Deep Neural Networks
ARN - PW4

Table des matières

1. Etudiants

Ce laboratoire a été effectué par :

- Gwendal Piemontesi, ISCL
- Guillaume Trueb, ISCL

2. Configuration d'apprentissage

2.1. Algorithme d'optimisation

L'algorithme utilisé pour optimiser les poids du réseau de neurones est **RMSprop** (Root Mean Square Propagation).

Il s'agit d'un optimiseur adaptatif qui ajuste automatiquement le taux d'apprentissage pour chaque poids en fonction de la moyenne des gradients récents.

2.2. Hyperparamètres

Les paramètres utilisés par défaut dans l'implémentation Keras de RMSprop sont :

Paramètre	**Valeur**
learning_rate	0.001
momentum	0.0
rho	0.9
epsilon	1e-07

rho représente le taux de décroissance pour la moyenne des carrés des gradients, tandis qu'*epsilon* est un petit terme pour éviter la division par zéro.

Pour nos expériences, nous réaliserons à chaque fois **50 époques** afin d'évaluer la performance du modèle.

2.3. Fonction de perte

La fonction de perte utilisée est **categorical_crossentropy**, qui correspond à l'entropie croisée pour des sorties catégorielles (classification multi-classes avec one-hot encoding).

La formule est exprimée comme suit :

$$L(y, \hat{y}) = -\sum_{i=1}^{C} y_i \log(\hat{y}_i)$$

$L(y, \hat{y})$ est la **perte d'entropie croisée catégorielle**.

y_i est la **valeur réelle** (0 ou 1 pour chaque classe), provenant du vecteur de sortie **encodé one-hot**.

$\hat{y}_i$ est la **probabilité prédite** pour la classe i.

C est le **nombre de classes**.

3. Modèles CNN: Profonds vs Peu Profonds

En général, **les modèles CNN profonds n'ont pas nécessairement plus de poids que les modèles superficiels**. Le nombre total de paramètres (poids) dépend de plusieurs facteurs:

- Le nombre de couches (profondeur)
- Le nombre de filtres par couche
- La taille des filtres de convolution
- La présence de couches entièrement connectées

3.1. Exemple comparatif

Comparons deux architectures CNN hypothétiques:

3.1.1. Modèle CNN superficiel

Architecture:

- 1 couche de convolution (100 filtres de taille 5×5×3)
- 1 couche entièrement connectée (1000 neurones ⟶ 10 classes)

Nombre de paramètres:

- Couche de convolution: $100 \times (5 \times 5 \times 3 + 1) = 7\,600$
- Couche entièrement connectée: $100 \times 32 \times 32 \times 1000 + 1000 \times 10 + 1010 = 102\,410\,010$
- **Total: environ 102,4 millions de paramètres**

3.1.2. Modèle CNN profond

Architecture:

- 5 couches de convolution (16, 32, 64, 64, 128 filtres de taille 3×3)
- 2 couches entièrement connectées (256 ⟶ 64 ⟶ 10 neurones)

Nombre de paramètres:

- Conv1: $16 \times (3 \times 3 \times 3 + 1) = 448$
- Conv2: $32 \times (3 \times 3 \times 16 + 1) = 4\,640$
- Conv3: $64 \times (3 \times 3 \times 32 + 1) = 18\,496$
- Conv4: $64 \times (3 \times 3 \times 64 + 1) = 36\,928$
- Conv5: $128 \times (3 \times 3 \times 64 + 1) = 73\,856$
- FC1: $128 \times 4 \times 4 \times 256 + 256 = 524\,544$
- FC2: $256 \times 64 + 64 = 16\,448$
- FC3: $64 \times 10 + 10 = 650$
- **Total: environ 676 mille paramètres**

3.2. Conclusion

Dans cet exemple, nous observons que:

1. Le modèle CNN superficiel possède environ 102,4 millions de paramètres
2. Le modèle CNN profond possède seulement environ 676 mille paramètres

Ceci démontre clairement qu'un modèle plus profond peut avoir significativement moins de paramètres qu'un modèle superficiel. La raison principale est que:

- Le modèle superficiel utilise des filtres plus grands et plus nombreux dès le début
- Le modèle superficiel a une énorme couche entièrement connectée qui constitue la majorité de ses paramètres
- Le modèle profond utilise des filtres de convolution de plus petite taille (3×3)
- Le modèle profond réduit progressivement la dimension spatiale, ce qui diminue le nombre de connexions dans les couches entièrement connectées

Cette efficacité en termes de paramètres est l'un des avantages des architectures profondes, qui peuvent capturer des caractéristiques hiérarchiques complexes avec moins de poids au total.

4. Multilayer Perceptron à partir de données brutes

Nous avons testé trois architectures de réseaux de neurones avec différents niveaux de complexité. Pour chaque expérience, nous détaillons la topologie du réseau, le nombre de paramètres et les performances obtenues.

4.1. Exemple de topologie de réseau pour la définition des caractéristiques d'entrée/sortie et le calcul du nombre de poids

4.1.1. Caractéristiques d'entrée/sortie

Entrées: Pixels bruts de l'image

- Pour une image 32×32 en niveaux de gris: 32 × 32 = 1 024 entrées
- Pour une image 32×32 en couleur (RGB): 32 × 32 × 3 = 3 072 entrées

Architecture:

- Couche d'entrée: 3 072 neurones (pour une image couleur 32×32)
- Couche cachée: 512 neurones avec activation ReLU
- Couche de sortie: 10 neurones (pour classifier 10 classes) avec softmax

Sorties: 10 valeurs représentant les probabilités d'appartenance à chaque classe

4.1.2. Calcul du nombre de poids

Entre la couche d'entrée et la couche cachée:

- Poids: 3 072 entrées × 512 neurones = 1 572 864 poids
- Biais: 512 (un par neurone de la couche cachée)
- Sous-total: 1 572 864 + 512 = 1 573 376 paramètres

Entre la couche cachée et la couche de sortie:

- Poids: 512 neurones × 10 sorties = 5 120 poids
- Biais: 10 (un par neurone de sortie)
- Sous-total: 5 120 + 10 = 5 130 paramètres

Nombre total de paramètres: 1 573 376 + 5 130 = 1 578 506 paramètres

Explication: Chaque connexion entre deux neurones nécessite un poids. De plus, chaque neurone d'une couche (sauf la couche d'entrée) possède un biais. Le nombre total de poids entre deux couches est donc le produit du nombre de neurones dans chaque couche, et le nombre total de biais est égal au nombre de neurones de la deuxième couche.

4.2. Modèle 1

Pour le premier modèle, nous avons choisi une architecture minimale avec seulement deux couches denses et un nombre d'époques fixé à 50 pour permettre un entraînement plus long.

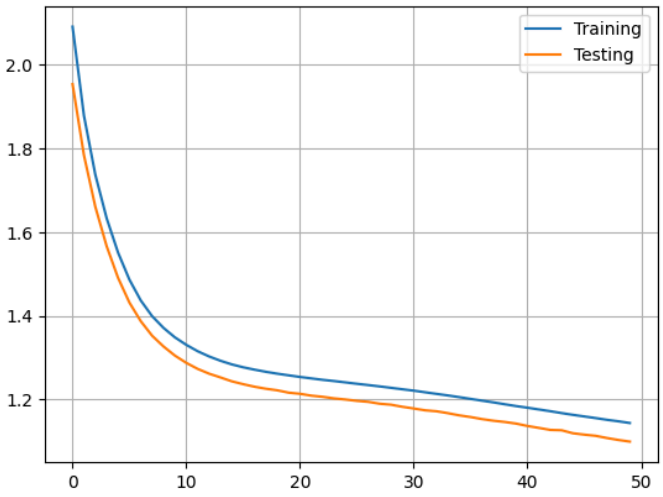
Architecture du modèle

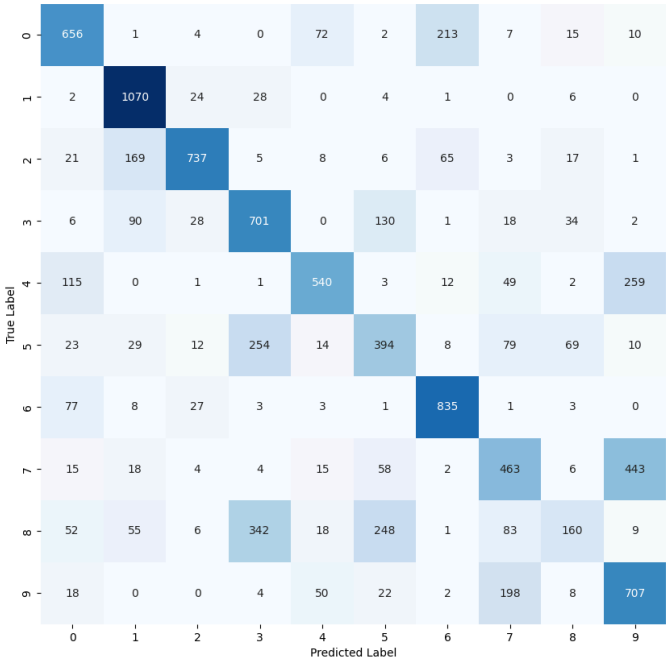
Layer (type)	Output Shape	Param #
dense_19 (Dense)	(None, 2)	1,570
dense_20 (Dense)	(None, 10)	30

```
Total params: 1,600 (6.25 KB)
Trainable params: 1,600 (6.25 KB)
Non-trainable params: 0 (0.00 B)
```

Performances du modèle

Métrique	Valeur
Test score	1.1458
Test accuracy	0.6263
F1 macro Score	0.6064
F1 weighted Score	0.6113
F1 micro Score	0.6263





Ce modèle est d'une efficacité relativement médiocre, avec une précision de test d'environ 62.6%. Sa simplicité (seulement 2 neurones dans la couche cachée) limite fortement sa capacité d'apprentissage.

4.3. Modèle 2

Ce deuxième modèle améliore l'architecture précédente en augmentant le nombre de neurones cachés et en introduisant une régularisation par dropout pour limiter le surapprentissage.

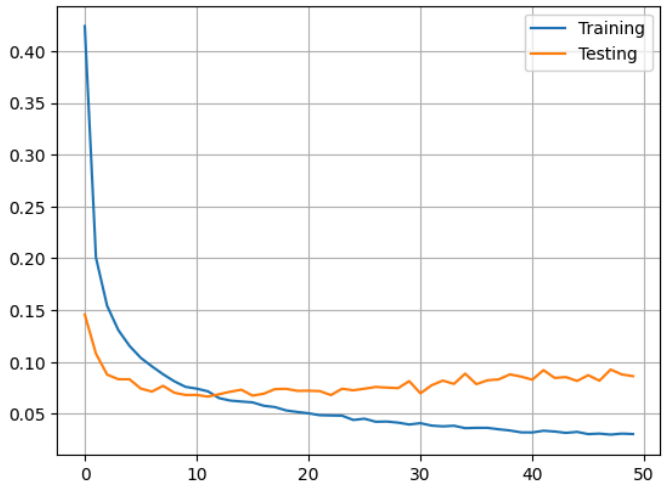
Architecture du modèle

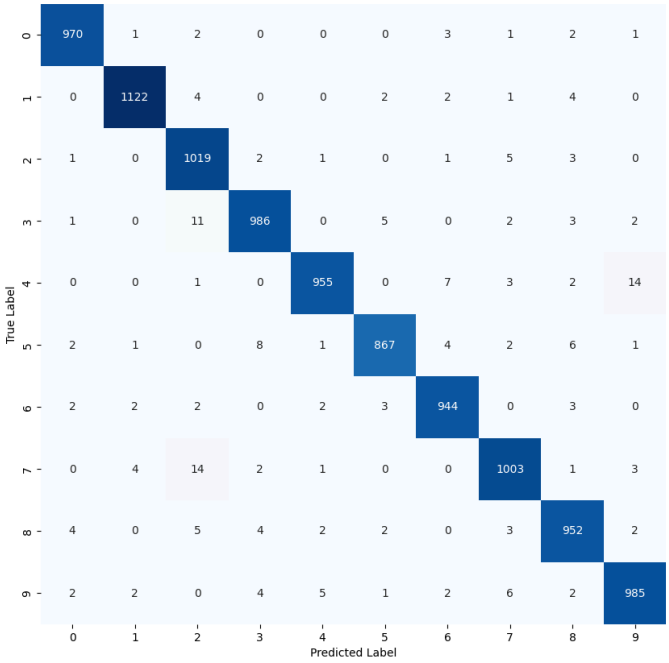
Layer (type)	Output Shape	Param #
dense_27 (Dense)	(None, 128)	100,480
dropout_11 (Dropout)	(None, 128)	0
dense_28 (Dense)	(None, 64)	8,256
dense_29 (Dense)	(None, 10)	650

```
Total params: 109,386 (427.29 KB)
Trainable params: 109,386 (427.29 KB)
Non-trainable params: 0 (0.00 B)
```

Performances du modèle

Métrique	Valeur
Test score	0.0951
Test accuracy	0.9803
F1 macro Score	0.9802
F1 weighted Score	0.9803
F1 micro Score	0.9803





Ce modèle offre un bon compromis entre complexité et performance avec une précision de test d'environ 98%. L'activation ReLU et la couche de dropout ont permis un apprentissage plus efficace et une meilleure généralisation.

4.3.1. Analyse des confusions fréquentes

Avec ce modèle plus performant, les confusions sont beaucoup moins fréquentes, mais certains motifs persistent:

- Le chiffre 4 est parfois confondu avec le 9
- Le chiffre 3 est parfois confondu avec le 2
- Le chiffre 7 est parfois confondu avce le 2

Ces confusions concernent principalement des chiffres qui partagent des caractéristiques visuelles similaires. L'augmentation significative du nombre de neurones dans la couche cachée (de 2 à 128) a permis au réseau de capturer des caractéristiques beaucoup plus complexes et distinctives.

4.4. Modèle 3

Ce troisième modèle adopte une architecture plus profonde avec deux couches cachées plus larges et introduit à la fois une régularisation avancée (dropout) et une normalisation (BatchNormalization).

Architecture du modèle

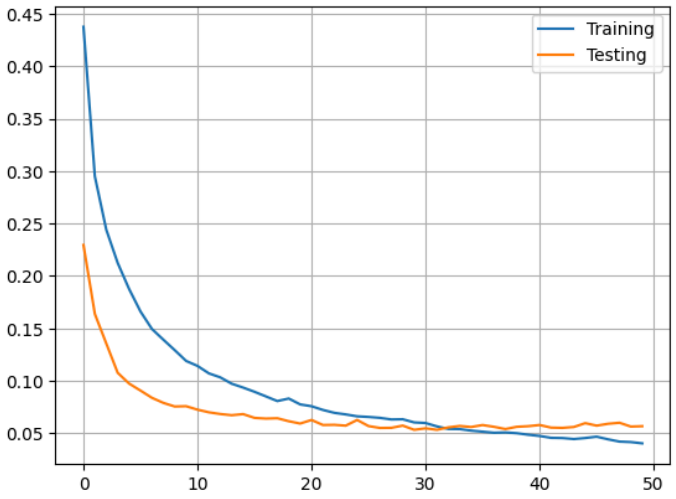
```
┏━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━┓
┃ Layer (type)                    ┃ Output Shape           ┃       Param # ┃
```

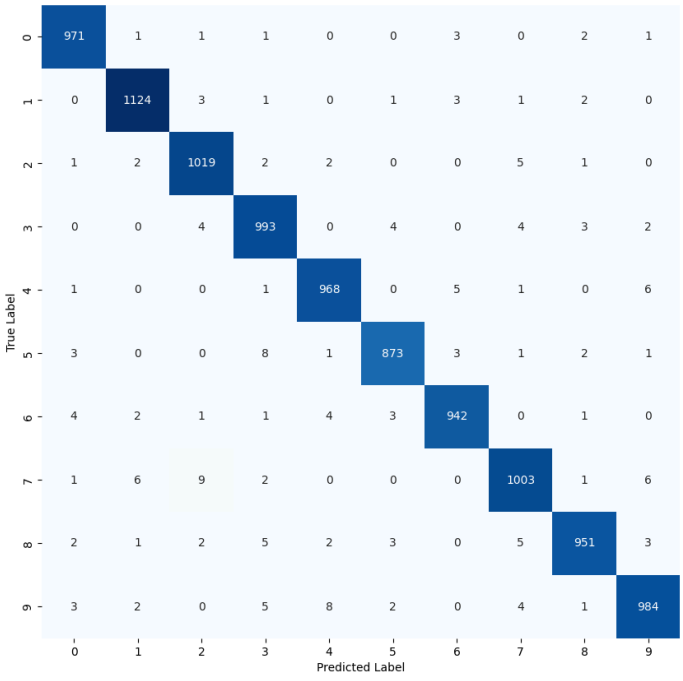
dense_34 (Dense)	(None, 256)	200,960
batch_normalization (BatchNormalization)	(None, 256)	1,024
dropout_13 (Dropout)	(None, 256)	0
dense_35 (Dense)	(None, 128)	32,896
batch_normalization_1 (BatchNormalization)	(None, 128)	512
dropout_14 (Dropout)	(None, 128)	0
dense_36 (Dense)	(None, 10)	1,290

```
Total params: 236,682 (924.54 KB)
Trainable params: 235,914 (921.54 KB)
Non-trainable params: 768 (3.00 KB)
```

Performances du modèle

Métrique	Valeur
Test score	0.0591
Test accuracy	0.9854
F1 macro Score	0.9853
F1 weighted Score	0.9854
F1 micro Score	0.9854





Ce modèle présente les meilleures performances avec une précision de test d'environ 98.5%. Les techniques de BatchNormalization et Dropout ont amélioré la stabilité de l'entraînement et réduit le surapprentissage.

4.4.1. Analyse des confusions fréquentes

Même avec cette architecture avancée, certaines confusions persistent, mais à une fréquence très faible:

- Le chiffre 9 est parfois confondu avec le 4
- Le chiffre 7 est parfois confondu avec le 2
- Le chiffre 5 est parfois confondu avec le 3

Ces erreurs résiduelles représentent des cas particulièrement difficiles où l'écriture manuscrite présente des ambiguïtés intrinsèques. La normalisation par lots et les couches plus larges ont permis de réduire encore davantage ces confusions par rapport au modèle précédent.

4.5. Conclusion

Cette étude a démontré l'importance de l'architecture du réseau dans les performances de classification. Le passage d'une architecture simple (Modèle 1) à des architectures plus complexes (Modèles 2 et 3) a permis d'améliorer significativement les performances:

Modèle	Précision	Paramètres
Modèle 1	62.63%	1,600
Modèle 2	98.03%	109,386
Modèle 3	98.54%	236,682

Le Modèle 3, avec sa combinaison de couches plus larges, normalisation par lots et régularisation par dropout, offre le meilleur équilibre entre complexité et performance pour cette tâche de classification.

5. Multilayer Perceptron avec caractéristiques HOG

Nous avons testé trois architectures de réseaux de neurones utilisant des caractéristiques HOG (Histogrammes de Gradients Orientés) comme entrée. Pour chaque expérience, nous détaillons la topologie du réseau, le nombre de paramètres et les performances obtenues.

5.1. Exemple de topologie de réseau pour la définition des caractéristiques d'entrée/sortie et le calcul du nombre de poids

5.1.1. Caractéristiques d'entrée/sortie

Entrées: Descripteurs HOG (Histogrammes de Gradients Orientés)

- Les descripteurs HOG extraient des caractéristiques de l'image
- Pour une configuration standard avec 8×8 cellules, 2×2 blocs et 9 orientations:
- Nombre d'entrées: environ 1 764 caractéristiques (dépend de la configuration exacte)

Architecture:

- Couche d'entrée: 1 764 neurones
- Couche cachée: 256 neurones avec activation ReLU
- Couche de sortie: 10 neurones avec softmax

Sorties: 10 valeurs représentant les probabilités d'appartenance à chaque classe

5.1.2. Calcul du nombre de poids

Entre la couche d'entrée et la couche cachée:

- Poids: 1 764 entrées × 256 neurones = 451 584 poids
- Biais: 256 (un par neurone de la couche cachée)
- Sous-total: 451 584 + 256 = 451 840 paramètres

Entre la couche cachée et la couche de sortie:

- Poids: 256 neurones × 10 sorties = 2 560 poids
- Biais: 10 (un par neurone de sortie)
- Sous-total: 2 560 + 10 = 2 570 paramètres

Nombre total de paramètres: 451 840 + 2 570 = 454 410 paramètres

Explication: L'utilisation des caractéristiques HOG réduit considérablement la dimension des entrées par rapport aux pixels bruts, ce qui diminue le nombre de paramètres nécessaires. Cela permet d'avoir un modèle plus compact qui peut être plus facile à entraîner et moins sujet au surapprentissage.

5.2. Modèle 1

Pour le premier modèle, nous avons choisi une architecture minimale avec seulement deux couches denses et un nombre très limité de neurones cachés (2).

Architecture du modèle

```
Input  layer : 392
Output layer : 10

orientations : 8
pix per cell : 4
```

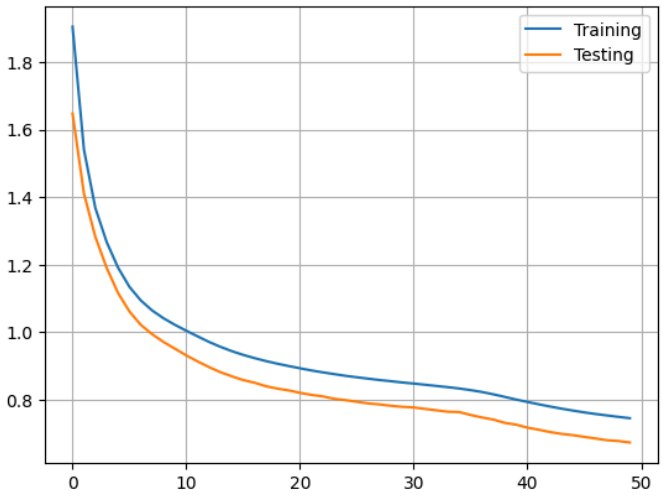
Layer (type)	Output Shape	Param #
dense (Dense)	(None, 2)	786
dense_1 (Dense)	(None, 10)	30

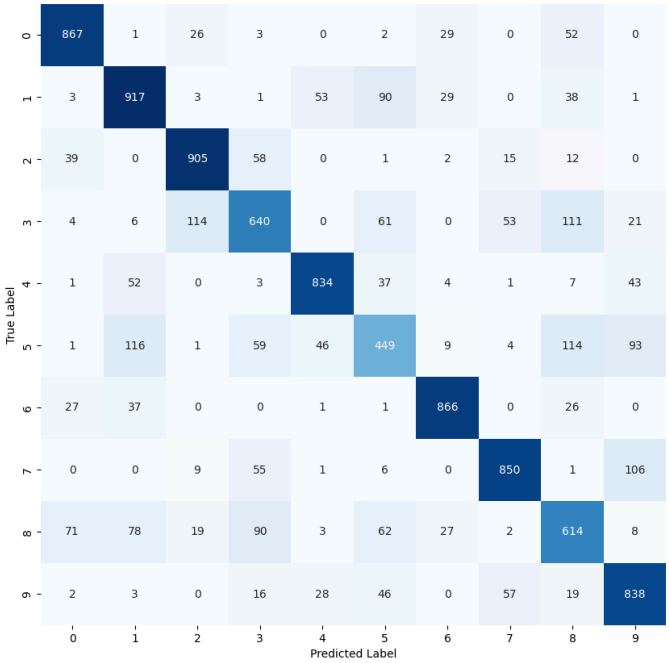
```
Total params: 816 (3.19 KB)
```

```
Trainable params: 816 (3.19 KB)
Non-trainable params: 0 (0.00 B)
```

Performances du modèle

Métrique	Valeur
Test score	0.7175
Test accuracy	0.7780
F1 macro Score	0.7733
F1 weighted Score	0.7758
F1 micro Score	0.7780





Ce modèle montre une efficacité modérée avec une précision de test d'environ 77.8%. Malgré sa simplicité (seulement 2 neurones dans la couche cachée), il parvient à atteindre des performances nettement supérieures au modèle équivalent basé sur les données brutes (62.63%). Cela démontre l'avantage d'utiliser des caractéristiques HOG prétraitées par rapport aux pixels bruts.

5.3. Modèle 2

Ce deuxième modèle améliore l'architecture précédente en augmentant le nombre de neurones cachés et en introduisant une régularisation par dropout. Nous avons également ajusté les paramètres HOG avec un `pix per cell` plus grand (7 au lieu de 4), réduisant ainsi la dimensionnalité des entrées de 392 à 128.

Architecture du modèle

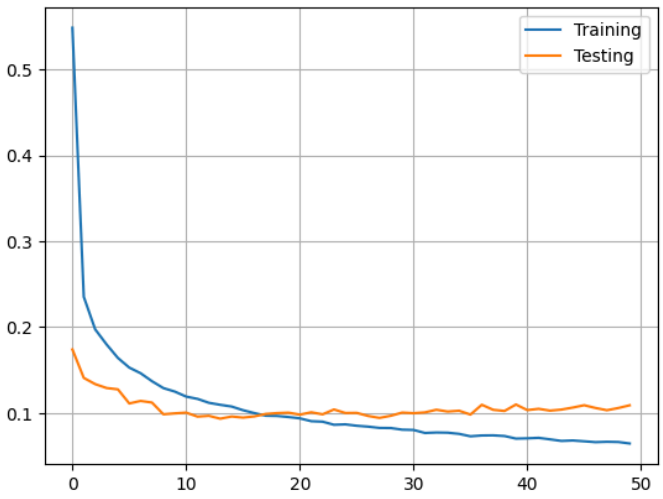
```
Input  layer : 128
Output layer : 10

orientations : 8
pix per cell : 7
```

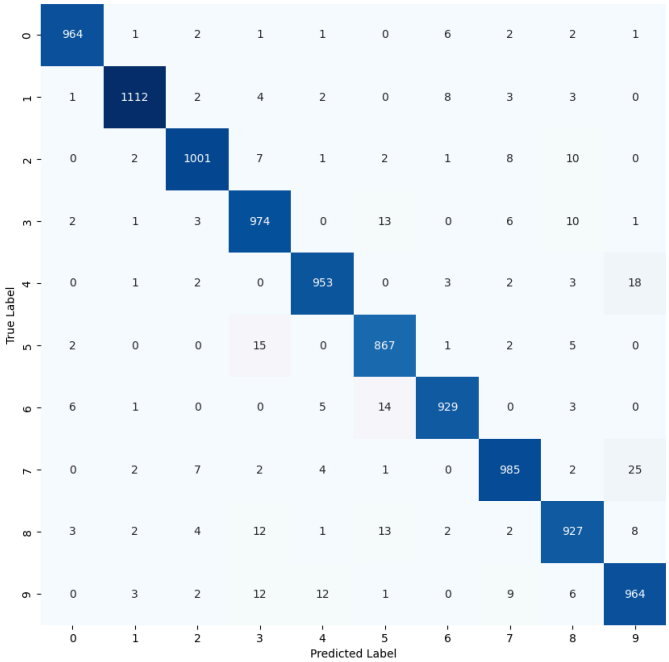
Layer (type)	Output Shape	Param #
dense (Dense)	(None, 128)	16,512
dropout (Dropout)	(None, 128)	0
dense_1 (Dense)	(None, 64)	8,256
dense_2 (Dense)	(None, 10)	650

```
Total params: 25,418 (99.29 KB)
Trainable params: 25,418 (99.29 KB)
Non-trainable params: 0 (0.00 B)
```

Performances du modèle

Métrique	**Valeur**
Test score	0.1151
Test accuracy	0.9676
F1 macro Score	0.9674
F1 weighted Score	0.9676
F1 micro Score	0.9676





Ce modèle offre une amélioration significative des performances avec une précision de test d'environ 96.8%. L'augmentation du nombre de neurones et l'ajout d'une couche supplémentaire ont considérablement amélioré la capacité du réseau à saisir des caractéristiques plus complexes. Le dropout a probablement contribué à une meilleure généralisation en évitant le surapprentissage.

5.3.1. Analyse des confusions fréquentes

Avec ce modèle plus performant, les confusions sont beaucoup moins fréquentes, mais certains motifs persistent :

- Le chiffre 9 est parfois confondu avec le 4 et le 7

- Le chiffre 5 est parfois confondu avec le 3 et le 6

Ces confusions concernent principalement des chiffres qui partagent des caractéristiques de gradient similaires dans la représentation HOG.

5.4. Modèle 3

Ce troisième modèle adopte une architecture plus profonde avec des couches cachées plus larges et introduit à la fois une régularisation par dropout et une normalisation par lots (BatchNormalization). Nous avons également augmenté le nombre d'orientations du descripteur HOG à 16 (au lieu de 8), doublant ainsi la dimensionnalité de l'entrée à 256.

Architecture du modèle

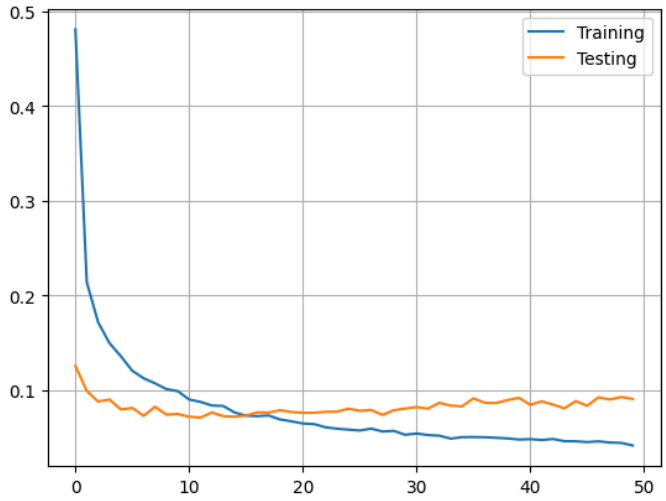
```
Input  layer : 256
Output layer : 10

orientations : 16
pix per cell : 7
```

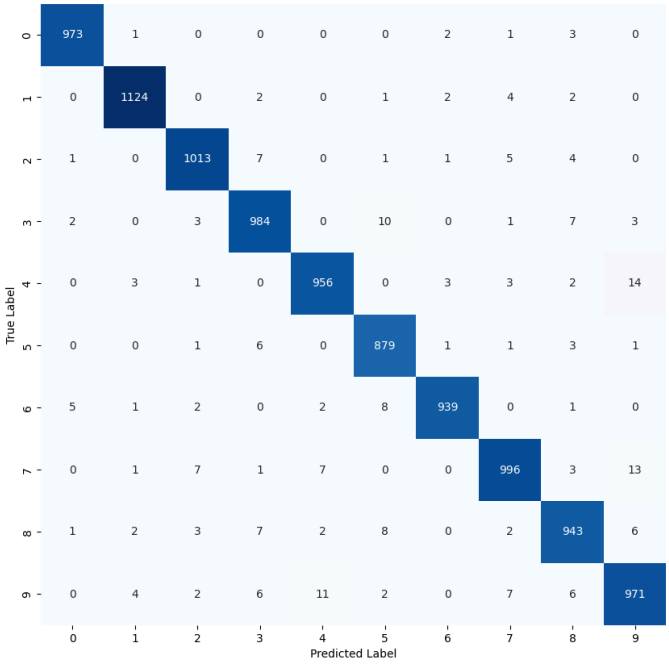
Layer (type)	Output Shape	Param #
dense_4 (Dense)	(None, 256)	65,792
dropout_2 (Dropout)	(None, 256)	0
batch_normalization_2 (BatchNormalization)	(None, 256)	1,024
dense_5 (Dense)	(None, 128)	32,896
dropout_3 (Dropout)	(None, 128)	0
batch_normalization_3 (BatchNormalization)	(None, 128)	512
dense_6 (Dense)	(None, 64)	8,256
dense_7 (Dense)	(None, 10)	650

```
Total params: 109,130 (426.29 KB)
Trainable params: 108,362 (423.29 KB)
Non-trainable params: 768 (3.00 KB)
```

Performances du modèle

Métrique	**Valeur**
Test score	0.0920
Test accuracy	0.9778
F1 macro Score	0.9777
F1 weighted Score	0.9778
F1 micro Score	0.9778





Ce modèle présente les meilleures performances parmi les trois testés avec une précision de test d'environ 97.8%. L'utilisation de caractéristiques HOG plus détaillées (16 orientations), combinée à une architecture plus profonde et à des techniques de régularisation avancées, a permis d'améliorer encore les résultats par rapport au modèle précédent.

5.4.1. Analyse des confusions fréquentes

Même avec cette architecture avancée, certaines confusions persistent, mais à une fréquence très faible :

- Le chiffre 9 est parfois confondu avec le 4
- Le chiffre 9 est parfois confondu avec le 7
- Le chiffre 3 est parfois confondu avec le 5

Ces erreurs résiduelles représentent des cas particulièrement difficiles où l'écriture manuscrite présente des ambiguïtés intrinsèques, même dans l'espace des caractéristiques HOG.

5.5. Conclusion

Cette étude a démontré l'efficacité de l'utilisation des caractéristiques HOG comme prétraitement pour la classification de chiffres manuscrits. Le passage d'une architecture simple (Modèle 1) à des architectures plus complexes (Modèles 2 et 3) a permis d'améliorer significativement les performances :

Modèle	Précision	Paramètres	Configuration HOG
Modèle 1	77.80%	816	8 orientations, 4 pixels par cellule
Modèle 2	96.76%	25,418	8 orientations, 7 pixels par cellule
Modèle 3	97.78%	109,130	16 orientations, 7 pixels par cellule

Comparativement aux modèles utilisant directement les pixels bruts (sans extraction HOG), nous observons que :

1. L'utilisation de caractéristiques HOG permet d'atteindre des performances similaires avec beaucoup moins de paramètres
2. Le Modèle 1 avec HOG (77.80%) est significativement meilleur que son équivalent sur données brutes (62.63%)
3. Les Modèles 2 et 3 avec HOG atteignent des performances proches de celles obtenues avec les données brutes, mais avec une réduction substantielle du nombre de paramètres

Le Modèle 3 avec HOG, combinant un descripteur enrichi (16 orientations), des couches cachées larges, une normalisation par lots et une régularisation par dropout, offre le meilleur équilibre entre complexité et performance pour cette tâche de classification.

Cette approche démontre l'intérêt de l'extraction de caractéristiques pertinentes (comme HOG) avant l'apprentissage, permettant de construire des modèles plus compacts et efficaces pour la reconnaissance de formes.

6. CNN

Nous avons testé trois architectures de réseaux de neurones convolutifs avec différents niveaux de complexité. Pour chaque expérience, nous détaillons la topologie du réseau, le nombre de paramètres et les performances obtenues.

6.1. Exemple de topologie de réseau pour la définition des caractéristiques d'entrée/sortie et le calcul du nombre de poids

6.1.1. Caractéristiques d'entrée/sortie

Entrées: Pixels bruts de l'image

- Image 32×32×3 (RGB)

Architecture:

- Couche de convolution 1: 32 filtres de taille 3×3
- Couche de pooling 1: Max pooling 2×2
- Couche de convolution 2: 64 filtres de taille 3×3
- Couche de pooling 2: Max pooling 2×2
- Couche de convolution 3: 128 filtres de taille 3×3
- Couche entièrement connectée 1: 256 neurones
- Couche entièrement connectée 2: 10 neurones (sortie)

Sorties: 10 valeurs représentant les probabilités d'appartenance à chaque classe

6.1.2. Calcul du nombre de poids

Couche de convolution 1:

- Filtres: 32 filtres de taille 3×3×3 (RGB)
- Poids: 32 × (3 × 3 × 3) = 864 poids
- Biais: 32 (un par filtre)
- Sous-total: 864 + 32 = 896 paramètres

Couche de convolution 2:

- Filtres: 64 filtres de taille 3×3×32 (profondeur de la couche précédente)
- Poids: 64 × (3 × 3 × 32) = 18 432 poids
- Biais: 64 (un par filtre)
- Sous-total: 18 432 + 64 = 18 496 paramètres

Couche de convolution 3:

- Filtres: 128 filtres de taille 3×3×64 (profondeur de la couche précédente)
- Poids: 128 × (3 × 3 × 64) = 73 728 poids
- Biais: 128 (un par filtre)
- Sous-total: 73 728 + 128 = 73 856 paramètres

Couche entièrement connectée 1:

- Après le max pooling, la taille de la feature map est réduite à 4×4×128
- Entrée aplatie: 4 × 4 × 128 = 2 048 valeurs
- Poids: 2 048 × 256 = 524 288 poids
- Biais: 256 (un par neurone)
- Sous-total: 524 288 + 256 = 524 544 paramètres

Couche entièrement connectée 2 (sortie):

- Poids: 256 × 10 = 2 560 poids
- Biais: 10 (un par neurone de sortie)
- Sous-total: 2 560 + 10 = 2 570 paramètres

Nombre total de paramètres: 896 + 18 496 + 73 856 + 524 544 + 2 570 = 620 362 paramètres

Explication:

1. Dans les couches de convolution, chaque filtre a des poids partagés qui sont appliqués sur toute l'image. Le nombre de paramètres dépend de la taille du filtre et du nombre de canaux d'entrée.
2. Les couches de pooling n'ont pas de paramètres à apprendre.
3. La réduction de dimensionnalité par les opérations de pooling permet de diminuer considérablement le nombre de connexions dans les couches entièrement connectées.
4. Malgré sa profondeur, le CNN a beaucoup moins de paramètres que le MLP avec données brutes grâce au partage de poids dans les couches de convolution.

6.2. Modèle 1

Pour le premier modèle, nous avons choisi une architecture simple avec trois couches convolutives suivies de couches de pooling, et deux couches denses finales.

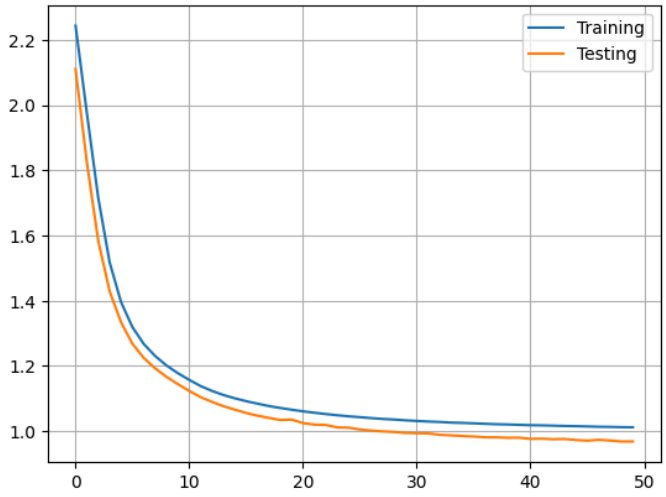
Architecture du modèle

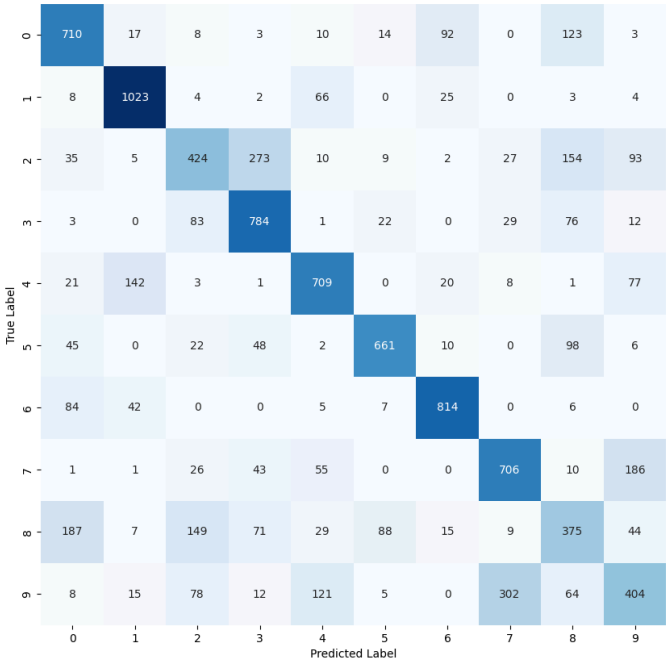
Layer (type)	Output Shape	Param #
l0 (InputLayer)	(None, 28, 28, 1)	0
l1 (Conv2D)	(None, 28, 28, 2)	10
l1_mp (MaxPooling2D)	(None, 14, 14, 2)	0
l2 (Conv2D)	(None, 14, 14, 2)	18
l2_mp (MaxPooling2D)	(None, 7, 7, 2)	0
l3 (Conv2D)	(None, 7, 7, 2)	18
l3_mp (MaxPooling2D)	(None, 3, 3, 2)	0
flat (Flatten)	(None, 18)	0
l4 (Dense)	(None, 2)	38
l5 (Dense)	(None, 10)	30

```
Total params: 114 (456.00 B)
Trainable params: 114 (456.00 B)
Non-trainable params: 0 (0.00 B)
```

Performances du modèle

Métrique	**Valeur**
Test score	0.9627
Test accuracy	0.6610
F1 macro Score	0.6540
F1 weighted Score	0.6544
F1 micro Score	0.6610





Ce modèle est d'une efficacité relativement médiocre, avec une précision de test d'environ 66.1%. Sa simplicité (seulement 2 filtres par couche convolutive) limite fortement sa capacité d'apprentissage des caractéristiques complexes des images. Le nombre extrêmement réduit de paramètres (114 au total) ne permet pas de capturer efficacement les motifs spatiaux nécessaires à la reconnaissance des chiffres.

6.3. Modèle 2

Ce deuxième modèle simplifie l'architecture précédente en réduisant le nombre de couches convolutives à une seule, mais augmente légèrement la complexité de la partie dense du réseau.

Architecture du modèle

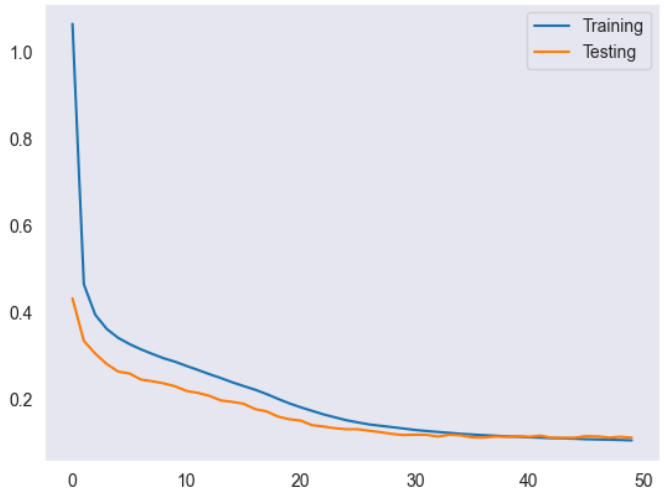
Layer (type)	Output Shape	Param #
l0 (InputLayer)	(None, 28, 28, 1)	0
l1 (Conv2D)	(None, 28, 28, 2)	20
l1_mp (MaxPooling2D)	(None, 14, 14, 2)	0
flat (Flatten)	(None, 392)	0
l4 (Dense)	(None, 8)	3,144
l5 (Dense)	(None, 10)	90

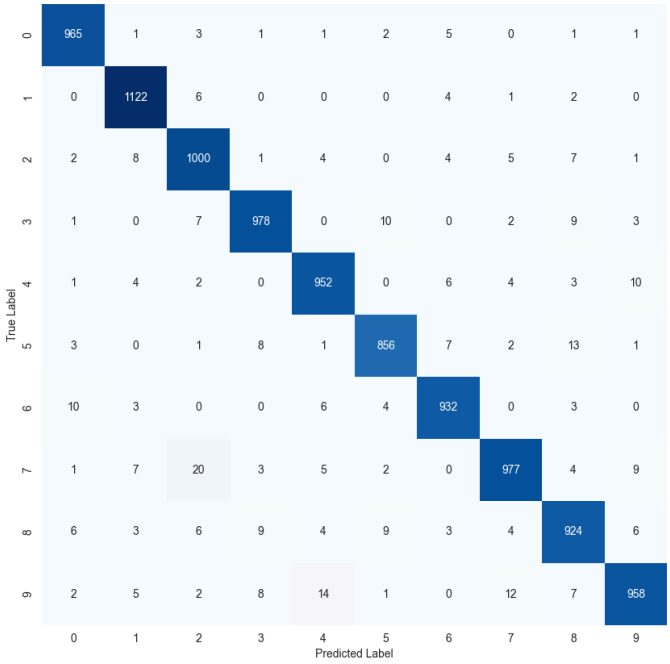
```
Total params: 3,254 (12.71 KB)
Trainable params: 3,254 (12.71 KB)
Non-trainable params: 0 (0.00 B)
```

Performances du modèle

Métrique	**Valeur**
Test score	0.1125

Test accuracy	0.9664
F1 macro Score	0.9662
F1 weighted Score	0.9664
F1 micro Score	0.9664





Ce modèle offre un excellent compromis entre simplicité et performance avec une précision de test d'environ 96.6%. Malgré l'utilisation d'une seule couche convolutive, l'augmentation significative du nombre de neurones dans la couche dense (8 au lieu de 2) permet un apprentissage beaucoup plus efficace des caractéristiques extraites par la couche convolutive.

6.3.1. Analyse des confusions fréquentes

Avec ce modèle plus performant, les confusions sont beaucoup moins fréquentes, mais certains motifs persistent:

- Le chiffre 9 est parfois confondu avec le 4
- Le chiffre 7 est parfois confondu avec le 2
- Le chiffre 8 est parfois confondu avec le 5

Ces confusions concernent principalement des chiffres qui partagent des caractéristiques visuelles similaires. L'architecture simplifiée avec seulement une couche convolutive mais une couche dense plus large semble suffisante pour capturer la plupart des caractéristiques distinctives des chiffres manuscrits.

6.4. Modèle 3

Ce troisième modèle adopte une architecture plus élaborée avec deux couches convolutives possédant davantage de filtres (4 puis 8), suivies de deux couches denses plus larges.

Architecture du modèle

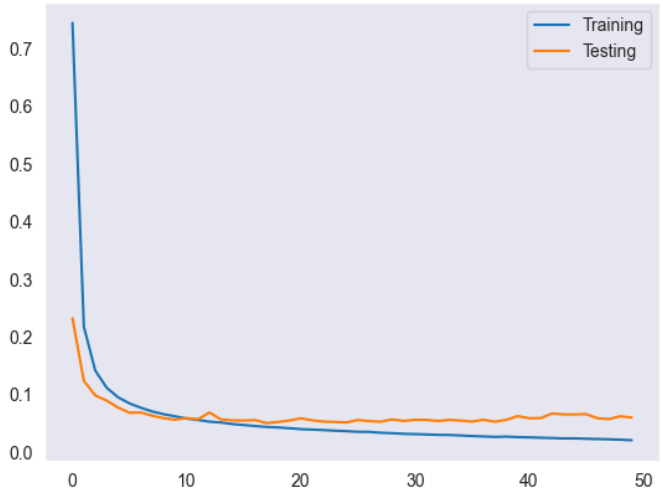
Layer (type)	Output Shape	Param #
l0 (InputLayer)	(None, 28, 28, 1)	0
l1 (Conv2D)	(None, 28, 28, 4)	40

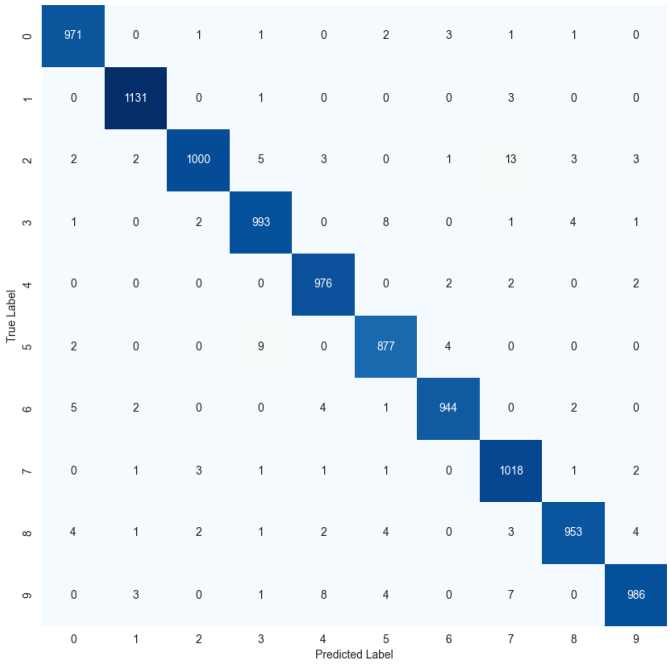
l1_mp (MaxPooling2D)	(None, 14, 14, 4)	0
l2 (Conv2D)	(None, 14, 14, 8)	296
l2_mp (MaxPooling2D)	(None, 7, 7, 8)	0
flat (Flatten)	(None, 392)	0
l4 (Dense)	(None, 16)	6,288
l5 (Dense)	(None, 10)	170

```
Total params: 6,794 (26.54 KB)
Trainable params: 6,794 (26.54 KB)
Non-trainable params: 0 (0.00 B)
```

Performances du modèle

Métrique	Valeur
Test score	0.053976
Test accuracy	0.9849
F1 macro Score	0.9848
F1 weighted Score	0.9849
F1 micro Score	0.9849





Ce modèle présente les meilleures performances avec une précision de test d'environ 98.5%. L'augmentation du nombre de filtres dans les couches convolutives (4 puis 8 au lieu de 2) ainsi que l'élargissement de la couche dense intermédiaire (16 neurones) ont permis d'améliorer significativement la capacité du réseau à reconnaître les caractéristiques distinctives des chiffres manuscrits.

6.4.1. Analyse des confusions fréquentes

Même avec cette architecture plus avancée, certaines confusions persistent, mais à une fréquence très faible:

- Le chiffre 7 est parfois confondu avec le 2
- Le chiffre 5 est parfois confondu avec le 3

Ces erreurs résiduelles représentent des cas particulièrement difficiles où l'écriture manuscrite présente des ambiguïtés intrinsèques. L'augmentation du nombre de filtres et de la complexité du réseau a permis de réduire considérablement ces confusions par rapport aux modèles précédents.

6.5. Conclusion

Cette étude a démontré l'efficacité des réseaux de neurones convolutifs pour la classification d'images, même avec un nombre très limité de paramètres. Le passage d'une architecture minimaliste (Modèle 1) à des architectures plus élaborées (Modèles 2 et 3) a permis d'améliorer significativement les performances:

Modèle	**Précision**	**Paramètres**
Modèle 1	66.10%	114
Modèle 2	96.64%	3,254
Modèle 3	98.49%	6,794

Le Modèle 3, avec son nombre plus élevé de filtres convolutifs et ses couches denses plus larges, offre le meilleur équilibre entre complexité et performance. Il est particulièrement remarquable que les CNNs parviennent à atteindre d'excellents résultats avec un nombre de paramètres nettement inférieur à celui des perceptrons multicouches traditionnels, démontrant ainsi leur efficacité pour les tâches de vision par ordinateur.

Un point important à noter est que le Modèle 2, malgré sa simplicité (une seule couche convolutive), offre déjà d'excellentes performances avec seulement 3,254 paramètres. Cela souligne l'importance fondamentale des opérations de convolution pour capturer les caractéristiques spatiales des images, même avec une architecture minimaliste.

7. CNN pour détection de pneumonie

Cette partie du travail pratique se concentre sur l'application des réseaux de neurones convolutifs à un problème de classification médicale. Le jeu de données utilisé contient des radiographies thoraciques représentant des poumons normaux et des poumons atteints de pneumonie. Notre objectif est de développer et d'entraîner un réseau de neurones convolutif capable de classifier automatiquement ces images en deux catégories distinctes.

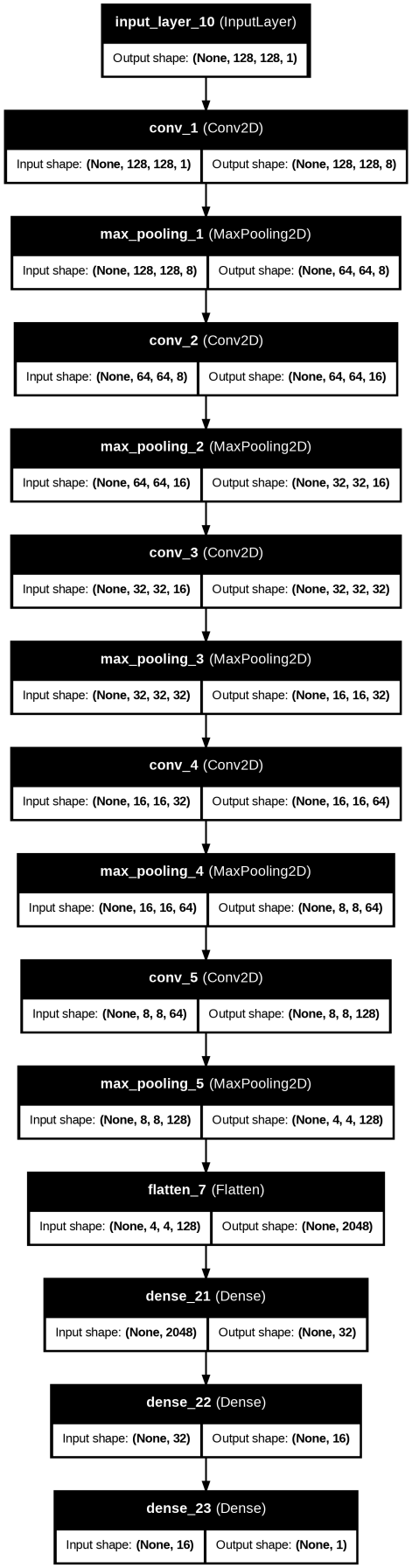
7.1. Architecture du modèle

Layer (type)	Output Shape	Param #
input_layer_10 (InputLayer)	(None, 128, 128, 1)	0
conv_1 (Conv2D)	(None, 128, 128, 8)	80
max_pooling_1 (MaxPooling2D)	(None, 64, 64, 8)	0
conv_2 (Conv2D)	(None, 64, 64, 16)	1,168
max_pooling_2 (MaxPooling2D)	(None, 32, 32, 16)	0
conv_3 (Conv2D)	(None, 32, 32, 32)	4,640
max_pooling_3 (MaxPooling2D)	(None, 16, 16, 32)	0
conv_4 (Conv2D)	(None, 16, 16, 64)	18,496
max_pooling_4 (MaxPooling2D)	(None, 8, 8, 64)	0
conv_5 (Conv2D)	(None, 8, 8, 128)	73,856
max_pooling_5 (MaxPooling2D)	(None, 4, 4, 128)	0
flatten_7 (Flatten)	(None, 2048)	0
dense_21 (Dense)	(None, 32)	65,568
dense_22 (Dense)	(None, 16)	528
dense_23 (Dense)	(None, 1)	17

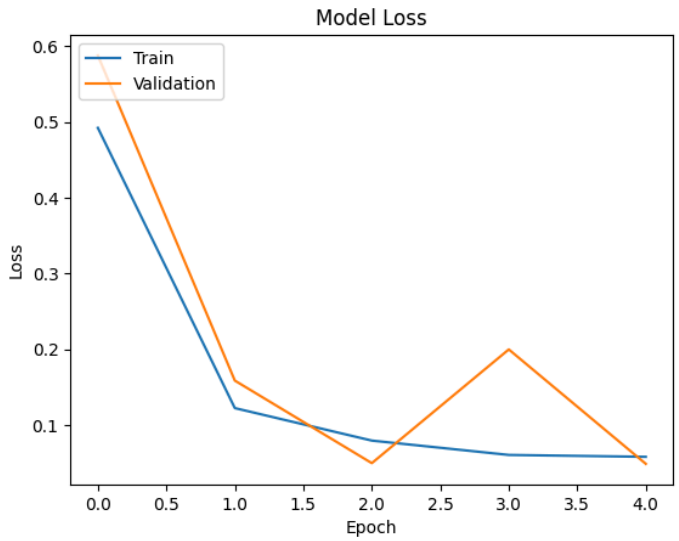
Total params: 164,353 (642.00 KB)

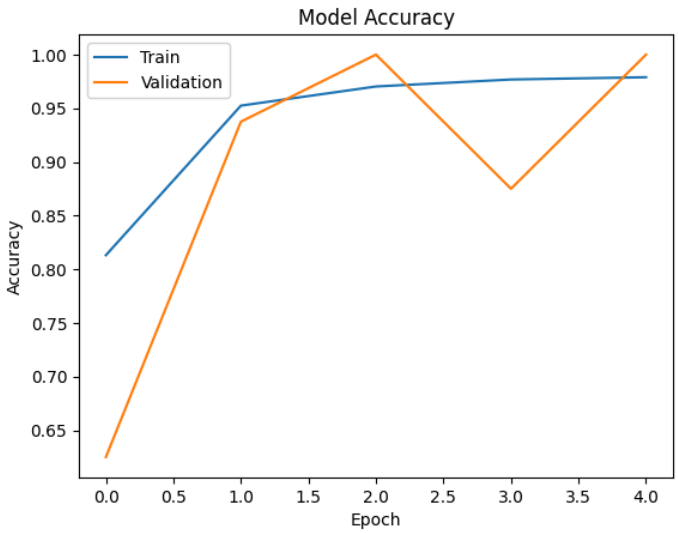
Trainable params: 164,353 (642.00 KB)

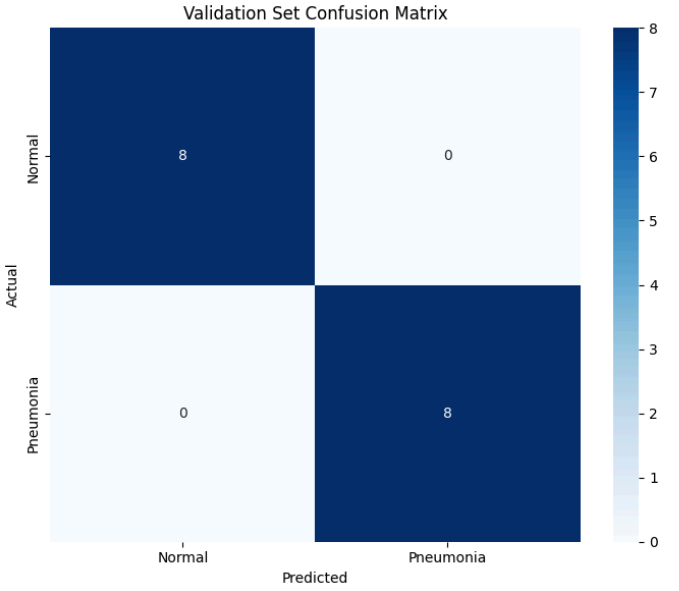
Non-trainable params: 0 (0.00 B)

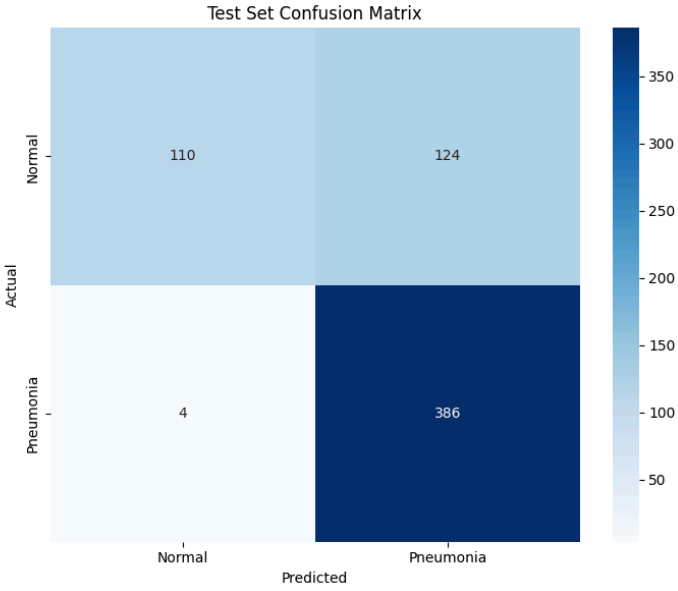

input_layer_10 (InputLayer)
Output shape: (None, 128, 128, 1)
conv_1 (Conv2D)
Input shape: (None, 128, 128, 1)
Output shape: (None, 128, 128, 8)
max_pooling_1 (MaxPooling2D)
Input shape: (None, 128, 128, 8)
Output shape: (None, 64, 64, 8)
conv_2 (Conv2D)
Input shape: (None, 64, 64, 8)
Output shape: (None, 64, 64, 16)
max_pooling_2 (MaxPooling2D)
Input shape: (None, 64, 64, 16)
Output shape: (None, 32, 32, 16)
conv_3 (Conv2D)
Input shape: (None, 32, 32, 16)
Output shape: (None, 32, 32, 32)
max_pooling_3 (MaxPooling2D)
Input shape: (None, 32, 32, 32)
Output shape: (None, 16, 16, 32)
conv_4 (Conv2D)
Input shape: (None, 16, 16, 32)
Output shape: (None, 16, 16, 64)
max_pooling_4 (MaxPooling2D)
Input shape: (None, 16, 16, 64)
Output shape: (None, 8, 8, 64)
conv_5 (Conv2D)
Input shape: (None, 8, 8, 64)
Output shape: (None, 8, 8, 128)
max_pooling_5 (MaxPooling2D)
Input shape: (None, 8, 8, 128)
Output shape: (None, 4, 4, 128)
flatten_7 (Flatten)
Input shape: (None, 4, 4, 128)
Output shape: (None, 2048)
dense_21 (Dense)
Input shape: (None, 2048)
Output shape: (None, 32)
dense_22 (Dense)
Input shape: (None, 32)
Output shape: (None, 16)
dense_23 (Dense)
Input shape: (None, 16)
Output shape: (None, 1)

7.2. Graphes et matrices des résultats









7.3. Analyse des Résultats du CNN pour la Détection de Pneumonie

Matrices de Confusion

	Prédit: Normal	**Prédit: Pneumonie**
Validation - Réel: Normal	100%	0%
Validation - Réel: Pneumonie	0%	100%
Test - Réel: Normal	47%	53%
Test - Réel: Pneumonie	1%	99%

Comparaison des Performances entre Validation et Test

Métrique	Validation	Test
Accuracy	1.0000	0.7900
F1 score (Normal)	1.00	0.63
F1 score (Pneumonia)	1.00	0.86
F1 macro avg	1.0000	0.7400
F1 weighted avg	1.0000	0.7700
Recall (Normal)	1.00	0.47
Recall (Pneumonia)	1.00	0.99
Precision (Normal)	1.00	0.96
Precision (Pneumonia)	1.00	0.76

7.3.1. Discussion des Résultats

Notre modèle CNN pour la détection de pneumonie sur radiographies thoraciques présente des performances contrastées entre les ensembles de validation et de test.

7.3.2. Performance sur l'Ensemble de Validation

Le modèle obtient des résultats parfaits sur l'ensemble de validation avec une précision de 100% et des scores F1 de 1.00 pour les deux classes. Ces résultats, bien qu'impressionnants en apparence, soulèvent des inquiétudes quant à un possible surapprentissage.

7.3.3. Performance sur l'Ensemble de Test

Sur l'ensemble de test, le modèle présente:

- Une précision globale de 79%, un résultat acceptable mais révélant des problèmes sous-jacents
- Une forte asymétrie dans les performances par classe:
  - Excellente détection des cas de pneumonie (recall de 99%)
  - Faible détection des cas normaux (recall de seulement 47%)

7.3.4. Analyse du Problème de Surapprentissage

L'écart significatif entre les performances de validation et de test indique clairement un surapprentissage. Le modèle a "mémorisé" les données de validation plutôt que d'apprendre des caractéristiques généralisables aux nouvelles données.

7.3.5. Implications Cliniques

Dans un contexte médical, notre modèle actuel:

- Détecterait presque tous les cas de pneumonie (haute sensibilité)
- Mais génèrerait de nombreux faux positifs (faible spécificité)

Exemple de topologie de réseau pour la définition des caractéristiques d'entrée/sortie et le calcul du nombre de poids Cette configuration pourrait être adaptée comme outil de dépistage initial, où manquer un cas de pneumonie (faux négatif) serait plus problématique que de signaler à tort un cas normal comme pneumonie (faux positif). Cependant, un taux élevé de faux positifs augmenterait la charge de travail des radiologues devant confirmer ces diagnostics.