIGURE 3 – Modèle utilisé pour la classification par CNN des données MNIST

2.4 Fashion MNIST

Cette dernière expérience va chercher à classifier, au moyen d'un réseau convolutif, des images au même format que celles de MNIST (28 par 28 pixels), mais dont les images sont des photos de vêtements vendus sur le site Zalando. Ce dataset est conçu pour être un remplacement plus difficile à classifier que celui du MNIST, généralement considéré comme trop simple. Ainsi, la dimension d'entrée reste la même.

Ce modèle étant plus complexe, nous avons rajouté une couche dense entre l'aplanissement des couches convolutives et la couche dense de sortie. Le reste du modèle, présenté dans la figure 4, est essentiellement le même que pour celui du dataset MNIST décrit dans la section précédente.

Le nombre de paramètres final reste sensiblement élevé ; nous le justifions par la complexité ajoutée du modèle Fashion par rapport au modèle standard. En effet, en reprenant directement le modèle CNN, nous avons obtenu des résultats peu satisfaisants, alors que pour le dataset standard, ils étaient très bons. La couche dense de 128 unités rajoute un nombre important de poids à déterminer, mais elle nous parait importante pour avoir une classification correcte.

Le réseau de neurones possède donc les caractéristiques topologiques suivantes :

- Dimensions de la couche d'entrée : $(28, 28, 1)$
- Couches convolutives :
  - Première couche : $32 \cdot 3 \cdot 3 \cdot 1 + 32 = 320$ poids. Dimensions : $(28, 28, 32)$; après pooling : $(14, 14, 32)$
  - Seconde couche : $64 \cdot 3 \cdot 3 \cdot 32 + 64 = 18496$ poids. Dimensions : $(14, 14, 64)$; après pooling : $(7, 7, 64)$
- Aplanissement : $7 \cdot 7 \cdot 64 = 3136$
- Première couche dense : $128 \cdot 3136 + 128 = 401536$ poids
- Seconde couche dense : $10 \cdot 128 + 10 = 1290$ poids
- Total des poids : $320 + 18496 + 401536 + 1290 = 421642$

3 Capacité des réseaux profonds vs peu profonds en termes de poids

> Do the deep neural networks have much more « capacity » (i.e., do they have more weights ?) than the shallow ones ?
>
> Question 5

La capacité des réseaux de neurones, en termes de poids, est déterminée simultanément par leur profondeur (nombre de couches) et leur largeur (nombre de neurones par couche). Bien que les réseaux de neurones profonds comportent plusieurs couches et que les réseaux peu profonds ne comportent qu'une ou deux couches cachées, il n'est pas nécessairement vrai que les réseaux profonds ont toujours un plus grand nombre de poids que les réseaux peu profonds.

Par exemple, un réseau de neurones profond comportant de nombreuses couches, mais relativement peu de neurones par couches, peut présenter moins de poids qu'un réseau peu profond comportant une seule grande couche cachée. À l'opposée, un réseau profond avec de nombreux neurones dans chaque couche pourrait posséder plus de poids qu'un réseau peu profond.

En conclusion, la relation entre la profondeur d'un réseau de neurones et sa capacité, mesurée par le nombre de poids, est complexe. La capacité dépend autant de la profondeur que de la largeur du réseau. Il est donc essentiel de tenir compte de la configuration spécifique des neurones et des couches lorsque l'on compare les capacités des réseaux profonds et peu profonds.

On ajoutera qu'il n'y a pas forcément une corrélation entre la capacité et la performance du système ; comme nous le discuterons plus loin, notre modèle de classification par MLP pour les données MNIST brutes contient plus de 200'000 paramètres et obtient une précision de 98.21%, alors que le modèle convolutif pour les mêmes données brutes obtient une précision de 99.26% avec uniquement 50'186 paramètres. Ce qui est logique : un modèle convolutif est capable d'identifier des parties d'une image qu'une couche dense aurait beaucoup plus de difficulté à observer.

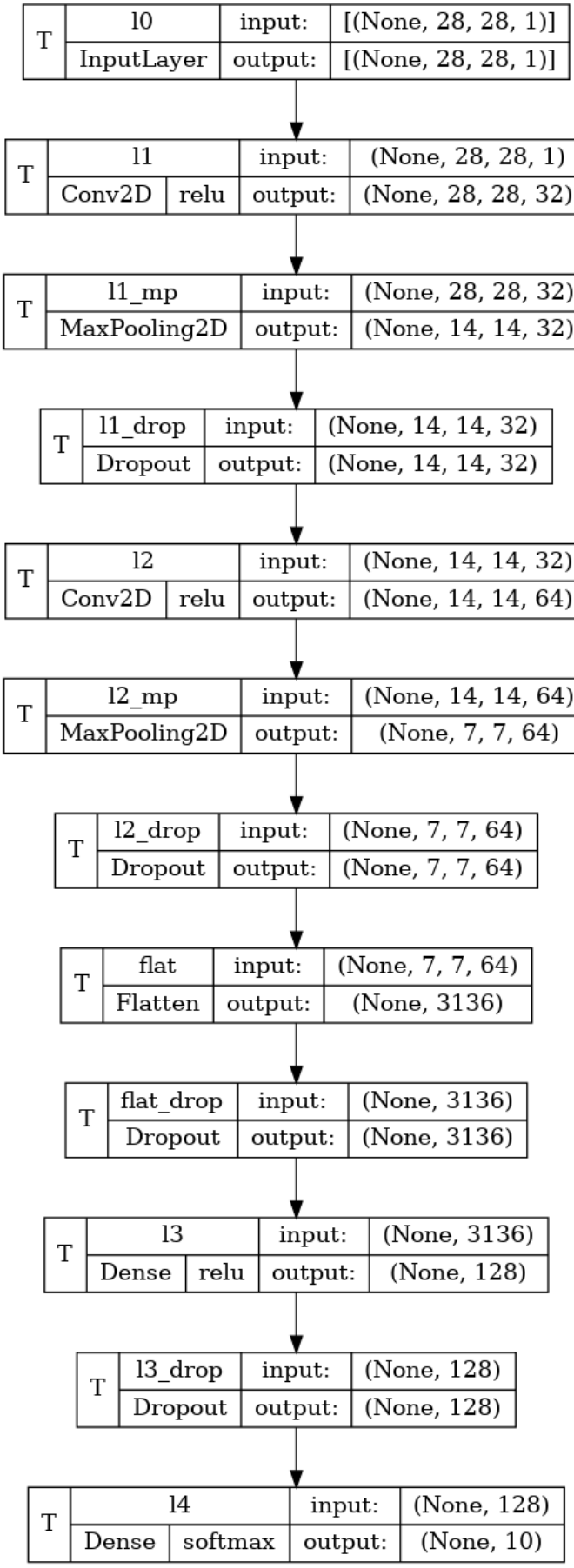
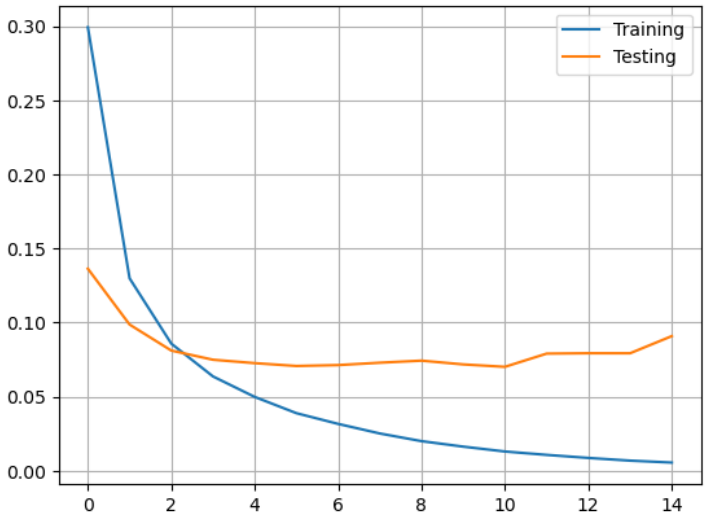


FIGURE 4 – Modèle utilisé pour la classification par CNN des données Fashion MNIST

```
Test score:    0.0857
Test accuracy: 0.9792
```



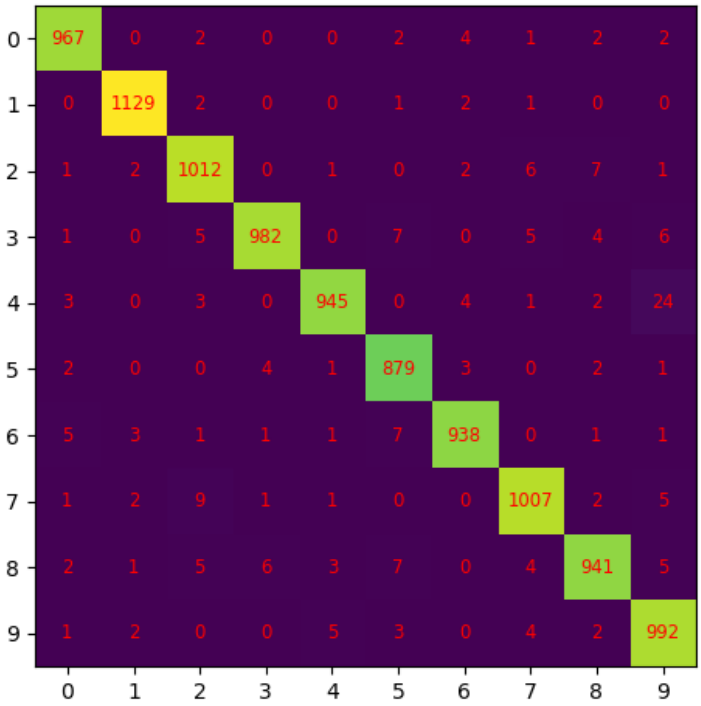


FIGURE 5 – Courbes de l'évolution des MSE et matrice de confusion [4] pour le modèle initial du réseau de neurone superficiel établi à partir des données brutes. On note ici une présence d'overfitting.

4 Cas significatifs, performances et chiffres confondus

> 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 6

4.1 Shallow network learning from raw data

Le modèle initial contient 300 neurones dans la couche cachée et pas d'effet *dropout*, un premier essai a donné des résultats corrects avec une précision de 97.92% présenté dans la figure 5.

Après un premier essai avec les 10 epochs initiales, nous avons constaté que le modèle avait encore de la place pour converger. Nous avons donc augmenté le nombre d'epochs à 15 pour le reste de cette expérience.

On constate dans les résultats initiaux un overfitting important que nous allons étudier en priorité.

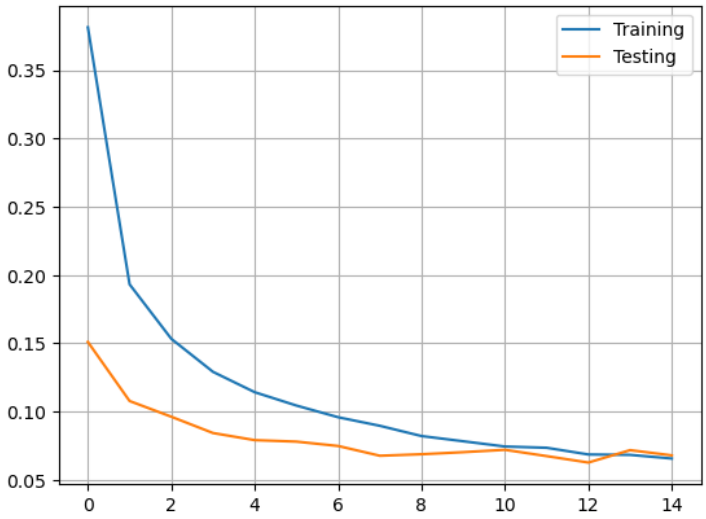
4.1.1 Introduction d'un effet *dropout*

Pour combattre l'overfitting, nous avons décidé d'implémenter en premier temps un *dropout* après la première couche dense. Nous faisons un essai avec un *dropout* à 0.5, ce qui signifie que 50% des neurones seront aléatoirement désactivés à chaque évolution.

Les résultats, affichés dans la figure 6, confirment notre théorie : on remarque immédiatement une amélioration de la précision à 98.20%.

4. **Précision** : les valeurs en colonnes sont les valeurs prédites, les valeurs en lignes sont les valeurs réelles.

```
Test score:    0.0716
Test accuracy: 0.9820
```



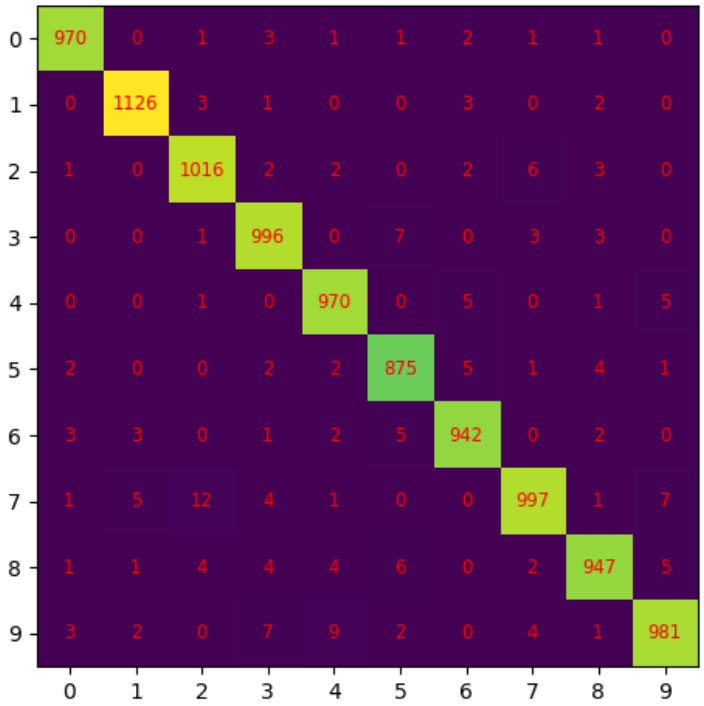


FIGURE 6 – Courbes de l'évolution des MSE et matrice de confusion pour le second modèle du réseau de neurone superficiel établi à partir des données brutes. Ce modèle possède 300 neurones et un effet *dropout* de 50% dans la couche cachée.

4.1.2 Simplification du modèle

Nous avons maintenant un modèle satisfaisant, mais il est assez complexe. On décide donc de changer la quantité de neurones de la couche cachée à 128 pour un premier essai.

On remarque dans les graphes de la figure 7 que le modèle a plus de difficulté à converger vers une solution, alors que l'erreur de test reste constamment inférieure à l'erreur d'entrainement. Cela pourrait être attribué au fait que les données de tests sont plus simples que les données d'entrainement. Cependant, cette hypothèse ne nous semble pas très plausible, car le modèle avait moins de difficulté avec un nombre plus élevé de neurones.

4.1.3 Modèle final

Nos hypothèses par rapport à l'écart entre les erreurs d'entrainement et de test découvert lors de l'essai précédent pouvait être dû à un effet *dropout* trop fort, ou un nombre de neurones insuffisants.

Nous avons fait un premier essai en réduisant l'effet *dropout* à 40%, ce qui a aidé le modèle, mais n'a pas amélioré les résultats. C'est pourquoi nous n'allons pas développer plus cet essai dans ce rapport.

Néanmoins, un essai concluant a été de garder cette valeur de *dropout* à 40% et d'augmenter le nombre de neurones de la couche cachée. Avec ce nouvel essai, nous avons pu obtenir une précision de 98.21%.

Les résultats de ce dernier modèle, dans la figure 8, affichent une très légère tendance d'overfitting vers les deniers epochs. Cela n'a pour autant pas empêché le modèle à se généraliser et la matrice de confusion affiche très peu d'erreurs.

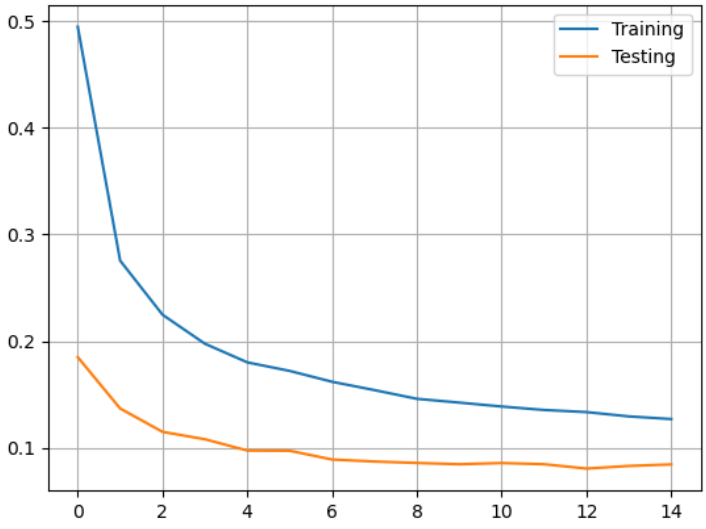
4.1.4 Conclusions

Après ces nombreux essais, nous avons décidé de garder le modèle à 256 neurones et l'effet *dropout* à 45%. La précision finale étant comparable au modèle présenté en sous-section 4.1.1, nous avons établi notre décision sur le nombre inférieur de neurones impliquant un nombre inférieur de poids à paramétrer.

On remarque dans les matrices de confusions présentées au long de cette expérience que les paires de chiffres $(2, 7)$ et $(4, 9)$ sont celles qui sont les plus souvent confondues. Ces deux confusions ne sont plus visibles dans le dernier modèle ; la matrice de confusion, comme dit précédemment, a une répartition beaucoup plus variée des erreurs. En effet, les f-score finaux sont tous à environ 0.98, ce qui est globalement très bon [5].

5. Un f-score de 1.00 représenterait une classification exacte

```
Test score:    0.0956
Test accuracy: 0.9756
```



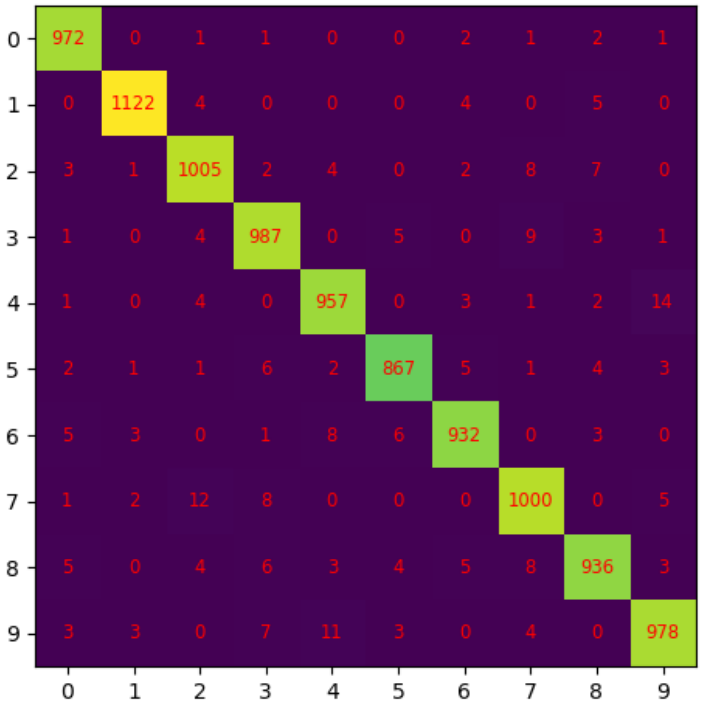
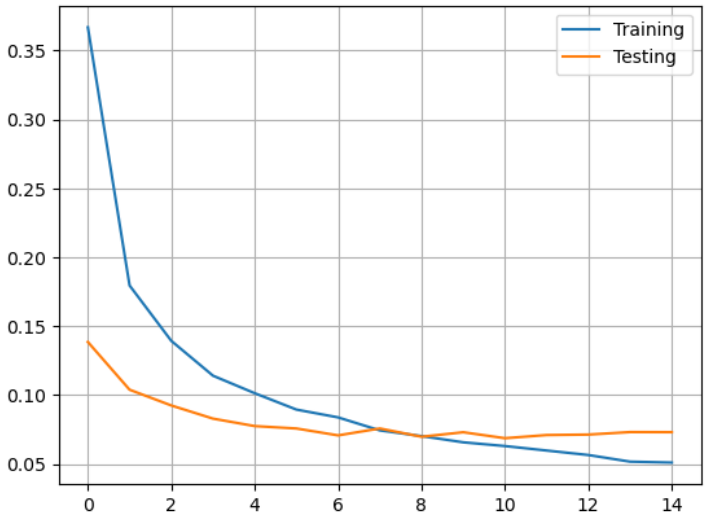


FIGURE 7 – Courbes de l'évolution des MSE et matrice de confusion pour le second modèle du réseau de neurone superficiel établi à partir des données brutes. Ce modèle possède 128 neurones et un effet *dropout* de 50% dans la couche cachée.

```
Test score:    0.0698
Test accuracy: 0.9821
```



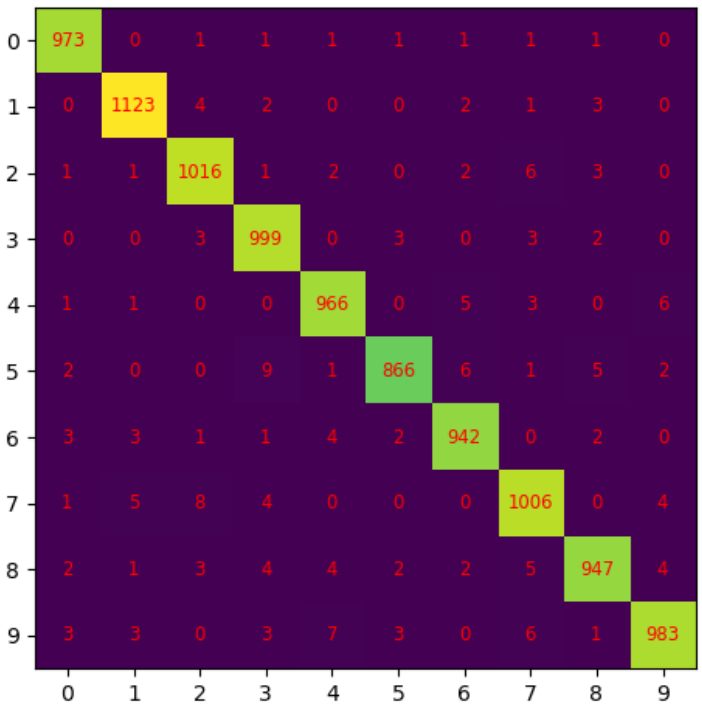


FIGURE 8 – Courbes de l'évolution des MSE et matrice de confusion pour le second modèle du réseau de neurone superficiel établi à partir des données brutes. Ce modèle possède 256 neurones et un effet *dropout* de 40% dans la couche cachée.

```
Test score:    0.0672
Test accuracy: 0.9787
```

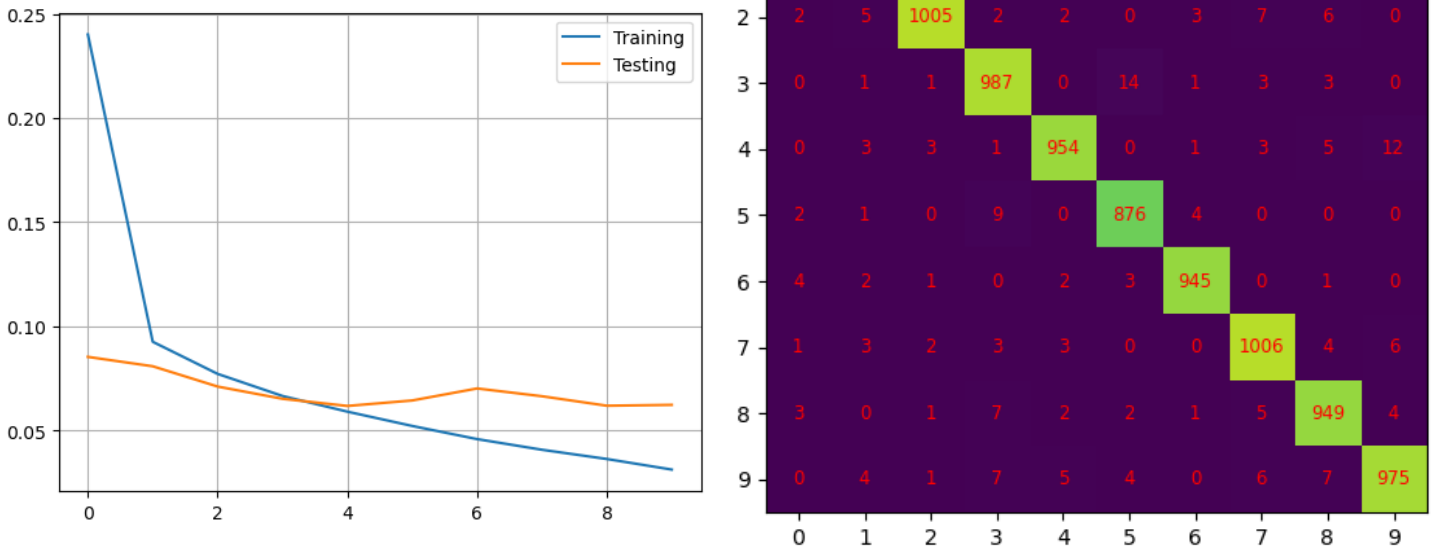


FIGURE 9 – Courbes de l'évolution des MSE et matrice de confusion pour le modèle initial du réseau de neurone superficiel établi à partir des données par feature.

4.2 Shallow network learning from features

Le modèle initial était paramétré avec 4 `pix_p_cell` , 8 orientations, 200 neurones dans la couche cachée, et 10 epochs. On peut remarquer que les résultats de ce modèle, présentés dans la figure 9 ne sont pas encore idéaux, avec un début d'overfitting et plusieurs confusions.

4.2.1 Introduction d'un *dropout*

Comme dans l'expérience précédente, nous avons décidé d'ajouter un effet *dropout* initialement paramétré à 40% pour tenter de réduire l'overfitting. Nous prenons aussi un raccourci en incrémentant le nombre de neurones de la couche cachée à 256, après avoir testé les deux séparément. On garde les paramètres des features tels quels.

Cela nous a donné un résultat satisfaisant pour un premier essai. Comme indiqué dans la figure 10, nous obtenons 98.22% de précision, ce qui est déjà très bien. Ce modèle confond principalement la paire de chiffres $(5, 3)$; cela nous semble logique, car ce sont deux chiffres qui peuvent être très similaires dans une écriture manuscrite. Nous décidons donc de garder la topologie du modèle identique pour le reste des essais, afin de garder une base fixe et de déterminer quels sont les paramètres de features idéaux.

4.2.2 4 `pix_p_cell` , 4 orientations

Nous continuons nos essais avec un changement du nombre d'orientations, en conservant le reste des hyper-paramètres identiques. Après plusieurs essais, nous constatons, à l'aide des graphes de la figure 11 que les résultats ne sont pas aussi bons que pour la section précédente.

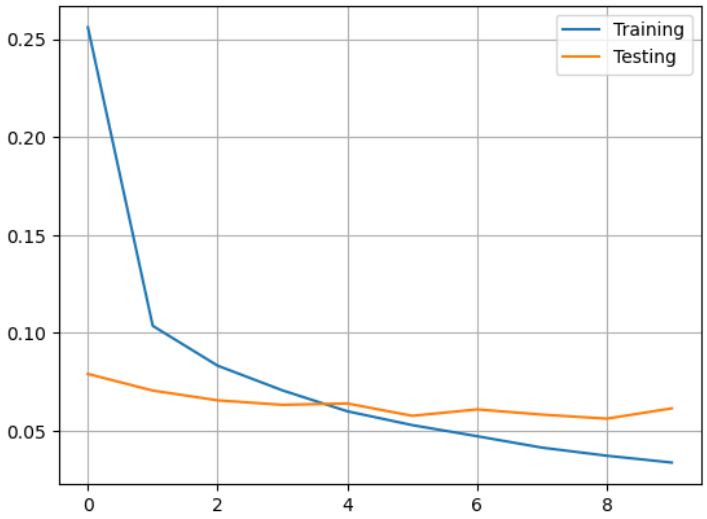
Ce modèle confond également plus de paires de chiffres que le modèle précédent, avec $(3, 8)$, $(4, 9)$, $(5, 3)$, et $(9, 3)$ étant les plus confondues.

4.2.3 7 `pix_p_cell` , 8 orientations

Une autre tentative est faite en utilisant le même nombre d'orientations que le premier essai, mais en augmentant le nombre de `pix_p_cell` à 7. Ce modèle ne performe pas mieux que les deux autres avec une précision à 96.28% seulement et un nombre accru de paires de chiffres confondues.

On pourrait émettre l'hypothèse, selon les graphes de la figure 12, que le modèle aurait besoin d'un nombre plus élevé de neurones ou alors un *dropout* inférieur. Nous avons toutefois décidé de continuer à essayer sur la base du premier essai concluant avec 4 `pix_p_cell` et 8 orientations, car nous voulons garder une topologie simple et le nombre de confusions laisse penser que les paramètres ne distinguent pas assez bien les features des images d'entrée.

```
Test score:    0.0622
Test accuracy: 0.9822
```



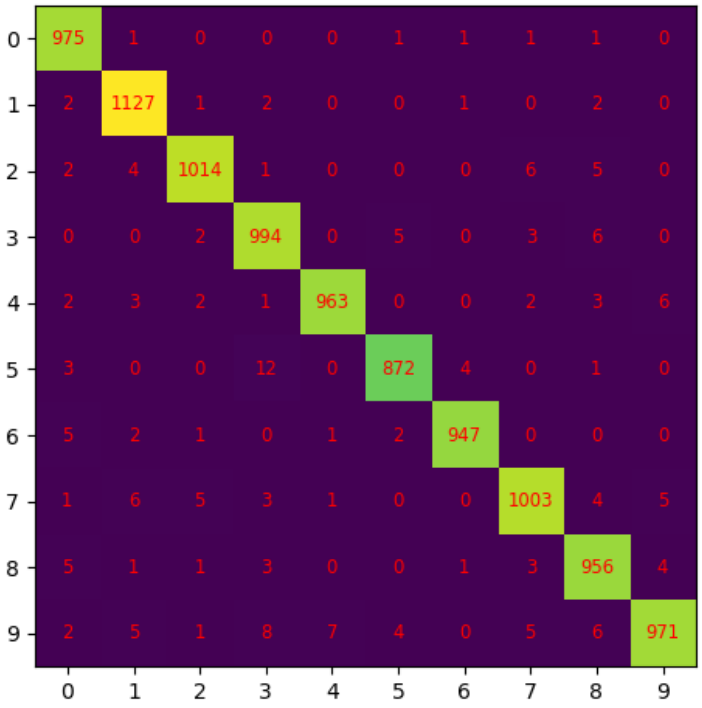
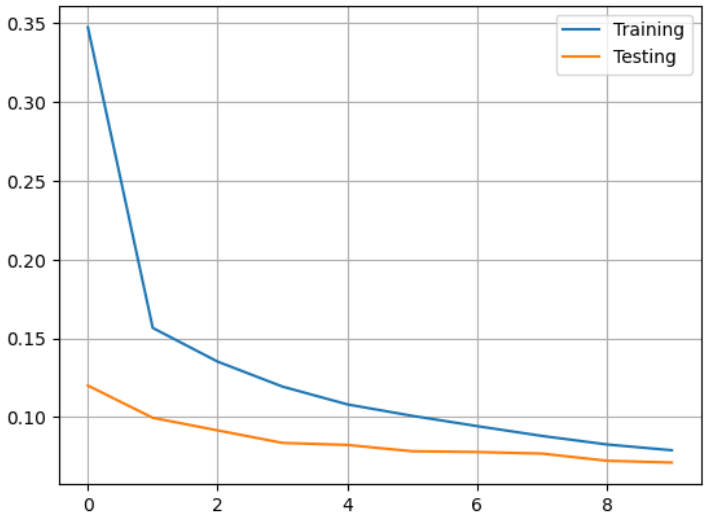


FIGURE 10 – Courbes de l'évolution des MSE et matrice de confusion pour le second modèle du réseau de neurone superficiel établi à partir des données par feature. Ce modèle possède 256 neurones et un effet *dropout* de 40% dans la couche cachée. Les paramètres des features pour entrainer ce modèle étaient de 4 `pix_p_cell` et 8 orientations.

```
Test score:    0.0759
Test accuracy: 0.9757
```



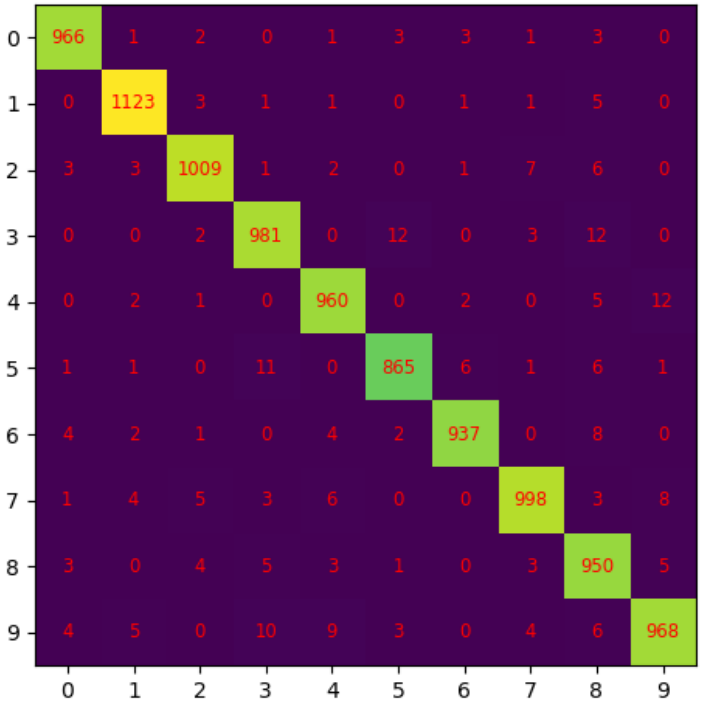


FIGURE 11 – Courbes de l'évolution des MSE et matrice de confusion pour le second modèle du réseau de neurone superficiel établi à partir des données par feature. Ce modèle possède 256 neurones et un effet *dropout* de 40% dans la couche cachée. Les paramètres des features pour entrainer ce modèle étaient de 4 `pix_p_cell` et 4 orientations.

```
Test score:    0.1110
Test accuracy: 0.9628
```

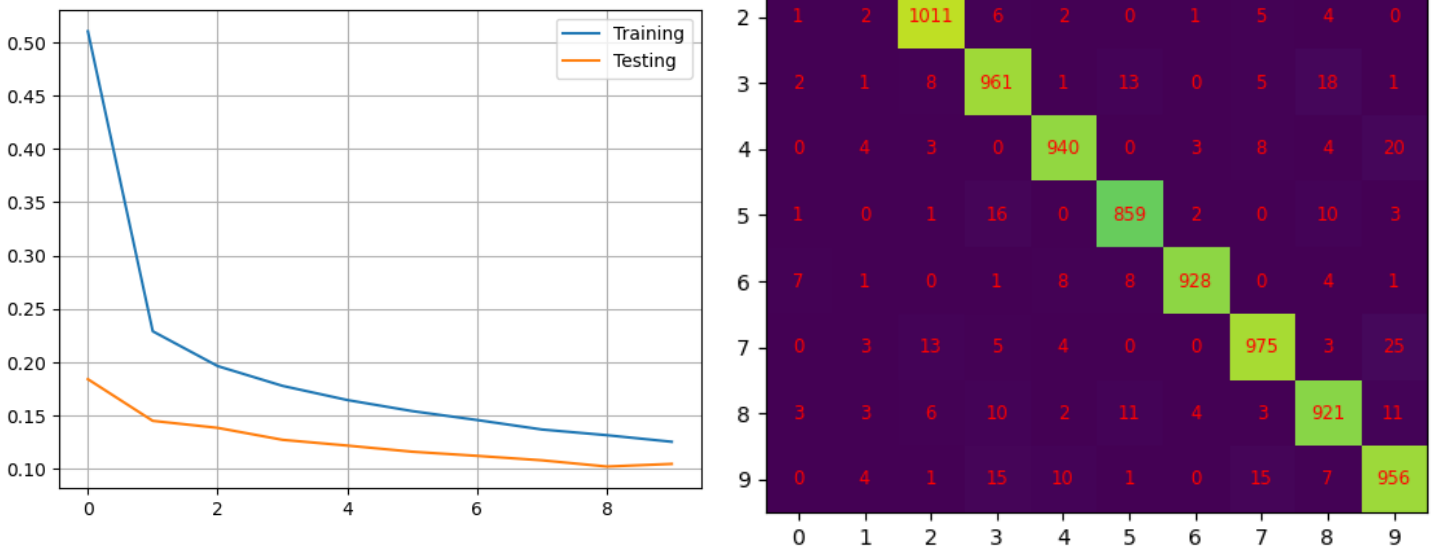


FIGURE 12 – Courbes de l'évolution des MSE et matrice de confusion pour le second modèle du réseau de neurone superficiel établi à partir des données par feature. Ce modèle possède 256 neurones et un effet *dropout* de 40% dans la couche cachée. Les paramètres des features pour entrainer ce modèle étaient de 7 `pix_p_cell` et 8 orientations.

4.2.4 7 `pix_p_cell` , 4 orientations

Étant donné le résultat précédent, nous avons quand même essayé de réduire le nombre d'orientations, en imaginant que ça aiderait peut-être le modèle. Nous avions cependant certains doutes quant à cette hypothèse.

Comme imaginé, le modèle a une performance encore plus médiocre que le modèle de la section précédente avec une précision à 93.86%. Nous avons quand même inclus les résultats dans la figure 13. Les résultats laissent cependant entrevoir une possibilité d'amélioration si l'on change la topologie du réseau, augmentant sa complexité, mais celui-ci pourrait afficher des meilleurs résultats.

4.2.5 Conclusion

En réponse à ces essais, nous avons décidé de garder le premier modèle, présenté en sous-section 4.2.1. Ce modèle a une précision relativement bonne, avec des f-score variant de 0.97 à 0.99, ce qui est très correct et prouve que les chiffres sont bien distingués, à l'exception de la paire $(3, 5)$ qui a plus de peine.

4.3 CNN

Le modèle initial est fourni avec 3 couches convolutives ayant, respectivement, 9, 9, et 16 filtres. Les deux premières ont une fenêtre de convolution de 5 par 5, alors que la dernière a une fenêtre de 3 par 3. Chaque couche convolutive est accompagnée d'une couche de pooling avec une taille d'observation de 2 par 2 pixels. Après l'aplanissement, il y a une couche dense cachée avec 25 neurones et la couche de sortie. Ce modèle ne possède pas d'effet *dropout* dans cette configuration.

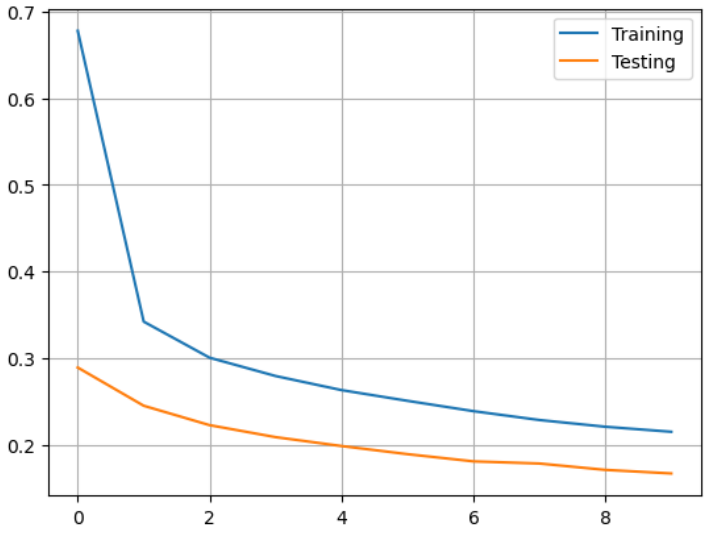
Les résultats de ce modèle initial sont présentés dans la figure 14. La performance est déjà très bonne avec une précision de 98.92% pour un premier essai, mais nous pensons que le modèle peut être simplifié.

4.3.1 Première simplification du modèle

Le modèle n'ayant aucune difficulté à converger, nous avons décidé de simplement retirer la couche cachée de la partie feed-forward du modèle.

La performance de ce second modèle est comparable à celle du modèle précédent avec une précision de 98.60%. On remarque toutefois qu'il y a plusieurs paires qui sont confondues. Nous avons donc décidé de retravailler les couches convolutives pour mieux distinguer les parties importantes du dataset qui vont aider la classification.

```
Test score:    0.1865
Test accuracy: 0.9386
```



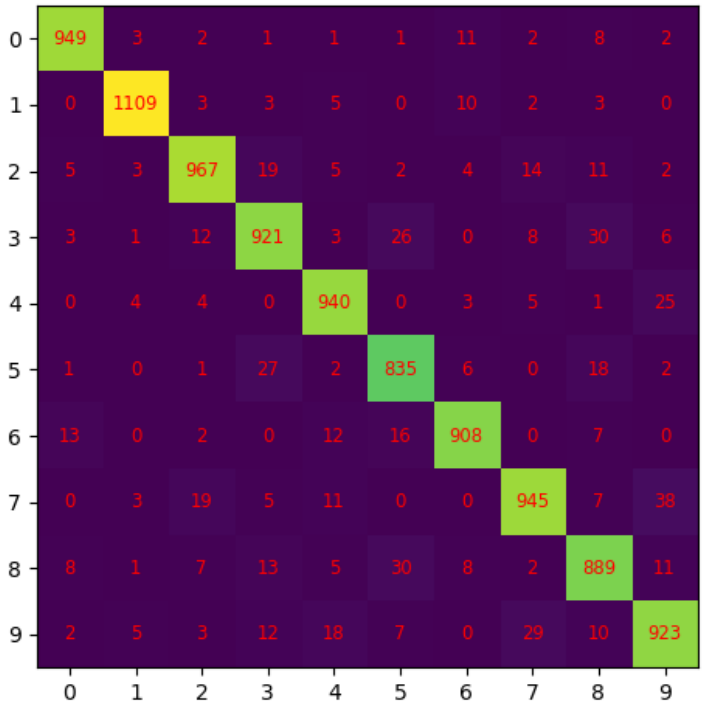
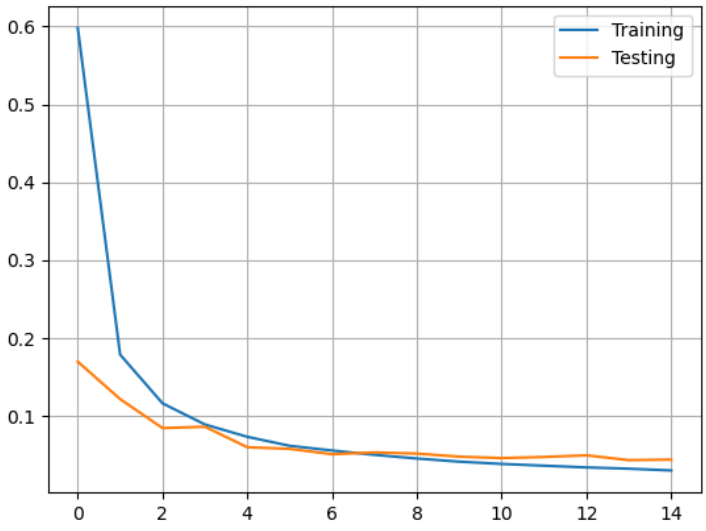


FIGURE 13 – Courbes de l'évolution des MSE et matrice de confusion pour le second modèle du réseau de neurone superficiel établi à partir des données par feature. Ce modèle possède 256 neurones et un effet *dropout* de 40% dans la couche cachée. Les paramètres des features pour entrainer ce modèle étaient de 7 `pix_p_cell` et 4 orientations.

```
Test score:    0.0332
Test accuracy: 0.9892
```



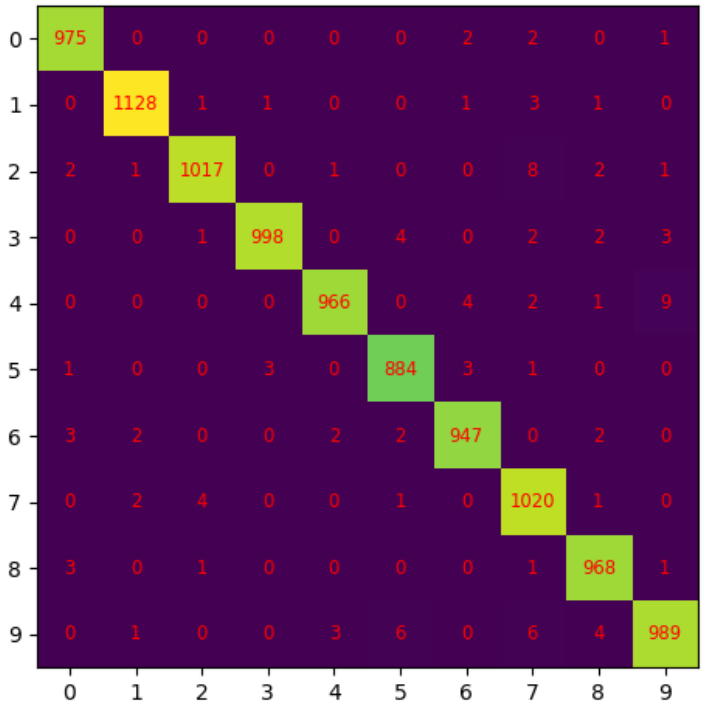


FIGURE 14 – Courbes de l'évolution des MSE et matrice de confusion pour le modèle initial du réseau de neurones convolutif

```
Test score:    0.0421
Test accuracy: 0.9860
```

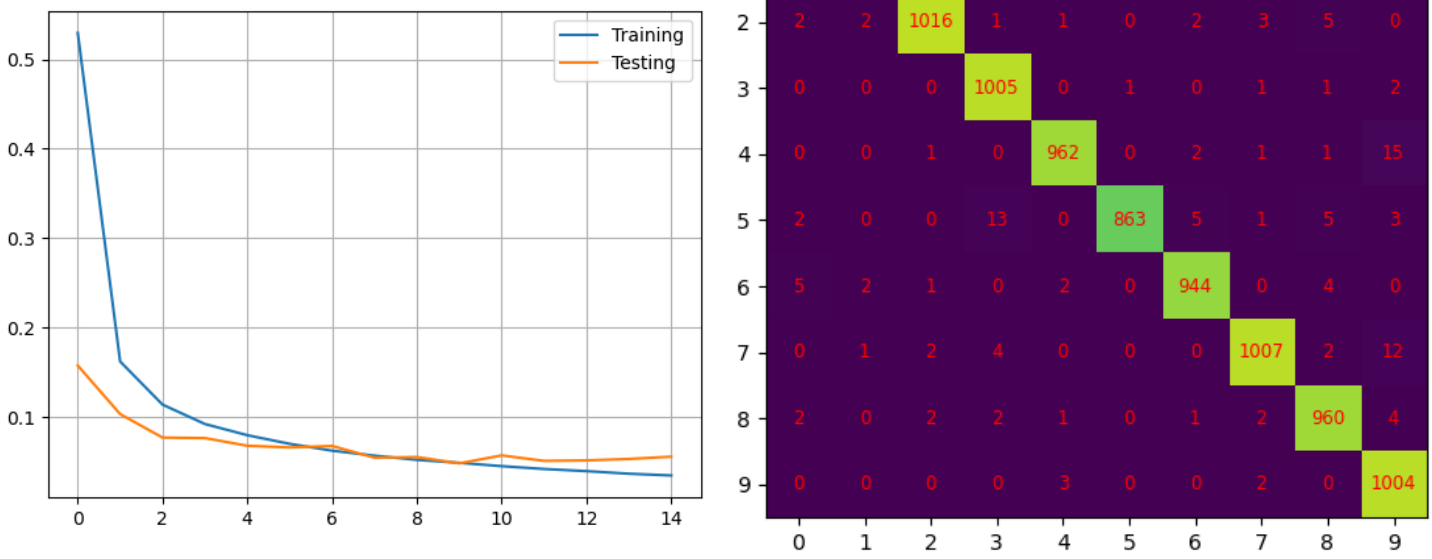


FIGURE 15 – Courbes de l'évolution des MSE et matrice de confusion pour le second modèle convolutif. Ce modèle contient les mêmes couches convolutives que celui de la figure 14 et ne possède pas de couche dense entre l'aplanissement et la couche de sortie.

4.3.2 Changement des hyper-paramètres de convolution

Pour améliorer les filtres des couches convolutives, nous avons décidé d'augmenter le nombre d'unités de chaque filtre, ainsi que réduire la taille des fenêtres des couches de convolution. Après quelques évolutions du modèle en changeant les paramètres un après un, nous avons obtenu la topologie suivante :

- Première couche convolutive : 16 filtres, fenêtre 3 par 3 pixels, pooling 2 par 2
- Deuxième couche convolutive : 32 filtres, fenêtre 3 par 3 pixels, pooling 2 par 2
- Aplanissement
- Couche dense de sortie

On observe dans les résultats de ce modèle, présentés dans la figure 16, que le modèle a une précision à 98.75% comparable à celle du modèle initial, mais développe un overfitting à partir de la $8^{\text{ème}}$ epoch. Il est notable que ce modèle a moins de confusions que les deux modèles précédents, avec des f-scores tous supérieurs à 0.98.

4.3.3 Introductions d'effets *dropout*

Comme nous avons observé un léger overfitting dans le dernier essai, nous avons ajouté des effets *dropout* après chaque couche du modèle. La topologie de ce dernier est celle présentée dans la sous-section 2.3.

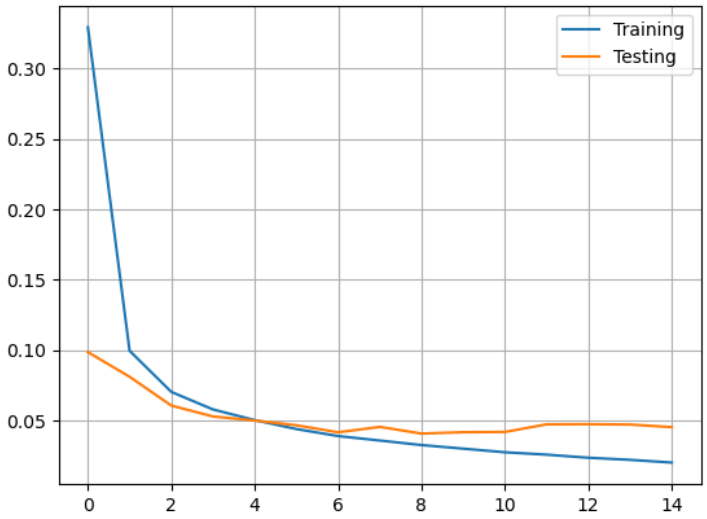
Les résultats de cet essai sont présentés dans la figure 17. Ce modèle a une précision de 99.26%, un score difficile à améliorer avec nos stratégies actuelles.

Les f-score sont tous à 0.99, indiquant très peu de confusions ; on remarque néanmoins que les deux paires les plus confondues sont $(2, 7)$ et $(4, 9)$. Ces confusions nous semblent raisonnables, car les formes de ces chiffres se ressemblent beaucoup.

4.3.4 Conclusions

Nous choisissons donc le dernier modèle présenté comme modèle final pour cette expérience. Ce modèle a quelques difficultés pour les paires de chiffres $(2, 7)$ et $(4, 9)$, mais la performance est globalement très bonne pour tous les chiffres du dataset MNIST. Nous sommes donc satisfaits de ce modèle et des essais que nous avons effectués jusqu'ici.

```
Test score:    0.0365
Test accuracy: 0.9875
```



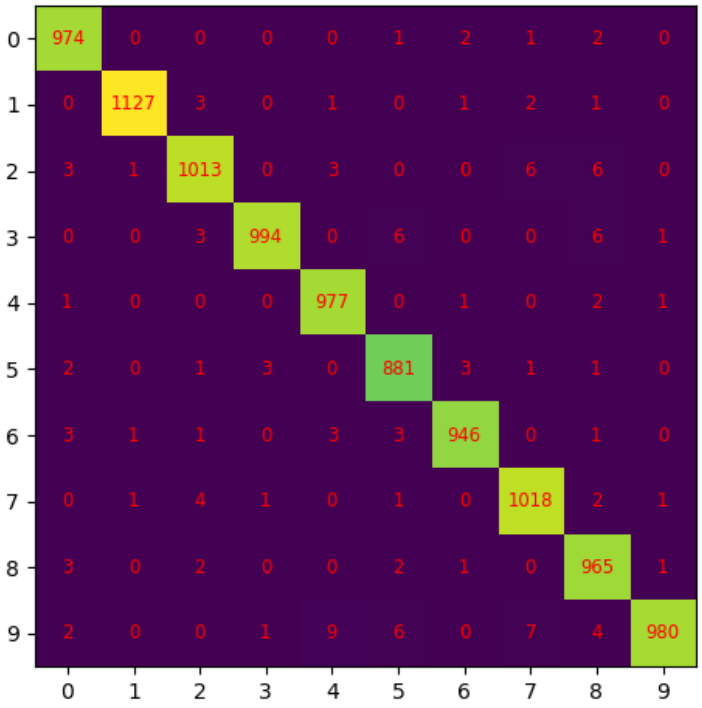
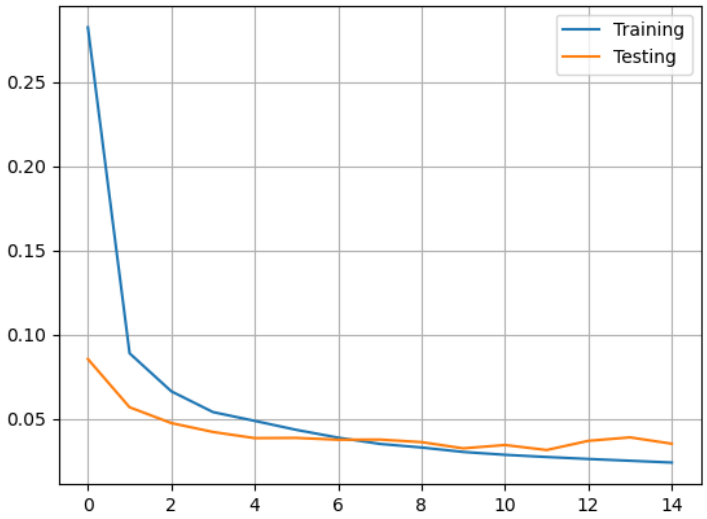


FIGURE 16 – Courbes de l'évolution des MSE et matrice de confusion pour le troisième modèle convolutif. Ce modèle contient une couche convolutive en moins que les deux modèles précédents et ne possède pas de couche dense entre l'aplanissement et la couche de sortie. La taille des fenêtres de convolution ont aussi été réduites à 3 par 3 pixels pour les deux couches. La première couche a 16 filtres et la deuxième 32.

```
Test score:    0.0237
Test accuracy: 0.9926
```



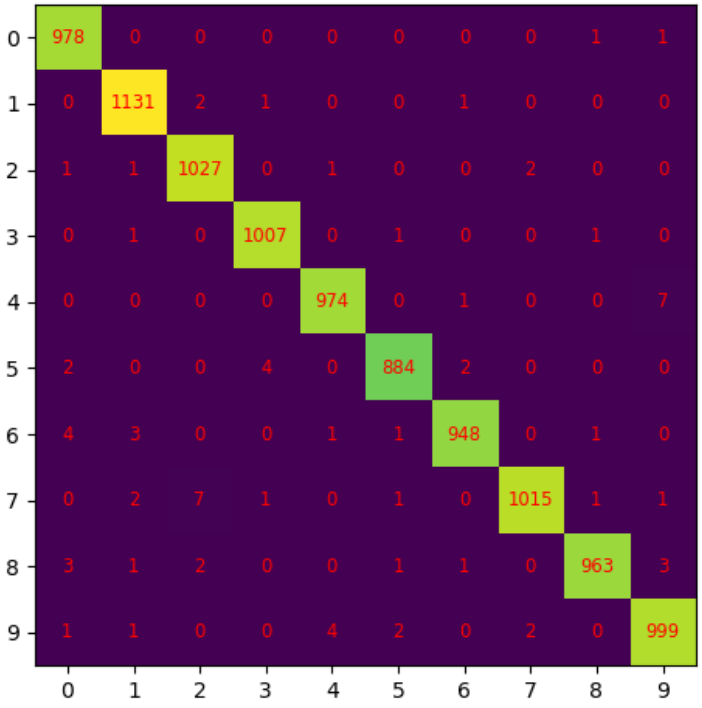


FIGURE 17 – Courbes de l'évolution des MSE et matrice de confusion pour le modèle convolutif final. Ce modèle a la même topologie que celui de la figure 16 à l'exception d'un ajout d'effets *dropout* après chaque couche.

5 Entraînement CNN pour Fashion MNIST, résultats et catégories confondues

> 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 7

Pour résoudre ce modèle, nous avons procédé de la même manière que pour les expériences précédentes, en partant du modèle final pour CNN obtenu précédemment. Comme le dataset Fashion MNIST est beaucoup plus difficile à classifier que le dataset MNIST, la couche dense entre l'aplanissement et la couche de sortie a été rajoutée avec un nombre initial de 128 neurones.

Le modèle obtenu avait encore quelques difficultés à distinguer certaines parties des images avec un nombre élevé de confusions, cela nous a donné l'idée d'augmenter le nombre de filtres des deux couches convolutives à 32 pour la première et 64 pour la seconde.

Cette seconde tentative a commencé à developper un overfitting important dans les dernières évolutions. C'est pourquoi nous avons changé les paramètres des effets de *dropout* en les augmentant légèrement.

Une fois les effets *dropouts* ajustés, le modèle n'avait plus autant d'overfitting. Cependant, nous avons remarqué que le modèle avait tendance à stagner à une erreur que nous jugions trop élevée, environ 0.22, et donc le modèle avait encore de la marge pour s'améliorer. Nous avons donc augmenté le nombre d'epochs à 25 et nous avons ajouté un *callback* nommé `ReduceLROnPlateau`. Comme son nom l'indique, ce *callback* est une fonction qui va réduire le learning rate d'un facteur de 0.1 quand il détecte que la fonction de coût donne un résultat qui stagne ou que la précision commence à tomber. Cela nous a permis de continuer à faire évoluer le modèle alors que la version sans *callback* aurait commencé à overfitter de plus en plus.

On peut observer l'effet du callback dans le graphe des MSE d'entrainement et de validation en figure 18, avec un changement de la tendance de la courbe important entre la $12^{\text{ème}}$ et la $13^{\text{ème}}$ epoch. Le modèle continue à s'améliorer ensuite même si l'on remarque moins d'évolution à partir de l'epoch n° 15.

Une idée pour une potentielle amélioration serait de réduire dynamiquement le nombre d'epochs – un *early-stopping* – une fois que la fonction de coût de validation commence à stagner, en priorisant d'abord les tentatives de réduction du learning rate.

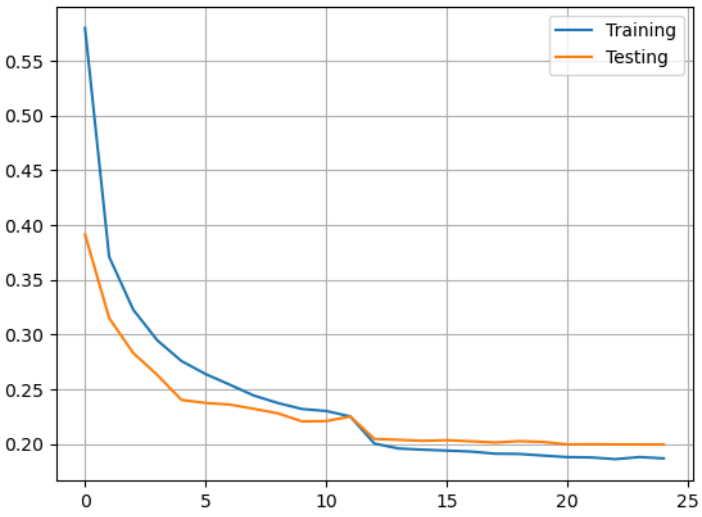
Nous émettons également l'hypothèse que le modèle aurait potentiellement besoin de plus d'échantillons pour l'entrainement, cela pourrait être géré avec des modifications aléatoires des images sources (rotations, zoom, inversions x/y, etc.) ce qui permettrait au modèle de mieux détecter plus de variantes des habits et, ainsi, améliorer la précision.

Les résultats sont, dans l'ensemble, corrects, avec une précision finale de 92.58%. Nous remarquons que les chemises (shirt) sont le plus souvent confondues avec d'autres classes du dataset. En outre, les vestes et les chaussures semblent aussi être des points de confusion communs.

Les f-score finaux, donnés dans le listing 2, confirment nos observations de la matrice de confusion du modèle.

En bref, nous sommes satisfaits de ce modèle qui classifie déjà bien les données en entrées avec quelques difficultés pour des classes d'habits qui sont quand même très semblables. La topologie nous semble suffisamment simple par rapport à la complexité du modèle en laissant quand même une certaine flexibilité pour une généralisation plus ample.

```
Test score:    0.2117
Test accuracy: 0.9258
```



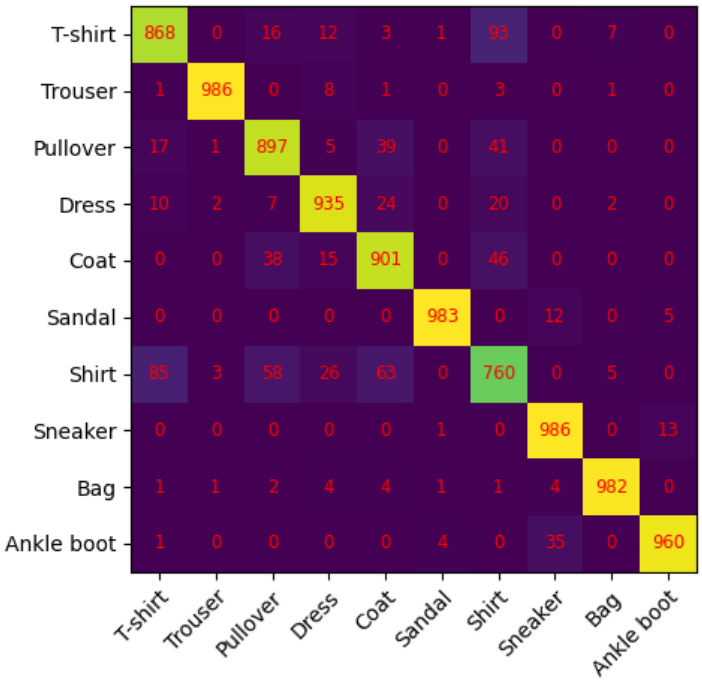


FIGURE 18 – Courbes de l'évolution des MSE et matrice de confusion pour le modèle convolutif de classification des données du dataset Fashion MNIST. La topologie de modèle est celle présentée dans la sous-section 2.4.

LISTING 2 – f-score obtenus par le modèle convolutif pour le dataset Fashion MNIST

```
F-score T-shirt:    0.8754
F-score Trouser:    0.9895
F-score Pullover:   0.8890
F-score Dress:      0.9327
F-score Coat:       0.8855
F-score Sandal:     0.9879
F-score Shirt:      0.7739
F-score Sneaker:    0.9681
F-score Bag:        0.9835
F-score Ankle boot: 0.9707
```