4I-SI2 - Deep Learning - Handwritten Digits Recognition

Charles Batchaev, Safi Hanifa

May 31, 2024

Sommaire

1 Introduction

Ce projet vise à développer un réseau de neurones apte à identifier des chiffres écrits à la main. Nous débuterons avec un perceptron simple avant de progresser vers un réseau multicouche perceptron (MLP), et de conclure avec l'expérimentation d'un réseau de neurones convolutif (CNN). Nous évaluerons et comparerons les performances de ces différents modèles pour comprendre leurs forces et leurs faiblesses.

Pour former ces réseaux, nous utiliserons le jeu de données MNIST, qui se compose d'images de chiffres manuscrits. Nous évaluerons et comparerons les performances de chaque modèle afin d'identifier le plus efficace. Dans le cadre de ce projet, nous emploierons des outils technologiques tels que Keras, une bibliothèque Python dédiée aux réseaux de neurones, ainsi que Matplotlib et TensorFlow.

Notre analyse portera sur plusieurs architectures de réseaux de neurones, des perceptrons multicouches aux réseaux convolutifs complexes, afin de déterminer leur efficacité dans la classification des chiffres du jeu de données MNIST.

2 Prétraitement des Données

Le prétraitement des données est une étape cruciale avant l'entraînement des modèles de réseaux de neurones. Dans cette section, nous décrivons le processus appliqué au jeu de données MNIST pour optimiser les performances des modèles en adaptant les images pour un traitement neuronal efficace.

2.1 Normalisation

La normalisation des images est essentielle en Deep Learning. Ce processus ajuste les valeurs de pixels de 0 à 255.0 à une échelle de 0 à 1, ce qui améliore la convergence des réseaux durant l'entraînement. Pour éviter toute erreur, nous normalisons les entrées de notre jeu de données (`entrainement_images` et `test_images`). Les `entrainement_images` contiennent 60,000 images et les `test_images` en contiennent 10,000, chaque image mesurant 28x28 pixels. Les images sont ensuite converties en `float32` et chaque pixel est divisé par 255.

2.2 Vectorisation des Images

Pour adapter les données à l'entraînement de réseaux de neurones denses, les images de 28x28 pixels sont transformées en vecteurs de 784 valeurs. Cette transformation permet de traiter chaque pixel de l'image comme une caractéristique individuelle lors de l'apprentissage.

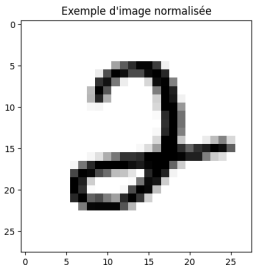


Figure 1: Exemple de chiffre normalisé du jeu de données MNIST

2.3 Encodage des Étiquettes

Les étiquettes, qui sont des chiffres de 0 à 9, sont converties en un format "one-hot". Cette technique de codage consiste à transformer les étiquettes en vecteurs de 10 dimensions où chaque dimension correspond à un chiffre potentiel.

La préparation méticuleuse des données est essentielle pour la performance des modèles de Deep Learning à développer, comme nous le verrons dans les sections ultérieures de ce document. Notons que cette transformation des images est spécifique aux modèles à couches réduites.

3 Implémentation d'un réseau de neurones multicouches pour la classification des données MNIST

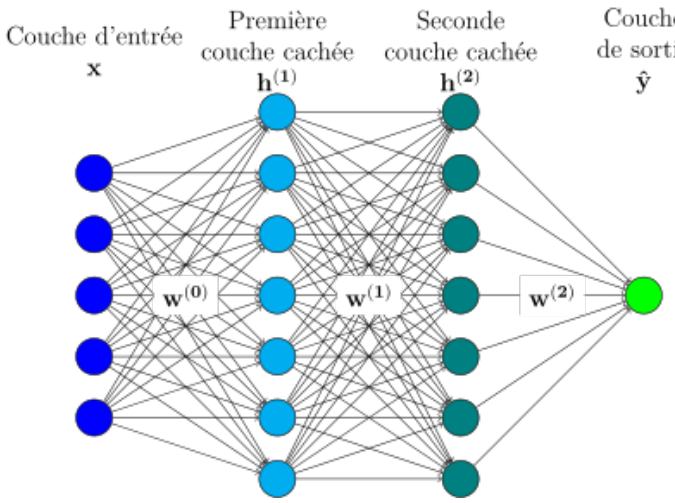


Figure 2: Réseau multicouche avec deux couches cachées

Un perceptron est un type de neurone artificiel utilisé dans les réseaux de neurones. Il prend plusieurs entrées binaires, les multiplie par des poids, et génère une sortie binaire. La formule mathématique pour le perceptron est :

$$\begin{cases} 1 & \text{si } \sum_i w_i x_i > \text{seuil} \\ 0 & \text{sinon} \end{cases}$$

où w_i sont les poids et x_i sont les entrées. Le perceptron effectue une classification binaire en ajustant les poids et le seuil pour apprendre à partir des données d'entraînement.

3.1 Perceptron simple

Ce projet a débuté par la création d'un perceptron simple, sans couches cachées, utilisant la fonction `entrainer_model`. Nous avons conçu un modèle doté d'une couche d'entrée contenant 784 neurones, correspondant aux pixels d'une image de 28x28, et d'une couche de sortie avec 10 neurones, chacun représentant une classe de chiffre. Pour la couche de sortie, nous avons opté pour une activation softmax, et la perte a été évaluée à l'aide de la fonction *categorical_crossentropy*, tandis que la performance a été mesurée par la métrique *accuracy*.

Au cours de nos premiers tests, nous avons modifié le nombre d'epochs. Il en ressort que l'augmentation des epochs améliore la précision, mais de façon non linéaire. Nous avons atteint une précision de test de 92,10% avec 10 epochs, 92,43% avec 20 epochs et 94,97% avec 30 epochs. Cependant, un nombre trop élevé d'epochs peut conduire à un surapprentissage, observable lorsque la précision de validation stagne ou diminue après 20 epochs. C'est pourquoi nous avons décidé de garder 20 epochs comme référentiel.

3.2 Perceptron à deux couches

Le modèle a été ensuite enrichi par l'ajout d'une couche cachée, introduisant une complexité supplémentaire grâce à une couche *Dense* activée par ReLU. Cette modification a nettement amélioré les performances par rapport au modèle initial. Nous avons également varié le nombre de neurones dans cette couche cachée pour étudier leur impact sur le modèle, concluant que si l'augmentation des neurones améliore la précision, elle peut aussi favoriser le surapprentissage.

3.3 Perceptron multicouche

L'expérimentation s'est poursuivie avec l'introduction de plusieurs couches cachées, en variant le nombre de couches et de neurones. Avec deux couches cachées, nous avons observé une précision de 97.8%, mais des signes de surapprentissage dès la 7ème epoch. Avec trois couches cachées, la précision a légèrement augmenté à 97.97%, mais le modèle a continué de montrer des signes de surapprentissage. Avec quatre couches cachées, la précision a atteint 97.61%, confirmant une tendance similaire.

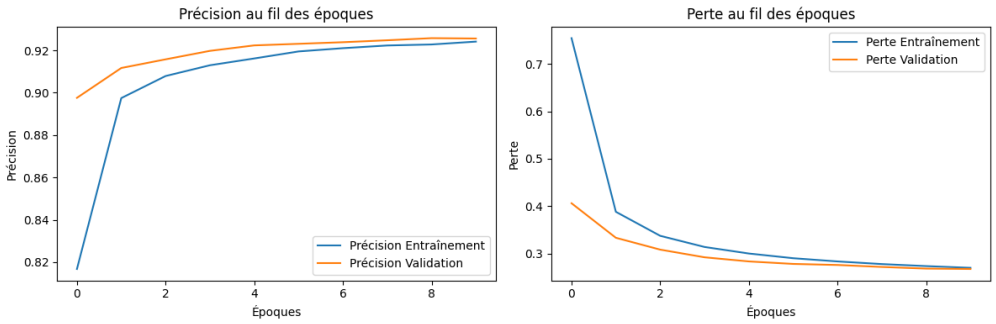
Bien que l'ajout de couches cachées ait légèrement amélioré la précision, il a également accentué le risque de surapprentissage. Il serait pertinent d'explorer d'autres méthodes pour le contrôler, telles que la régularisation ou l'utilisation de dropout. Les matrices de confusion illustrent également que les modèles commettent certaines erreurs plus fréquemment, souvent dues à des similitudes entre les chiffres.

4 Évaluation des Performances des Modèles

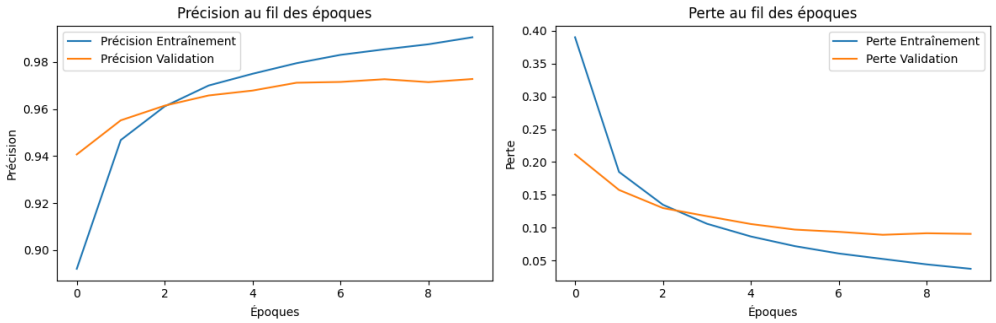
Pour évaluer les performances de nos modèles de perceptrons, nous avons tracé les courbes d'apprentissage et analysé les matrices de confusion pour chaque modèle.

4.1 Courbes d'Apprentissage

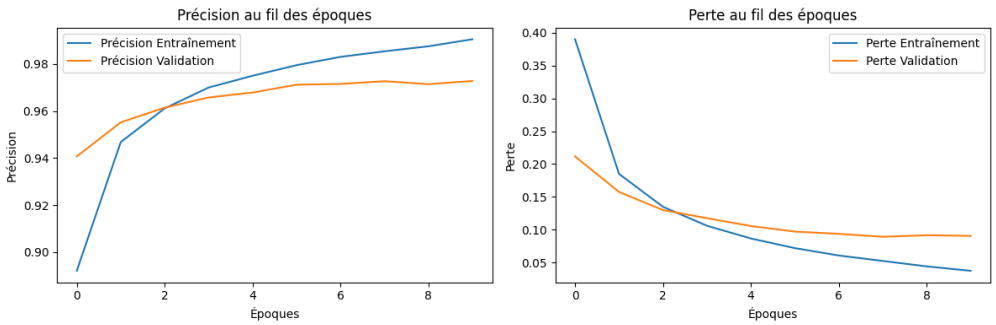
La figure suivante présente une grille de graphiques illustrant l'évolution de l'accuracy durant les dix époques d'entraînement pour chaque modèle, montrant à la fois l'accuracy en entraînement (bleu) et en validation (orange).



(a) Modèle à 1 couche



(b) Modèle à 2 couches



(c) Modèle à 3 couches

Figure 3: Évolution de l'accuracy durant les dix époques d'entraînement pour chaque modèle.

Évaluation Approfondie

- **Perceptron à 1 couche** : Ce modèle démontre une convergence rapide au début de l'entraînement, ce qui indique une forte capacité initiale à apprendre les caractéristiques des données. Cependant, la précision en validation atteint rapidement un plateau et commence à diverger légèrement de la précision en entraînement, suggérant un début de surajustement.

- **Perceptron à 2 couches** : La précision en entraînement progresse de manière régulière, tandis que la précision en validation, après une hausse initiale, commence à décroître légèrement. Cette tendance suggère que le modèle commence à surajuster après quelques époques, bien que ce phénomène soit moins marqué que dans le modèle à une seule couche.

- **Perceptron à 3 couches** : la précision du modèle d'entraînement augmente régulièrement pour atteindre plus de 98%, tandis que la précision de validation reste légèrement inférieure mais proche de 97%. La perte diminue également, indiquant une bonne convergence du modèle.

4.2 Prédictions sur les Images de Test

Pour ce faire, nous utilisons un modèle de réseau de neurones à trois couches (`modele_3couches`) pour prédire les classes des images de test. Les variables `predicted_classes` et `true_classes` servent à comparer les prédictions du modèle aux étiquettes réelles des images.

Nous affichons ensuite les 200 premières images de test avec une couleur verte pour les prédictions correctes et rouge pour les incorrectes. Cela permet de visualiser où le modèle fait des erreurs.

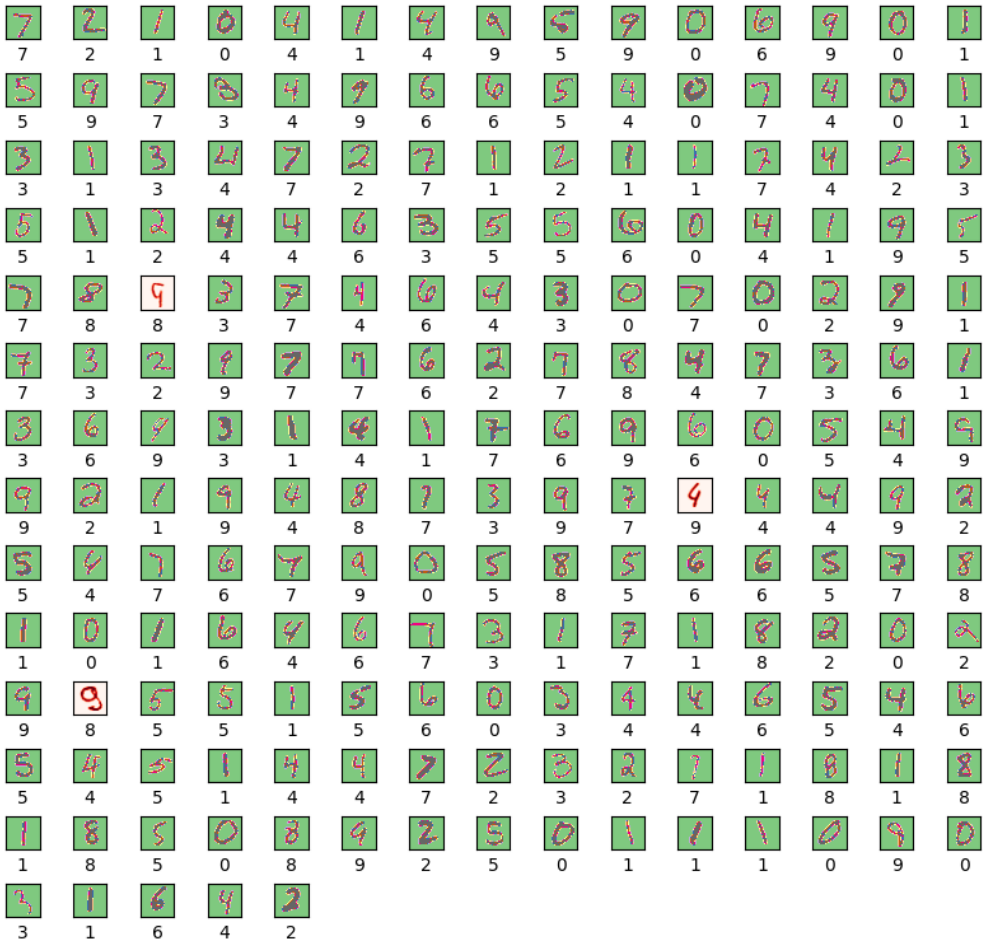


Figure 4: Prédictions sur les images de test. Les chiffres correctement prédits sont affichés en vert, tandis que ceux mal prédits sont en rouge.

Résultats et Observations

- **Précision** : La majorité des prédictions sont correctes, indiquées par la dominance de la couleur verte.
- **Erreurs** : Quelques erreurs de prédiction apparaissent en rouge. Par exemple, un 9 prédit comme un 8 ou 4.

- **Visualisation** : Permet d'identifier visuellement où le modèle fait des erreurs et d'analyser les motifs récurrents dans les erreurs de classification.

5 Approche en utilisant les Réseaux de Neurones Convolutionnels (CNN)

Les réseaux de neurones convolutifs (CNN) ont révolutionné le domaine de la vision par ordinateur en raison de leur capacité à capturer des motifs spatiaux et à extraire des caractéristiques pertinentes à partir d'images. Dans cette section, nous explorerons l'application des CNN à la tâche de reconnaissance de chiffres manuscrits en utilisant le jeu de données MNIST.

5.1 Architecture du Modèle CNN

Le modèle CNN que nous avons mis en place comprend plusieurs couches, chacune jouant un rôle crucial dans le processus de traitement de l'image et de classification des chiffres manuscrits.

- **Couche de Convolution** : Cette couche applique des filtres pour détecter des caractéristiques telles que les bords, les coins et les textures dans les images. Les filtres sont appliqués de manière convolutive sur l'image pour générer des cartes de caractéristiques.

- **Couche de Pooling** : Après chaque couche de convolution, une couche de pooling est ajoutée pour réduire la dimensionnalité des cartes de caractéristiques. Cela permet de conserver les informations importantes tout en réduisant le nombre de paramètres du modèle.

- **Couche Entièrement Connectée** : Ces couches finales combinent les caractéristiques extraites par les couches de convolution et de pooling pour effectuer la classification des chiffres. Les résultats sont transmis à une couche softmax pour obtenir les probabilités de chaque classe.

5.2 Entraînement et Évaluation du Modèle CNN

Nous avons entraîné le modèle CNN en utilisant le jeu de données MNIST prétraité. Pendant l'entraînement, nous avons utilisé des techniques telles que la régularisation et le dropout pour prévenir le surapprentissage et améliorer la généralisation du modèle.

Une fois l'entraînement terminé, nous avons évalué les performances du modèle en utilisant les images de test du jeu de données MNIST. Nous avons analysé la précision du modèle ainsi que les erreurs de classification pour évaluer sa robustesse et son efficacité dans la reconnaissance de chiffres manuscrits.

5.3 Résultats et Observations

Les performances du modèle CNN ont été impressionnantes, avec une précision élevée sur les images de test. L'analyse des prédictions a révélé que la majorité des chiffres ont été correctement identifiés, avec seulement quelques erreurs mineures.

En comparaison avec les modèles précédents, le CNN a démontré une capacité supérieure à capturer les motifs complexes et à généraliser à de nouvelles données. Sa capacité à apprendre des caractéristiques spatiales des images lui confère un avantage significatif dans la tâche de reconnaissance de chiffres manuscrits.

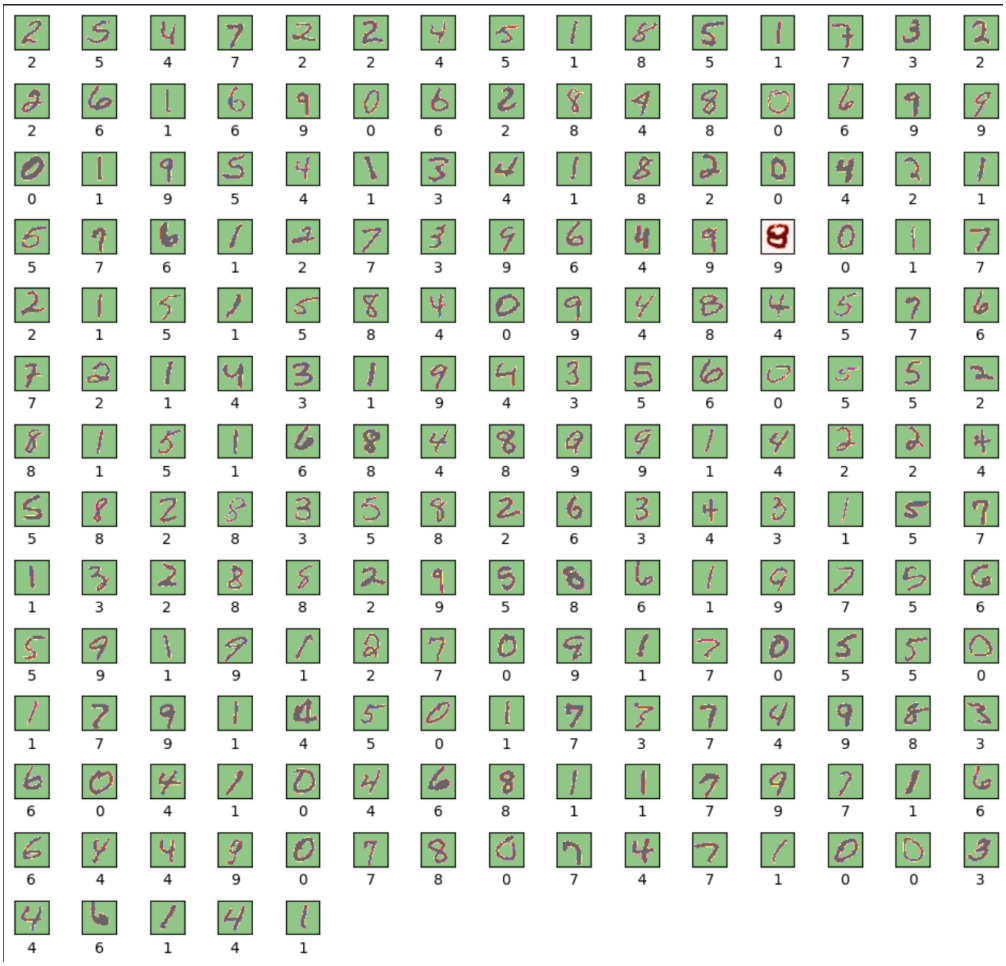


Figure 5: Prédictions sur les images de test. Les chiffres correctement prédits sont affichés en vert, tandis que ceux mal prédits sont en rouge.

5.4 Prédictions sur les Images de Test

Pour évaluer visuellement les performances du modèle, nous avons sélectionné quelques images de test et leurs étiquettes associées. Nous avons ensuite utilisé le modèle CNN pour prédire les classes de ces images et les avons comparées avec les étiquettes réelles. Les

prédictions correctes sont affichées en vert, tandis que les prédictions incorrectes sont affichées en rouge.

Résultats et Observations

- **Précision** : En observant la coloration des images, on constate que la majorité des prédictions sont correctes, comme indiqué par la dominance de la couleur verte. Cela suggère que le modèle CNN a une capacité élevée à classer correctement les images.
- **Erreurs** : Dans notre exemple, une seule erreur est observée, où le modèle prédit le chiffre "9" au lieu de "8". Cette erreur est représentée en rouge sur l'image correspondante.
- **Visualisation** : Permet d'identifier visuellement où le modèle fait des erreurs et d'analyser les motifs récurrents dans les erreurs de classification.

Comme illustré à la figure 5, la majorité des prédictions du modèle CNN sont correctes, ce qui est indiqué par la dominance de la couleur verte. Cependant, il reste néanmoins parfois des erreurs, ici nous avons un 9 prédit au lieu d'un 8. Cette visualisation permet d'identifier visuellement où le modèle fait des erreurs et d'analyser les motifs récurrents dans les erreurs de classification.

5.5 Courbes d'Apprentissage et Évaluation Approfondie

Pour une analyse plus détaillée, nous avons examiné les courbes d'apprentissage du modèle CNN, montrant l'évolution de l'accuracy et de la perte sur les données d'entraînement et de validation au fil des époques.

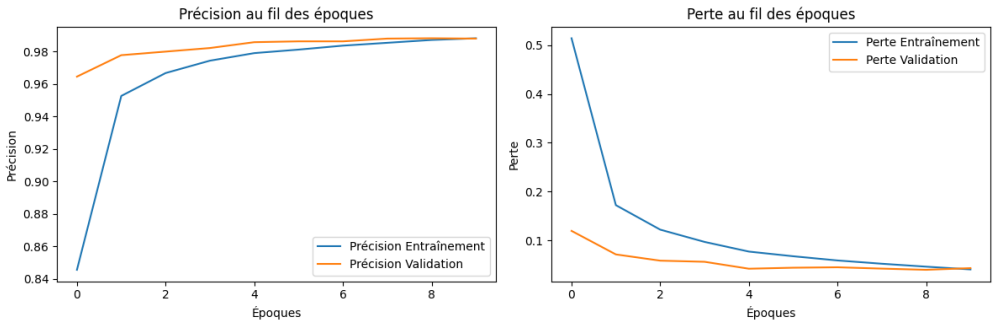


Figure 6: Courbes d'apprentissage du modèle CNN montrant l'évolution de l'accuracy et de la perte pour les données d'entraînement et de validation.

- **Courbe d'Accuracy** : La courbe d'accuracy montre une amélioration continue de la précision tant sur les données d'entraînement que de validation. La précision en validation atteint des valeurs élevées, démontrant une bonne capacité de généralisation du modèle. La courbe d'accuracy de la figure **??** montre que la précision augmente régulièrement et se stabilise autour de 99%.

- **Courbe de Perte** : La courbe de perte diminue régulièrement, indiquant une réduction progressive de l'erreur du modèle. La perte en validation reste proche de celle de l'entraînement, suggérant un faible surajustement. La figure **??** montre que la perte diminue rapidement au début de l'entraînement et se stabilise par la suite.

5.5.1 Matrice de Confusion

La matrice de confusion suivante illustre les performances du modèle CNN sur les images de test. Elle met en évidence les classes correctement prédites ainsi que les erreurs les plus fréquentes.

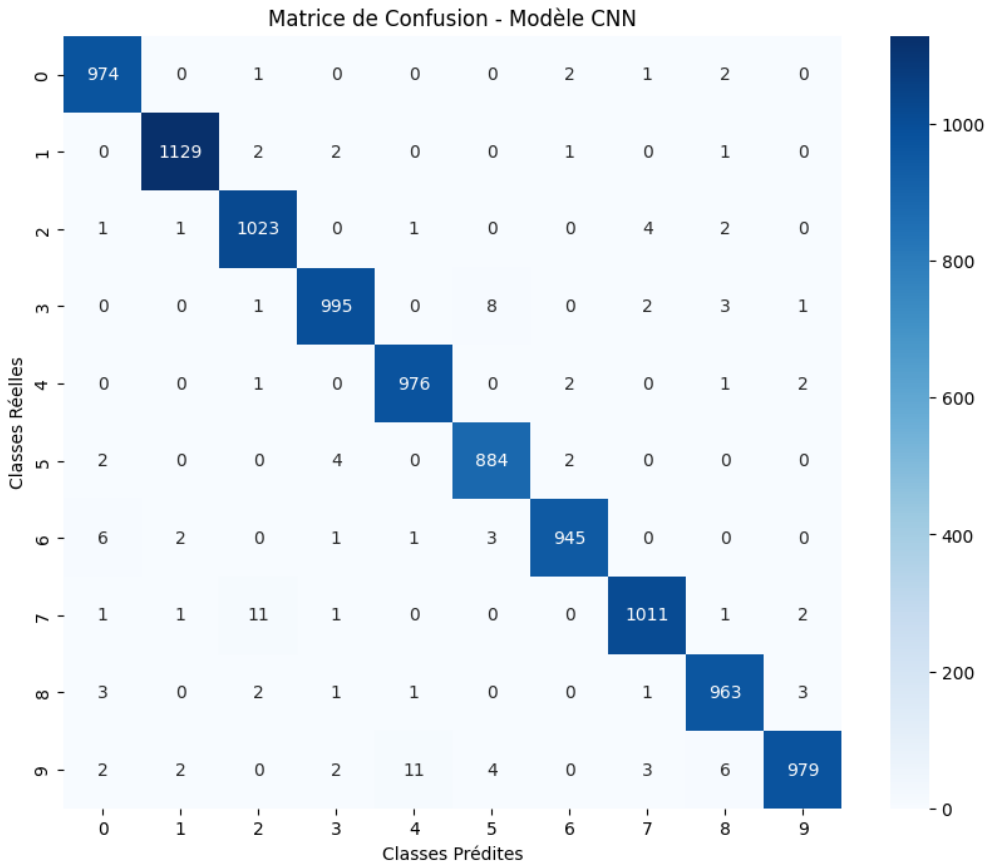


Figure 7: Matrice de confusion du modèle CNN sur les images de test du jeu de données MNIST

6 Conclusion générale

En ajustant l'architecture des réseaux de neurones, par exemple en augmentant le nombre de neurones dans une couche cachée ou en intégrant des couches supplémentaires, il est possible d'améliorer les capacités du modèle à représenter des données de manière plus complexe.

Toutefois, ces modifications peuvent compliquer l'entraînement et élever le risque de surapprentissage. Il est essentiel de suivre de près les changements dans les mesures de perte et de précision sur l'ensemble de validation pour identifier le surapprentissage précoce.