

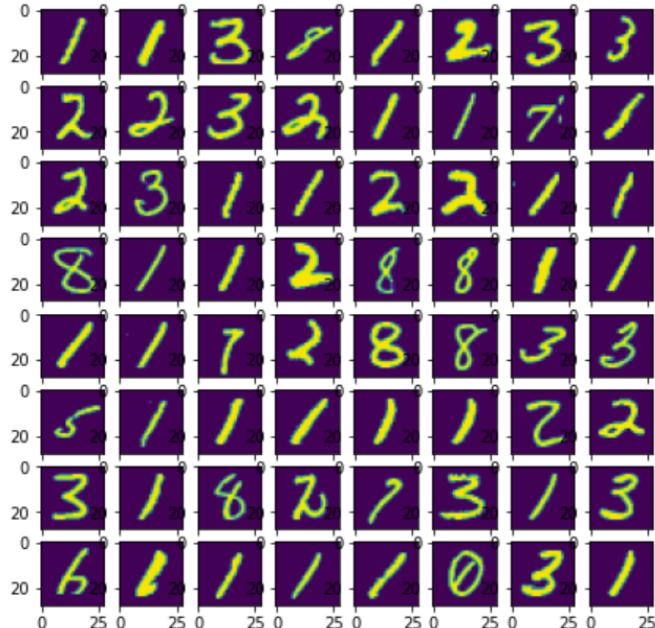
# **Clustering et Reconnaissance de chiffres manuscrits**

## **Table des matières :**

- 1. Introduction**
- 2. Classification non supervisée des chiffres**
  - 2.1 Premières implémentations du K-Mean
  - 2.2 Nombre de classes optimal
  - 2.3 Comparaison des performances pour différentes initialisations
  - 2.4 Comparaison au K-Medoid
- 3. Classification supervisée des chiffres**
  - 3.1 Implémentation du MLP
  - 3.2 Implémentation du CNN
- 4. Comparaison des méthodes et conclusion générale**

## 1. Introduction

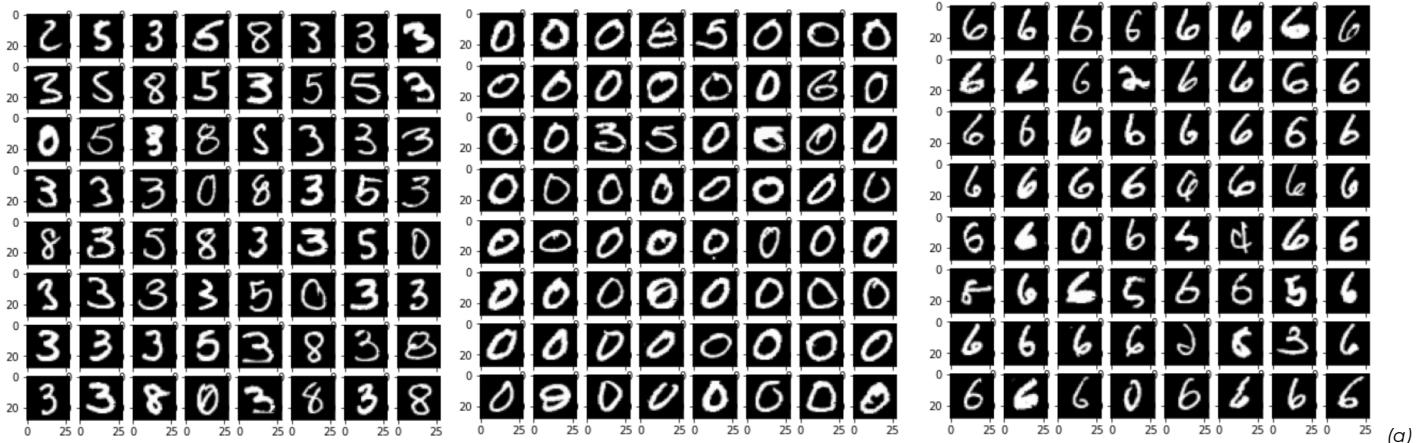
Ce rapport porte sur l'étude de représentation de chiffres manuscrits acquis sur des outils numériques. Pour cela, la base données MNIST est utilisée, celle-ci réunit un ensemble de 70000 exemples soit 7000 exemples de chaque chiffre, ainsi que leurs étiquettes, afin de permettre de catégoriser les représentations dans différentes classes. Chaque image est de 28 x 28 pixels, et chaque pixel est représenté par un nombre (niveau de gris). Ces matrices peuvent être aplatis en vecteurs de  $28 \times 28 = 784$  nombres. On peut alors voir chaque image comme un point dans un espace vectoriel à 784 dimensions. On travaillera ici sur les 10 000 premières données d'entraînement.



## 2. Classification non-supervisée des chiffres

### 2.1 Première implémentation du K-Mean

Une première idée pour classifier les chiffres allant de 0 à 9 est la classification non supervisée et consiste simplement à regrouper entre eux les chiffres écrits d'une manière similaire. Dans cet objectif, l'algorithme du K-Mean vu en cours est envisagé. On effectue alors l'hypothèse intuitive que le K idéal vaut K=10, un cluster correspondant à chaque chiffre selon nous. La classe prédéfinie par scikit learn donnant des résultats assez probants pour cette base de données dans la littérature elle sera utilisée ici. Pour l'instant, une initialisation aléatoire est utilisée. (figure 1)



Cluster estimé : 3 Cluster estimé : 7 Cluster estimé : 1 Cluster estimé : 6 Cluster estimé : 0 Cluster estimé : 2 Cluster estimé : 7 Cluster estimé : 1 Cluster estimé : 4 Cluster estimé : 8

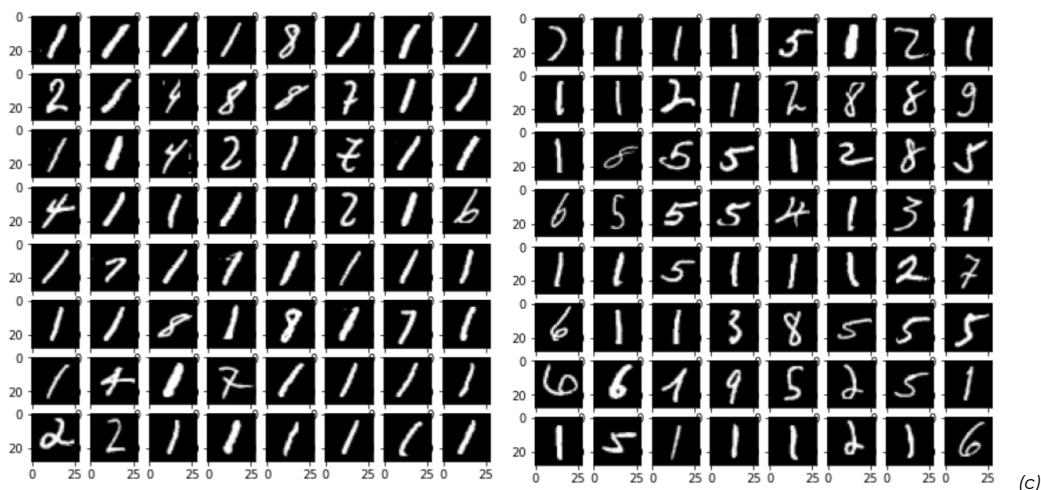
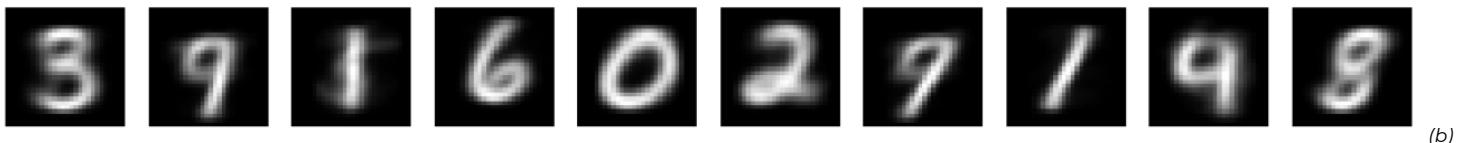


Figure 1a) : Représentation d'un cluster de 3, de 0 et de 6 pour le K-Mean avec K=10

Figure 1b) : Visualisation des centroids pour chaque classe pour le K-Mean avec K=10. Chaque centroid étant la moyenne des individus du cluster. Le chiffre du cluster estimé correspondant au chiffre majoritaire du cluster

Figure 1c) : Comparaison des deux clusters de 1

La figure 1 fournit premièrement une représentation de 64 individus tirés au hasard dans chacun des 10 clusters ainsi que la visualisation du représentant de chacun des clusters. On constate ainsi que les clusters ne tendent pas à se former selon le chiffre écrit par l'individu mais selon l'écriture des individus. Ce phénomène est explicable par le fait que les individus dont proviennent les chiffres écrits dans la base de données possèdent chacun des façons différentes d'écrire un chiffre. On constate par exemple que les 1 écrits dans le premier cluster estimé à 1 diffèrent des 1 du second cluster estimé à 1, les individus du premier cluster tendent effectivement à écrire leur plus penché que ceux du second cluster. La figure 2 illustre la précision du K-Mean pour détecter les chiffres de la base de données. Le tableau 1 fournit quant à lui des valeurs d'inertie intra-classe, d'homogénéité et de précision pour le K-Mean.

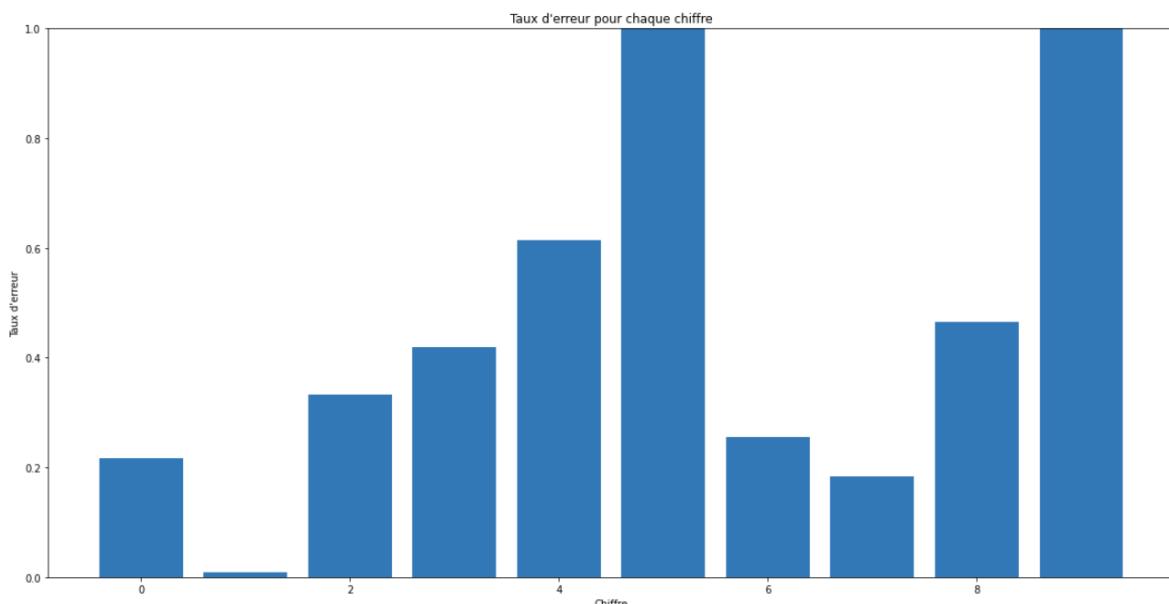


Figure 2 : Taux d'erreur pour la prédiction de chaque chiffre pour K-Mean avec K=10.

Inertie intra-classe	388132.96
Homogénéité	0.46
Précision	0.57
Indice de silhouette	0.14

Tableau 1: Données du K-Mean pour 10 clusters.

Pour K=10, il apparaît donc que certains chiffres sont tout le temps mal classés, dans notre cas le 5 et le 9 ne sont tout simplement jamais détectés, le modèle ne considère pas leur existence car leur écriture se rapproche trop de l'écriture d'autres chiffres dans l'échantillon de 10 000 chiffres sélectionnés au

préalable. En revanche, on constate que le taux de précision pour la détection de 1 est très élevé, cela est dû au fait que deux clusters représentent deux nuances dans l'écriture du 1. Le taux d'erreur pour les autres chiffres reste cependant sous la barre des 50% ce qui semble être un échec. Cependant, étant donné la haute variabilité dans l'écriture des individus cela reste correct. Nous pouvons malgré cela optimiser cette méthode de clustering afin de créer des classes d'écritures plus fines et d'ainsi potentiellement mieux détecter les chiffres écrits. Pour cela, on décide d'optimiser le nombre de clusters K. Il est néanmoins important de préciser qu'il semble très difficile d'envisager la méthode du K-Mean pour obtenir un taux de précision élevé pour la classification des chiffres, cette méthode peut cependant permettre d'identifier les différents types d'écriture au sein de la base de données. Pour obtenir une précision assez élevée il faudrait en effet créer un nombre très important de catégories d'écriture comme nous allons le voir.

## **2.2 Choix du nombre de clusters optimal**

### **2.2.a) Choix du nombre de clusters optimal à l'aide d'une méthode hiérarchique**

Dans l'optique de créer des catégories d'écriture les plus adaptées possibles à la base de données, on envisage plusieurs méthodes. La première consiste en la détermination de ce K à l'aide d'une méthode dite hiérarchique. Cette méthode consiste à construire itérativement des clusters à partir de la base de données complète. Dans la méthode hiérarchique que nous employons, initialement, chaque individu constitue un cluster initial et l'algorithme s'arrête lorsqu'il a formé finalement un unique cluster rassemblant tous les individus. Des informations concernant les différentes étapes de formations des clusters est ensuite visible par le biais d'un dendrogramme. (voir figure 3)

Plusieurs mesures existent pour définir la distance entre deux clusters utilisée au cours de cet algorithme.

Après avoir essayé plusieurs mesures (Single Linkage, Average Linkage), les deux méthodes les plus adéquates pour notre base de données s'avèrent être l'Average Linkage (distance utilisée pour calculer la distance

entre deux clusters a et b de taille respective n et m: $d(a, b) = \sum_{i,j=1}^{n,m} \frac{d(a[i], b[j])}{(|a| \times |b|)}$ ) et le

Complete Linkage( $\forall i, j \in [1, n], [1, m], d(a, b) = \max(d(a[i], b[j]))$ ). Elles sont illustrées en Figure 3.

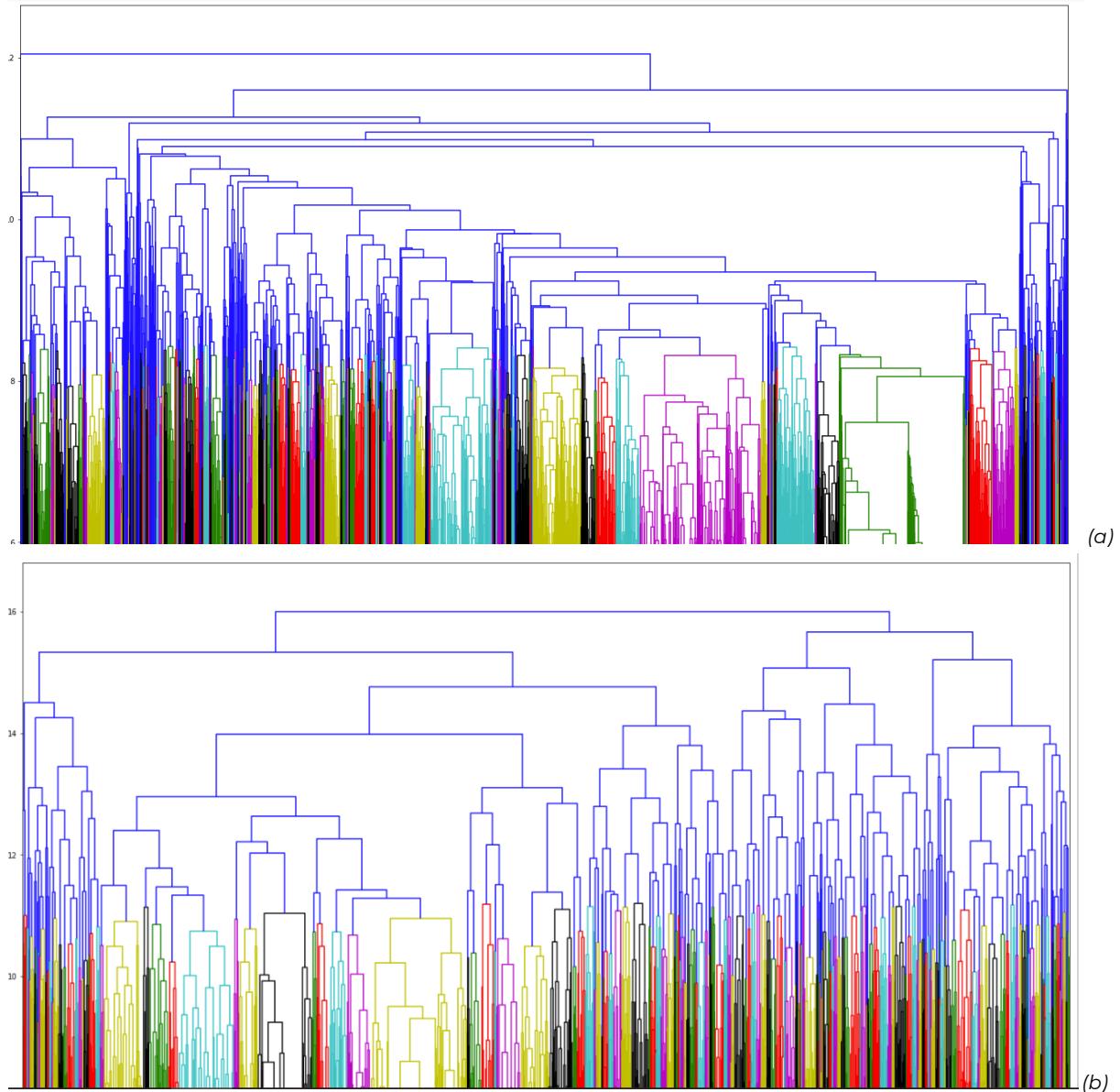


Figure 3 a) : Dendrogramme pour un Average Linkage.

Figure 3 b) : Dendrogramme pour un Complete Linkage.

Pour l'Average Linkage, il apparaît donc que le nombre de clusters idéals semblent se situer aux alentours d'une valeur de la distance située entre 0,5 et 1. La distance entre les différents clusters est trop faible avant et après cette valeur. Pour cette valeur on compte alors **11 à 12** clusters comme nombre idéal ici.

Le Complete Linkage est bien plus clair à lire, on constate par la même méthode que le nombre de clusters maximisant la distance entre les clusters semble être situé entre **11 et 14**.

Cette méthode nous amène donc à conclure que le nombre optimal de clusters se situerait légèrement au-dessus de 10 si l'objectif est de stratifier au mieux la base de données.

## **2.2.b) Choix du nombre de clusters à l'aide de la méthode du coude et de l'indice de silhouette**

Après avoir considéré la distance entre les différents clusters grâce à la méthode hiérarchique, on envisage maintenant de considérer l'inertie intra-classe des clusters grâce à l'utilisation d'une technique spécifique à la méthode de classification K-Mean : la méthode du coude (elbow method en anglais). Cette méthode consiste à observer l'évolution de l'inertie intra-classe des clusters générés par la méthode du K-Mean pour différentes valeurs de K et d'observer à partir de quelle valeur cette inertie se met à chuter. (voir Figure 4)

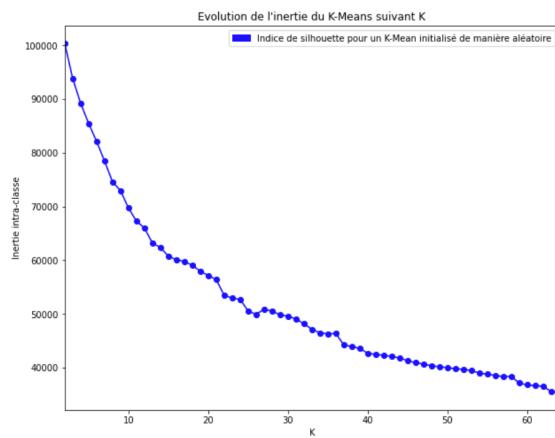


Figure 4 : Evolution de l'inertie intra-classe pour un K-Mean initialisé aléatoirement suivant K

On observe alors graphiquement que la chute de l'inertie semble s'accentuer pour une valeur de **K=15**.

Une seconde méthode pour estimer le nombre optimal K consiste à choisir le K pour lequel l'indice de silhouette est maximum. En effet, par le calcul de l'indice silhouette, celui-ci sera d'autant plus proche de 1 que l'ensemble des points appartenant à un cluster seront proches de leur cluster et éloignés des autres clusters. (voir Figure 5)

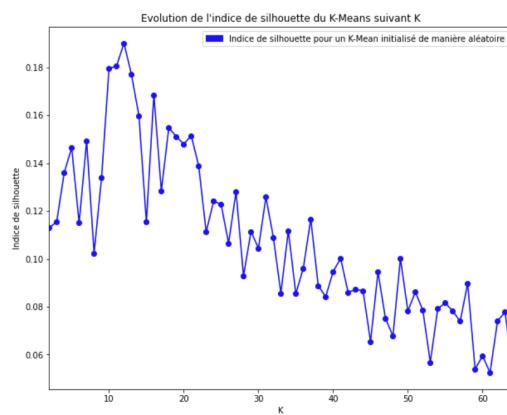


Figure 5 : Evolution de l'indice de silhouette pour un K-mean initialisé aléatoirement

Le calcul de l'indice de silhouette pour différentes valeurs de K nous indique que la valeur optimale de clusters pour minimiser l'inertie intra-classe serait **K=12**. L'ensemble de ces valeurs optimales sont cependant **difficiles à déterminer** (particulièrement pour l'elbow method) ce qui laisse à penser que l'optimalité demeure assez faible ou que le K optimal se trouve bien au-delà de 20.

Le tableau 2 fournit alors les valeurs obtenues pour un K-Means avec un nombre de clusters optimal pour l'inertie intraclasse et interclasse. Ces résultats apparaissent certes meilleurs que pour K=10 mais la différence est moindre surtout si l'on compare ces résultats à un K-Mean pour K=64, on conclut ainsi que la classification supervisée du K-Mean est d'autant plus efficace que le nombre de catégories augmente.

	K=15 (optimal présumé pour l'inertie intra-classe)	K=12 (optimal présumé pour l'inertie inter-classe)	K=64
Inertie intra-classe	76153.65	75638.57	289753.38
Homogénéité	0.30	0.30	0.71
Précision	0.43	0.43	0.82
Indice de silhouette	0,12	0,17	0.09

Tableau 2

On constate ainsi que la précision de la classification ne cesse d'augmenter avec K, cela s'explique par le fait que les écritures de certains chiffres sont de mieux en mieux catégorisées (pour reprendre l'exemple vu avec les deux clusters de 1 pour K=10, on a cette fois-ci plusieurs clusters de 1 représentant les différentes nuances dans l'écriture de 1 pour un K=64). Cela montre finalement qu'une méthode de classification supervisée serait plus adaptée pour augmenter la précision de la prédiction. Il apparaît néanmoins que l'indice de silhouette diminue lorsque le nombre de K augmente et semble atteindre un maximum pour K=12. Finalement, le K optimal trouvé (12 dans notre cas) est valable dans un objectif de clustering des individus mais pas en précision de prédiction, l'objectif des méthodes de clustering non supervisée sera donc d'améliorer cet indice qui consiste en une séparation optimal des manières d'écrire les chiffres. Pour optimiser la précision, une méthode supervisée semble plus adéquate.

## **2.3) Comparaison des performances pour différentes initialisations**

Outre le nombre de clusters, l'initialisation joue également un rôle important dans la performance de la classification. On compare ici ces différentes initialisations ainsi que les différentes méthodes. Les performances évaluées seront le temps d'exécution ainsi que l'indice de silhouette et l'inertie intra-classe.

Le K-Mean étudié en première partie étant initialisé avec une distribution aléatoire des clusters, on envisage ici une autre façon d'initialiser.

La première méthode d'initialisation envisagée est l'initialisation "k-means ++" qui consiste à choisir des points initiaux les plus éloignés possibles les uns des autres. On choisit premièrement un point aléatoirement, et les points les plus éloignés ont ensuite plus de chances d'être sélectionnés comme deuxième représentants. Le même processus est iteré pour choisir les autres représentants, on minimise toujours l'importance des potentiels représentants proches des autres représentants.

Une seconde méthode vise, quant à elle, à minimiser le temps d'exécution. Pour cela, on applique une ACP (Principal Component Analysis) aux données avant d'appliquer le K-Means. L'objectif étant de réduire la dimension du vecteur d'entrée du K-Means afin d'améliorer les performances du modèle, on choisit de réduire à une dimension égale au nombre de clusters testé K, valeur préconisée par la littérature traitant du sujet.

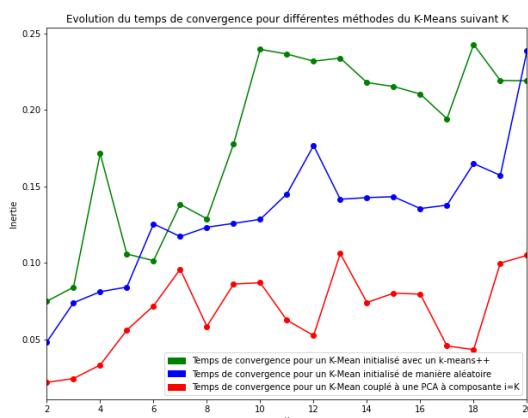


Figure 8 : Évolution du temps de convergence pour les différentes méthodes selon K.

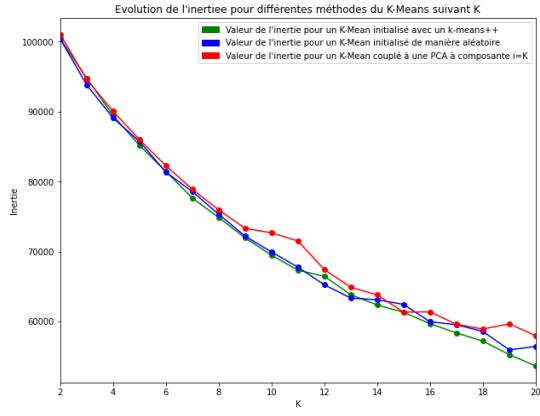


Figure 9 : Évolution de l'inertie intra-classe pour les différentes méthodes selon  $K$ .

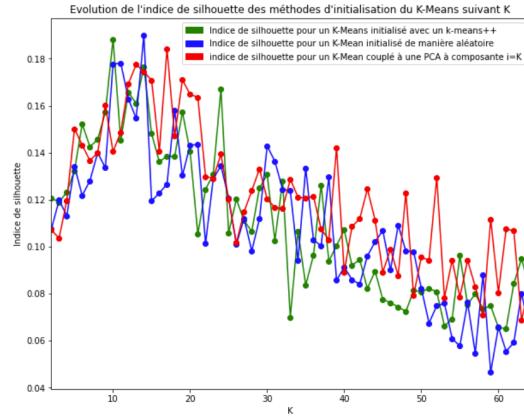


Figure 10 : Évolution de l'indice de silhouette pour les différentes méthodes selon  $K$ .

La figure 8 montre ainsi que l'implémentation d'une PCA avant d'effectuer le K-Means diminue significativement le temps de convergence. L'évolution de l'indice de silhouette présenté en figure 10 indique de plus que le clustering à l'aide de l'initialisation k-means ++ n'a que peu d'influence sur le clustering final dans notre cas. On constate également que le  $K$  optimal est différent pour les différentes initialisations. On note que le  $K$  optimal est situé entre 10 et 20 mais que celui-ci est différent selon l'initialisation envisagée. Il semble donc que cette valeur  $K$  optimal puisse être définie selon l'initialisation privilégiée.

Finalement, le K-Means couplé au PCA atteignant un maximum d'indice de silhouette similaire aux deux autres méthodes et un temps d'exécution plus faible, on privilégie donc cette méthode.

## **2.4) Performances du K-Medoid**

### **2.4 a) Visualisation des performances du K-Medoid**

Dans l'optique d'obtenir de meilleures performances de classification non supervisée, on envisage maintenant d'employer la méthode du K-Medoid utilisant des éléments de la base de données comme centroid contrairement au K-Mean. L'objectif étant de comparer ces deux méthodes de classification. La figure 1 fournit alors le clustering suivant pour K=10 :

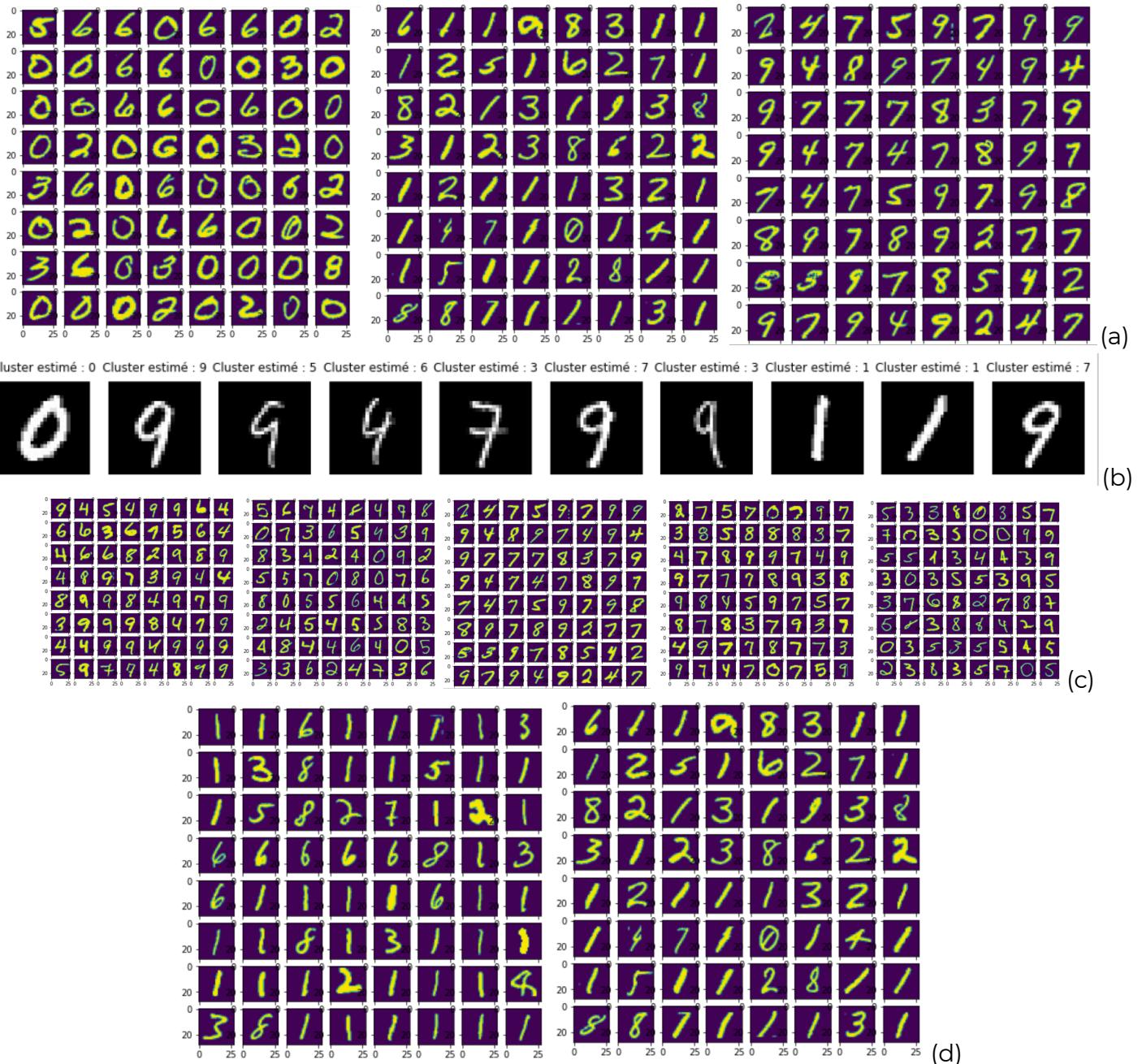


Figure 6 a) : Représentation du cluster estimé 0, du second cluster estimé 1 et du premier cluster estimé 7

Figure 6 b) : Représentation des centroïdes qui sont ici des individus du cluster.

Figure 6 c) : Représentation des clusters dont le centroid est un 9.

Figure 6 d) Représentation des deux clusters estimés à 1.

Le K-Medoid fournit donc des clusters moins représentatifs des chiffres que le K-Mean. On retrouve en effet sur la figure 6 b) que 5 des 10 centroïdes se trouvent être des 9 et qu'un seul d'entre eux est en fait le représentant d'une classe majoritaire en 9. Le K-Medoid semble donc avoir plus de difficultés que le K-Mean pour détecter le chiffre écrit par l'individu. Le choix du centroïde semble également peu pertinent pour  $K=10$ . On constate néanmoins sur la figure 6)c) que les individus dont le représentant est un 9 sont peuplés très différemment. Le chiffre 9 semble ainsi être le chiffre à partir duquel on reconnaît le mieux la variété d'écriture. On effectue également le même constat que pour le K-Mean avec les deux clusters représentant de 1 sur la figure 6 d), la classification est semblable. La précision pour la détection de chaque chiffre est donnée sur la figure 7.

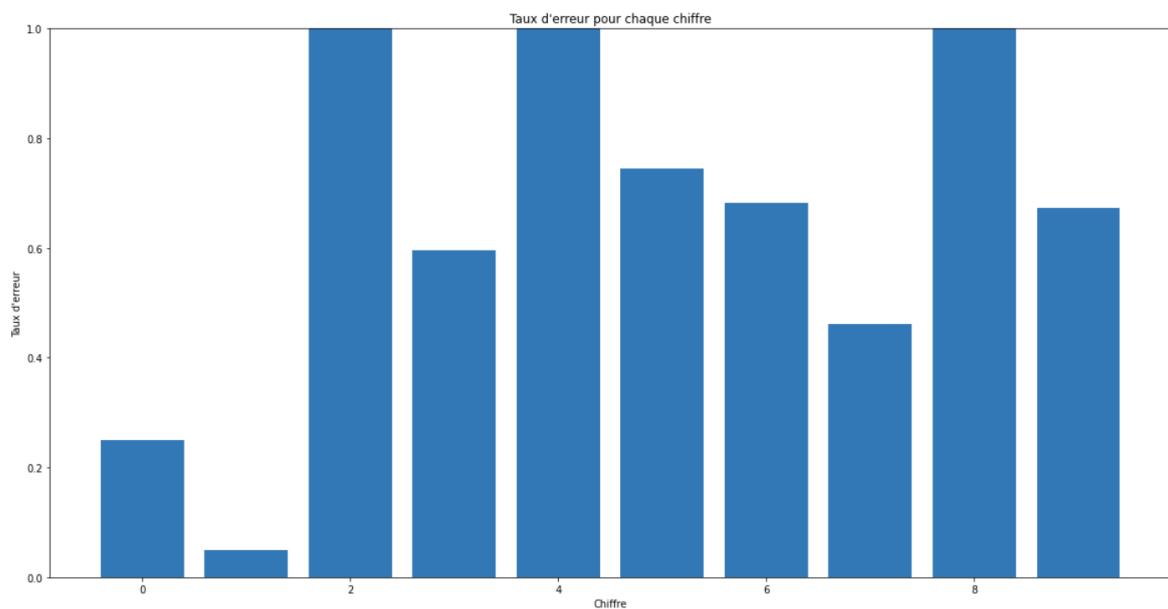


Figure 7 : Taux d'erreur de prédiction pour chaque chiffre pour un K-Medoid avec  $K=10$ .

Que ce soit pour le K-Mean ou pour le K-Medoid à 10 clusters, il semble que le chiffre 1 puisse être écrit de deux manières différentes pour la plupart des individus ce qui conduit à une excellente détection de ce chiffre quelle que soit la méthode non supervisée employée. Le K-Medoid est cependant beaucoup moins performant pour la détection des autres chiffres que le K-Mean. Les valeurs du tableau 2 confirment cela si on les compare à celle du tableau 3.

	K=10	K=64
Inertie intra-classe	77177.36	66595.30
Homogénéité	0.23	0.50
Précision	0.37	0.67
Indice de silhouette	0.12	0.07

Tableau 3 : Données du K-Medoid pour 10 et 64 clusters.

On constate de plus que l'évolution de la précision est globalement améliorée avec l'augmentation du nombre de K. Tout comme pour le K-Mean, on constate également une baisse de l'indice de silhouette avec l'augmentation de K.

#### **2.4.b) Comparaisons des performances entre le K-Mean et le K-Medoid**

On évalue enfin les deux modèles que sont le K-Mean et le K-Medoid sur les critères de temps d'exécution, d'inertie et d'indice de silhouette. On sélectionne le modèle de K-Means couplé au PCA pour effectuer cette comparaison avec le K-Medoid.

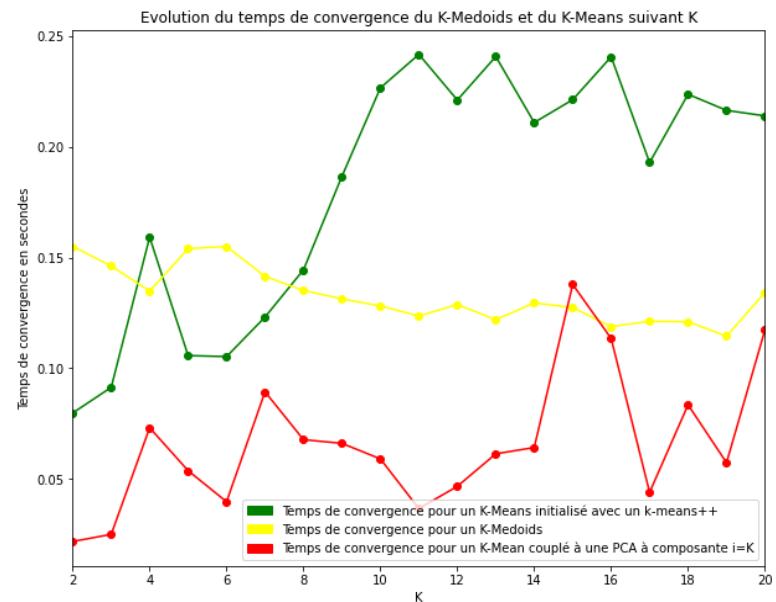


Figure 11 : Comparaison du temps de convergence pour différentes valeurs de K pour le K-Means et le K-Medoid.

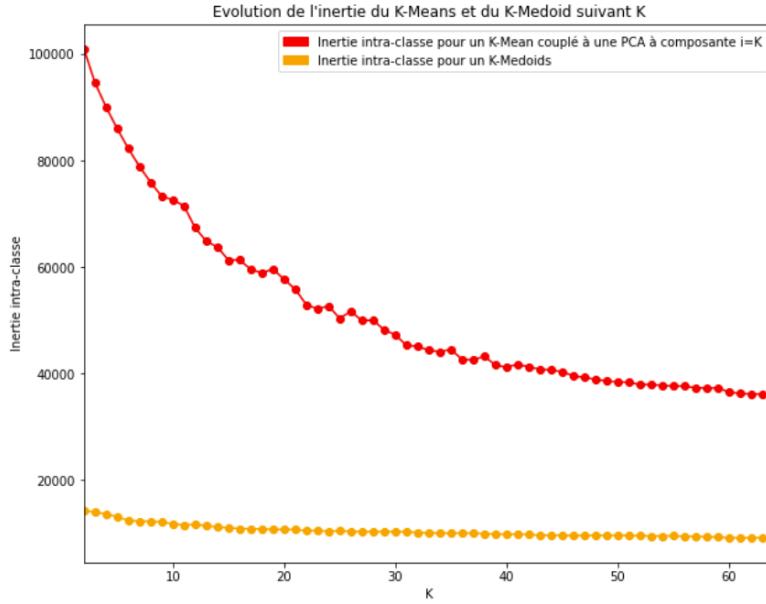


Figure 12 : Comparaison de l'inertie intra-classe pour différentes valeurs de K pour le K-Means et le K-Medoid.

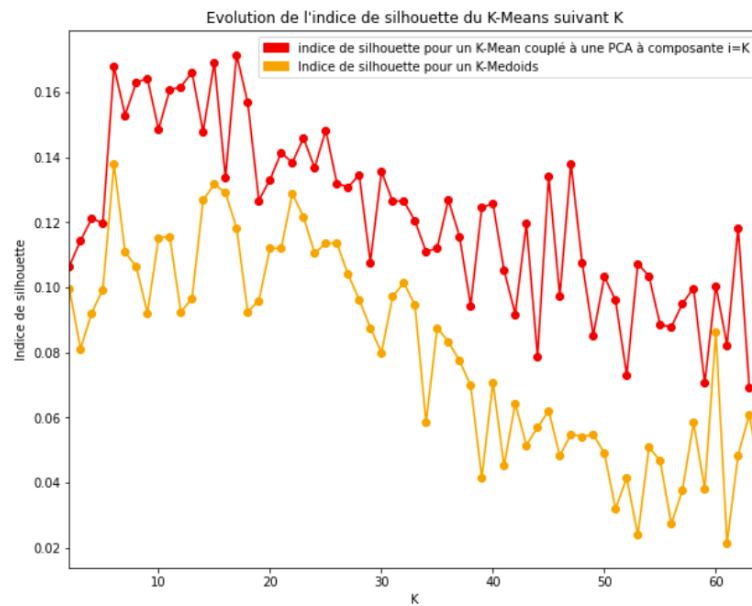


Figure 13 : Comparaison de l'indice de silhouette pour différentes valeurs de K pour le K-Means et le K-Medoid.

La figure 11 montre d'abord que l'évolution du temps de convergence est bien plus faible pour le K-Mean couplé au PCA, cela montre aussi une potentielle indépendance entre le nombre de clusters et la durée de l'algorithme pour le K-Medoid. L'inertie intra-classe demeure cependant toujours bien plus faible pour le K-Medoid ce qui n'induit pas nécessairement une meilleure

classification puisque la figure 13 montre que l'indice de silhouette demeure toujours meilleur pour le K-Means.

En conclusion, la classification non supervisée permet, dans le cas de la base MNIST, de stratifier l'écriture de l'ensemble des individus d'une manière optimale. Cela ne permet cependant pas d'atteindre des degrés de précision très importants quant à la prédiction du chiffre écrit, il n'existe donc pas de K optimal pour la précision. L'étude du K-Medoid et du K-Mean plus en profondeur nous amène donc à la conclusion que le K-Mean permet un clustering assez intéressant pour une valeur de K=12 qui ne maximise certes pas la précision mais qui maximise la distinction entre les différents chiffres écrits sur un plan informatique.

### **3. Classification supervisée des chiffres**

Ce n'est cependant pas le plan informatique qui nous intéresse ici, l'objectif est bien de maximiser la précision des prédictions. Étant donné que chacun est capable de discerner le chiffre écrit dans la base de données, les méthodes de classification supervisée semblent particulièrement adaptées. On envisage alors d'implémenter deux de ces méthodes : un Multi Layer Perceptron ainsi qu'un réseau de neurones profonds. Nous comparerons ensuite les performances de ces deux méthodes. Pour les deux méthodes employées nous utiliserons la base d'apprentissage proposée par le MNIST comme base d'apprentissage.

#### **3.1) Implémentation du Multi Layer Perceptron**

On implémente d'abord un Multi-Layer Perceptron, pour cela on réduit d'abord les images de 28x28 en un vecteur à 784 composantes. On fixe le nombre de nœuds de chaque couche caché à 32, la figure 14 offre un comparatif des performances pour des nombres différents de couches cachés. On fixe également une batch size de 128 pour l'entraînement, la fonction de perte "categorical cross entropy" ainsi que le SGD pour optimiser le processus. On fixe finalement le nombre d'epochs à 20 pour l'entraînement. L'architecture du modèle est la suivante : Une première couche Dense (fonction d'activation=sigmoid) + un nombre variable de couche Dense (fonction d'activation=sigmoid) + une couche softmax.

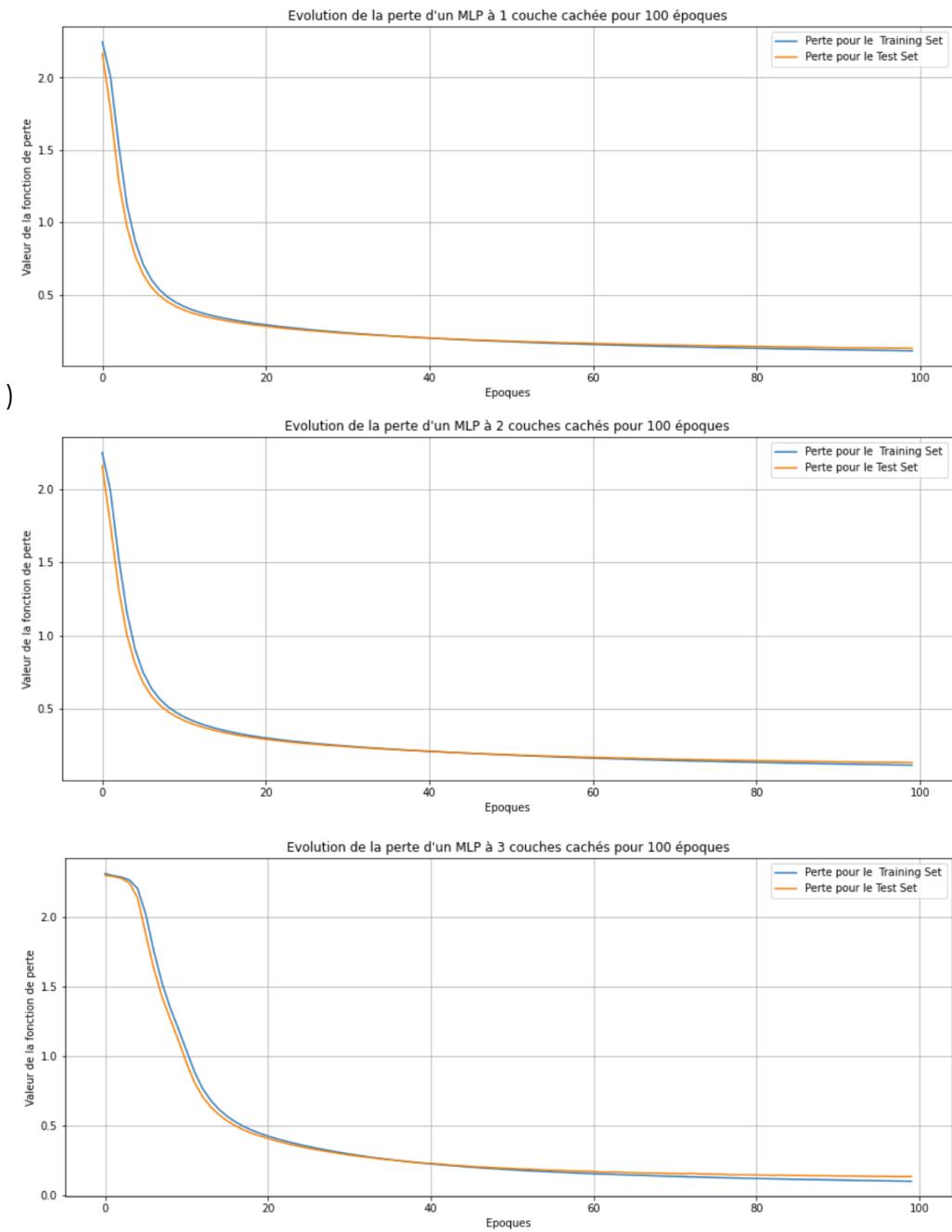


Figure 14 a) - Evolution de la fonction de perte pour un MLP pour un différent nombre de couches cachées.

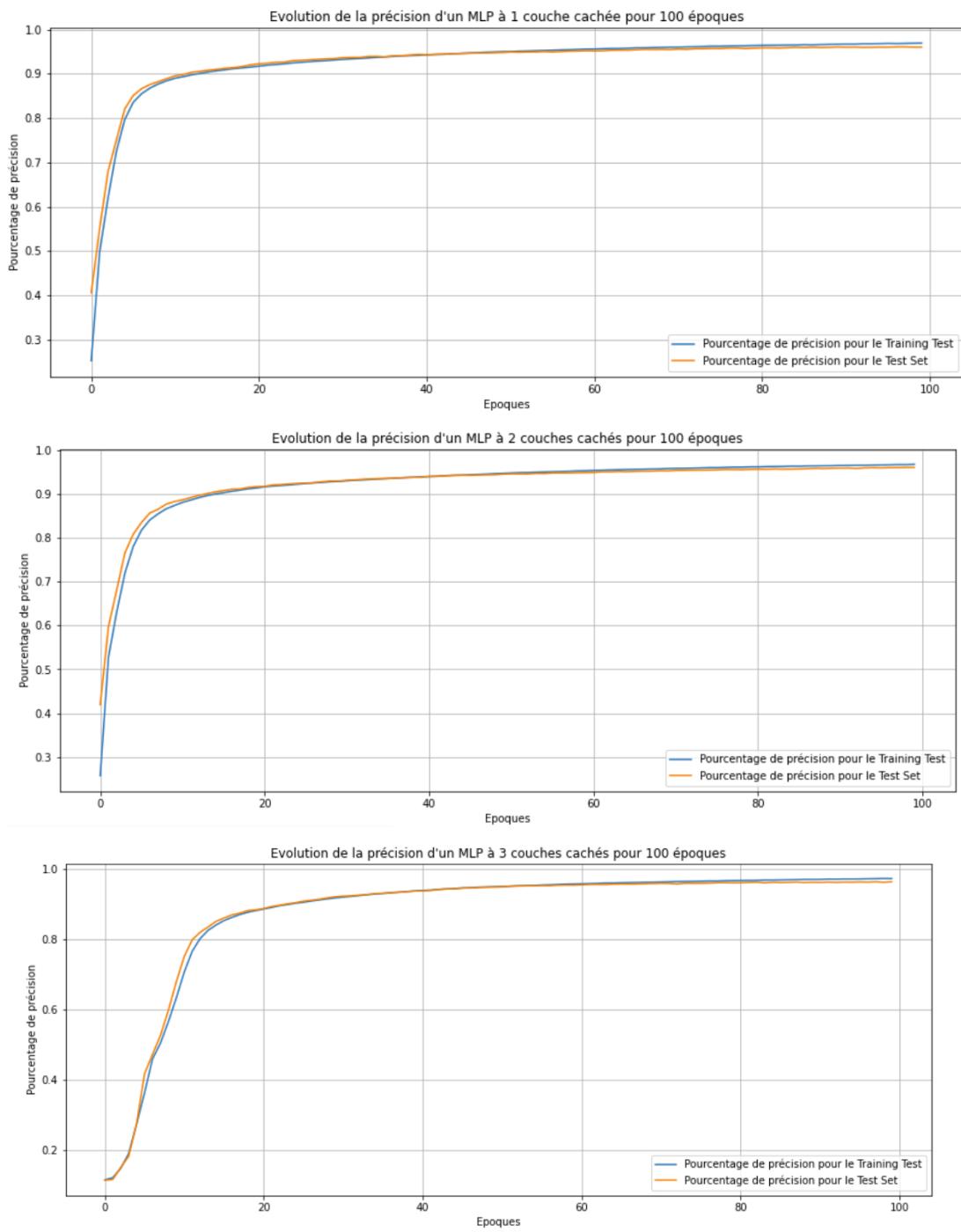


Figure 14 b) - Evolution de la précision pour un MLP pour un différent nombre de couches cachées.

On constate graphiquement que la fonction de perte diminue significativement pour les différents nombres de couches cachées et que la précision converge aux alentours de 97% pour tous les modèles à partir de 20 epochs. La convergence semble également s'effectuer plus rapidement pour une et deux couches cachées. Cependant, la diminution de la fonction de perte semble être plus lente lorsque le nombre de couches cachées

augmente. On note aussi que le modèle n'est pas sujet à l'over-fitting au vu de l'écart assez faible entre la précision du Test Set et du Training Set.

On effectue ensuite un K-Fold pour effectuer une validation croisée. On sépare ici le jeu de données en 5 par souci de complexité. On effectue celui-ci sur les 60 000 chiffres de la base d'entraînement. L'objectif est ici de vérifier si la précision trouvée n'évolue pas en changeant la portion de la dataset dédiée au test. (Figure 15)

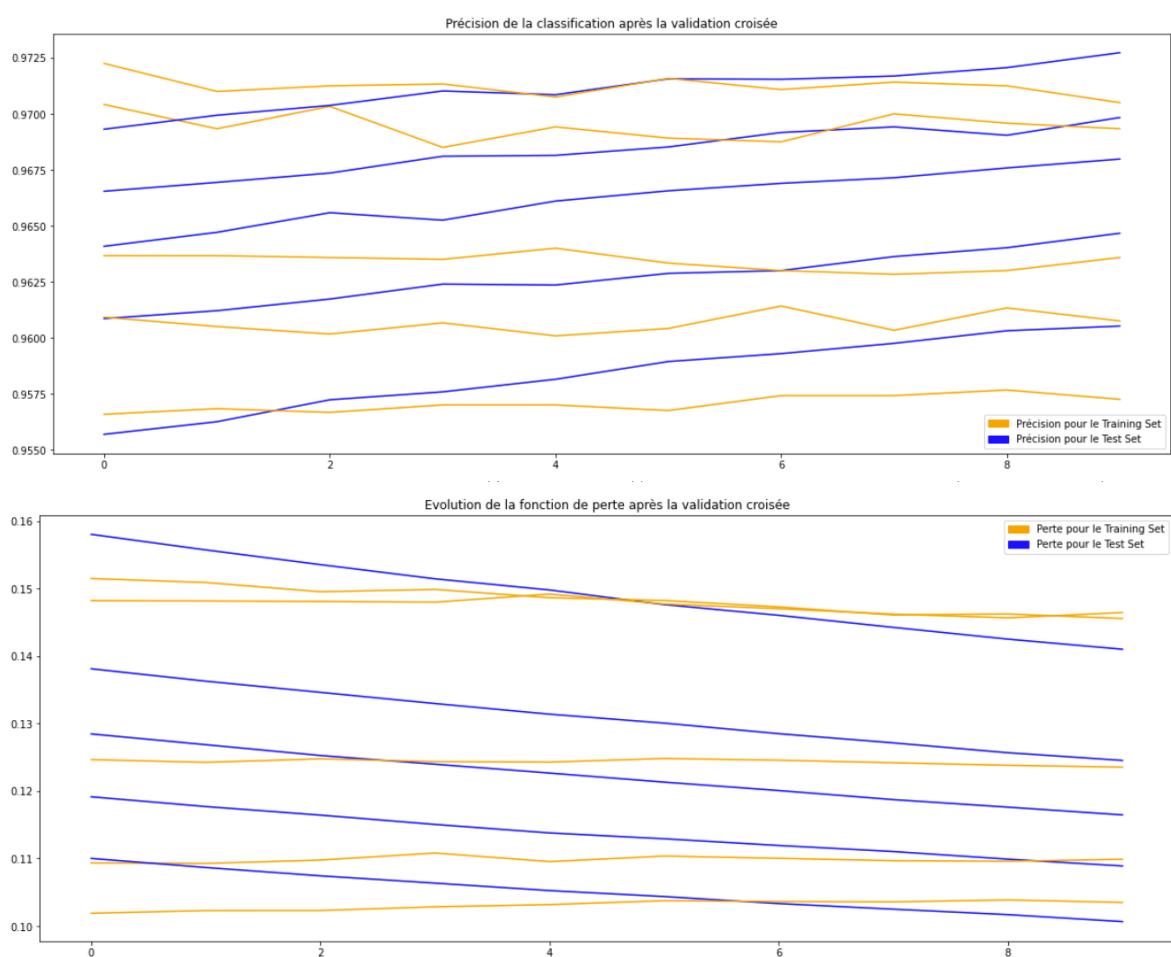


Figure 15 : Evolution de la précision et de la fonction de perte pour différentes séparations des données.

La Figure 15 nous indique donc que notre modèle n'est ni soumis à l'over-fitting ni à des biais quelconque de sélection des données pour l'apprentissage.

Le MLP est donc déjà bien plus adapté en termes de précisions que les autres méthodes de classification non supervisée. La précision s'approche ici de 97%, cette précision est visible sur la matrice de confusion en figure 16.

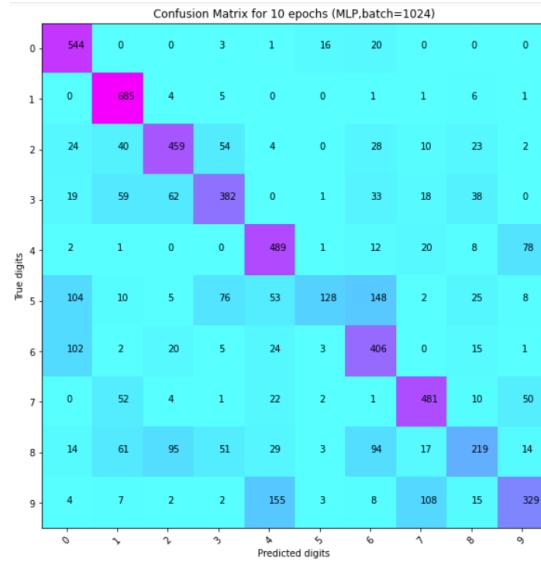


Figure 16 : Matrice de confusion pour 10 époques.

La matrice de confusion affichée ici étant valable pour 10 époques, la précision de détection est donc moindre. Cela est cependant intéressant pour observer les variations en termes de détection pour chaque chiffre. On constate par exemple que le 1 est le chiffre le mieux détecté tout comme pour les méthodes non supervisées. On constate aussi que le 1, le 2, le 6 et le 7 sont également bien détectés.

On observe également les erreurs les plus répandues à travers cette matrice. On constate ainsi que les 9 sont souvent pris pour des 4 et des 7 et que le 5 est le chiffre le moins bien détecté de tous. On note que le K-Mean ne crée pas de cluster de 5 pour 10 clusters et avait un taux d'erreur de 100% pour ce chiffre.

### **3.2) Implémentation du CNN**

L'implémentation du MLP nous amène donc à la conclusion que les méthodes de classification supervisée semblent être plus adéquates pour maximiser la précision. On cherche alors à trouver une méthode maximisant la précision de la détection des chiffres tout en maintenant une certaine rapidité dans l'exécution de ce modèle. On se penche alors sur une architecture CNN pour la construction du modèle. On opte pour une architecture classique (voir figure 17)

Layer (type)	Output Shape	Param #
conv2d_2 (Conv2D)	(None, 26, 26, 32)	320
max_pooling2d_2 (MaxPooling2 (None, 13, 13, 32)	0	
conv2d_3 (Conv2D)	(None, 11, 11, 64)	18496
max_pooling2d_3 (MaxPooling2 (None, 5, 5, 64)	0	
flatten_1 (Flatten)	(None, 1600)	0
dropout_1 (Dropout)	(None, 1600)	0
dense_4 (Dense)	(None, 10)	16010
Total params:	34,826	
Trainable params:	34,826	
Non-trainable params:	0	

Figure 17 : Architecture employée pour le CNN, la littérature montre que la succession de deux paires de convolution-subsamplings étaient suffisantes. On ajoute donc seulement une couche flatten, dropout et softmax pour finir.

On utilise ici un optimiseur Adam, une fonction de perte dite "categorical cross-entropy" avec un batch size d'entraînement toujours fixé à 128. Les performances sont illustrées sur la figure 18 :

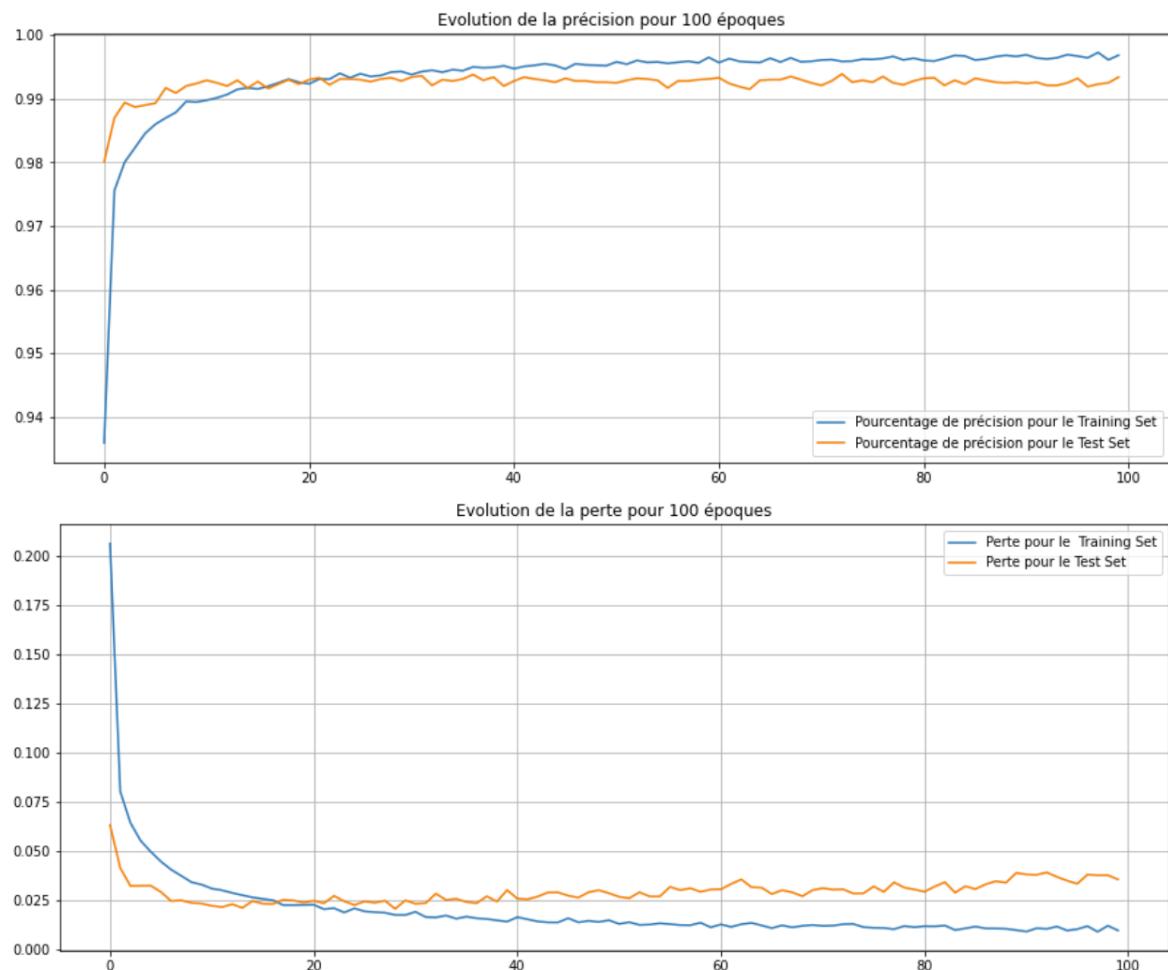


Figure 18 - Evolution de la précision et de la fonction de perte pour un CNN.

Le CNN possède ainsi une vitesse de convergence bien plus élevée que le MLP mais également une précision bien meilleure. Une optimisation des hyperparamètres pourrait sans doute encore améliorer ces performances mais celles-ci sont déjà plus que satisfaisantes si on les compare à celles des autres méthodes employées dans ce rapport.

La mise en place du K-Fold donne de plus le graphique suivant. (voir Figure 19)

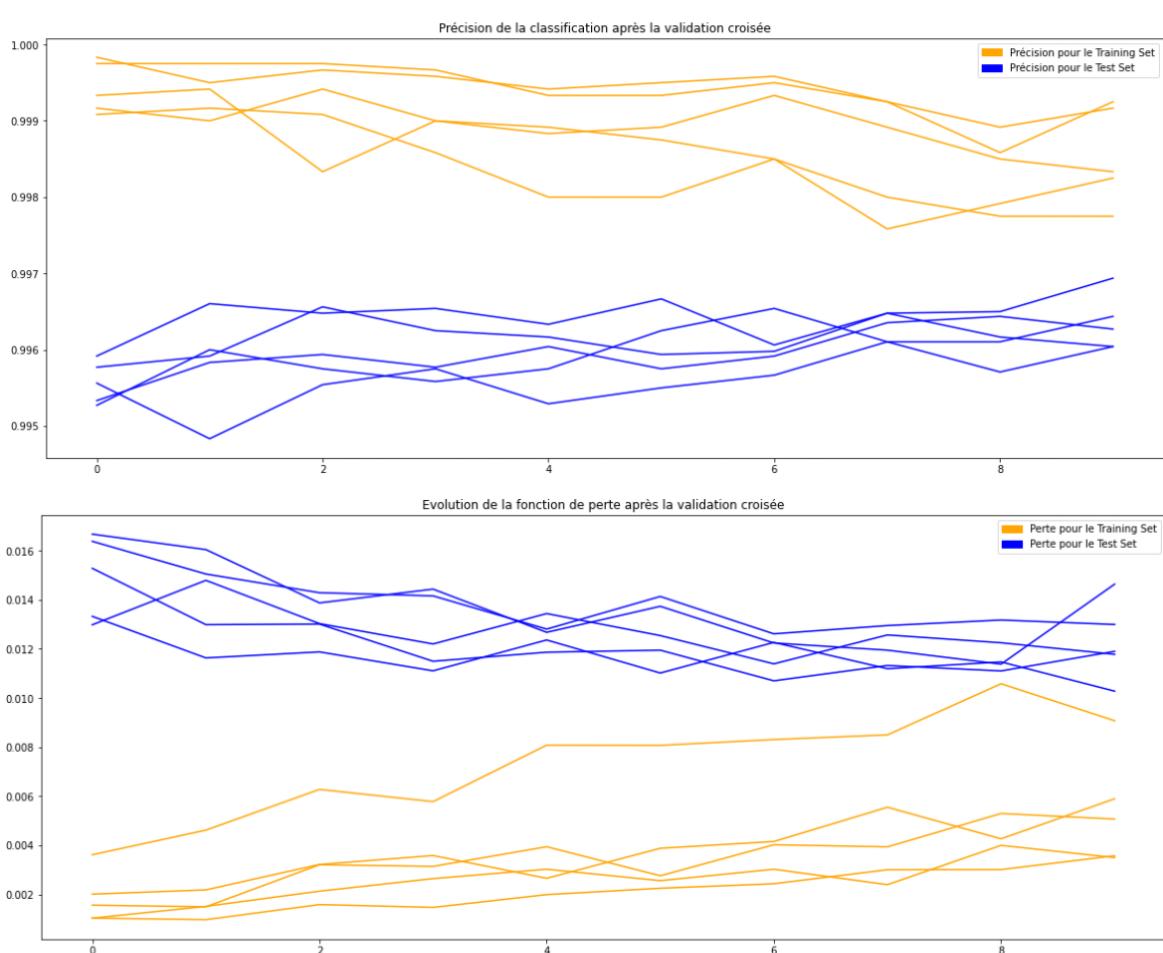


Figure 19 : Évolution de la précision et de la fonction de perte pour différentes séparations des données.

Le K Fold nous permet de constater un très léger overfitting du modèle. Cela n'est cependant pas problématique puisque la précision du modèle est plus faible de 3e-3 lorsque l'on applique le modèle à la base de test. Cet overfitting est également visible pour la fonction de perte. Cette perte de précision reste cependant très raisonnable puisque la précision du fold le moins précis est déjà bien plus importante que celle pour un MLP.

Cette précision est notamment visible à travers la matrice de confusion. (figure 20)

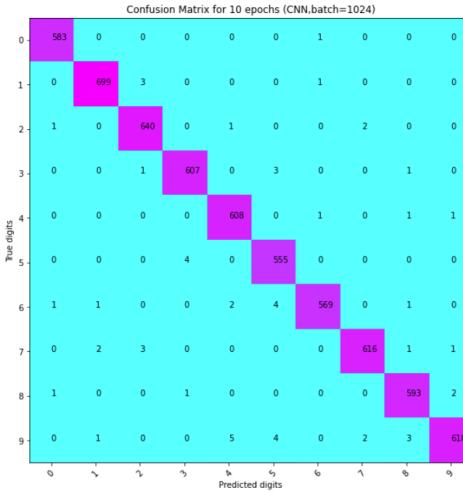


Figure 20 : Matrice de confusion pour 10 époques .

La comparaison de la figure 20 et de la figure 16 nous permet donc de conclure quant à l'adéquation du CNN avec la base de données MNIST. Conformément à la figure 15, la précision atteinte pour 10 epochs atteint 99%, chaque chiffre est ainsi particulièrement bien détecté pour l'architecture employée.

#### 4. Comparaison des méthodes et conclusion générale

Les trois méthodes diffèrent donc totalement en termes de performances. Les méthodes de classification non supervisées sont très peu efficaces en ce qui concerne la précision, ces dernières sont en effet capables de performances correctes (environ 80% de détection réussie) mais pour un nombre K trop importants pour que la complexité temporelle de l'algorithme n'en pâtit pas. Ces méthodes sont cependant adéquates pour générer des catégories spécifiques d'individus. En ce qui concerne les performances en termes de précision, les méthodes de classification supervisée sont bien plus efficaces. Le MLP possède par exemple une précision atteignant 97% pour un temps de calcul assez faible par rapport à l'exécution d'un K-Means avec un K très élevé. Le CNN relève quant à lui de la meilleure option pour optimiser la précision. En effet, bien que celui-ci soit beaucoup plus long à exécuter, celui-ci converge très rapidement et donne des résultats proches de 99,9%. Cette méthode est donc la plus adaptée, ce qui n'est pas surprenant puisque le CNN est une des méthodes les plus utilisées en traitement d'images. Le MLP demeure une bonne option pour détecter les chiffres par sa rapidité d'exécution et sa précision plus que correcte.