Rapport : Projet Digit Recognizer (Partie 2)

I. Introduction

Il s'agit de la seconde partie du projet sur la reconnaissance de chiffres manuscrits à l'aide de modèles prédictifs.
Pour rappel, ce projet, proposé par *Kaggle* (lien vers la compétition), vise à prédire correctement des chiffres manuscrits représentés dans des images.
A partir de données fournies, nous devons entrainer un modèle prédictif et l'utiliser pour prédire les valeurs de données sans labels également fournies.

Le premier rapport se concentrait sur l'utilisation d'un **SVM** (Support Vector Machine) et d'un **PCA** (Principal Component Analysis) pour réduire les dimensions des données et améliorer les performances du modèle.

Dans cette seconde partie, nous abordons un autre outil fondamental en apprentissage automatique : **les réseaux de neurones**. En particulier, nous utiliserons un **MLPClassifier** (Multilayer Perceptron Classifier) de la bibliothèque `sklearn` pour entraîner et évaluer un modèle capable de reconnaître ces chiffres manuscrits.

Objectifs du Projet

Le projet repose sur les mêmes objectifs que le précédent, à savoir :

- **Analyser les données** fournies pour comprendre leur structure et leur organisation.
- **Concevoir et entraîner un modèle** prédictif, cette fois-ci basé sur un réseau de neurones multicouches (MLP).
- **Évaluer les performances** du modèle à l'aide de métriques comme la précision et la matrice de confusion.
- **Optimiser le modèle** en testant différentes architectures (nombre de couches et de neurones par couche).

Enfin, nous utiliserons le modèle entraîné pour prédire la valeur d'échantillons sans label dans les données de test.

Présentation des données

Pour rappel, les données fournies sont issues de la base MNIST, un ensemble de référence pour la reconnaissance de chiffres manuscrits. Chaque exemple est une image en niveaux de gris de dimensions 28x28 pixels, représentant un chiffre entre 0 et 9. Les valeurs des pixels vont de 0 (blanc) à 255 (noir).

Le fichier `train.csv` contient 42 000 exemples, avec une colonne label indiquant le chiffre représenté et 784 colonnes correspondant aux valeurs des pixels.
Le fichier `test.csv`, quant à lui, comprend 28 000 exemples non étiquetés, organisés de manière similaire, mais sans la colonne label.

L'objectif de ce projet est donc de donner un label pour chaque image du fichier `test.csv`

II. Réseau de Neurone

Pour rappel, un réseau de neurones est un modèle d'apprentissage inspiré de la structure et du fonctionnement des neurones biologiques.
Il est composé de couches successives de neurones, où chaque neurone prend des entrées, applique une transformation (appelé fonction d'activation), puis transmet un résultat pondéré aux couches suivantes.

Pour ce projet, nous présenterons en entrée du réseau les luminances de chaque pixel de l'image, et voudrons en sortie la valeur du chiffre représenté sur l'image (de 0 à 9).

Les Hyperparameters choisi

Le modèle a été configuré avec les paramètres suivants :

```
mlp = MLPClassifier(
    hidden_layer_sizes=hidden_layer_sizes,
    activation='relu',
    solver='adam',
    max_iter=max_iter,
    random_state=random_state,
    verbose=True
)
```

- *hidden_layer_sizes* : Définit le nombre de couches cachées et le nombre de neurones par couche.
- *activation='relu'* : Utilise la fonction d'activation ReLU, un choix courant pour des réseaux profonds car elle évite le problème du gradient qui disparaît.
- *solver='adam'* : Spécifie l'algorithme de mise à jour des poids, ici Adam.
- *max_iter* : Limite le nombre d'itérations pour éviter les calculs excessifs ou la divergence.
- *random_state* : Assure la reproductibilité des résultats en initialisant les poids aléatoirement de manière fixe.

Nous n'avons pas utilisé de mini-batchs (par défaut dans MLPClassifier, le batch complet est utilisé) car nous n'avons pas noté de différence notable entre l'utilisation ou non de cette technique.

AdAM

L'optimiseur **AdAM** (pour Adaptive Moment Estimation) est utilisé pour mettre à jour les poids du réseau.
Il combine les avantages de deux techniques :

- *RMSProp*, qui adapte le taux d'apprentissage pour chaque paramètre en fonction de sa variance.
- *Moment*, qui accélère la descente de gradient en tenant compte des mises à jour

précédentes.

AdAM est particulièrement efficace pour des problèmes complexes et des réseaux profonds, car il nécessite peu de réglages manuels et converge rapidement.

Pour ce projet, nous avons choisi d'utiliser cette technique car elle ne nécessite pas de régler finement les hyperparameters.

ReLU

La fonction d'activation **ReLU** (Rectified Linear Unit) applique une transformation simple :
[
f(x) = \max(0, x)
]
Elle présente deux avantages principaux :

1. *Simplicité computationnelle* : ReLU est rapide à calculer par rapport à d'autres fonctions comme la sigmoïde ou tanh.
2. *Évite le problème du gradient qui disparaît* : Contrairement à la sigmoïde, elle maintient un gradient non nul pour des valeurs positives, favorisant un apprentissage efficace dans les couches profondes.

Les Architectures Choisies et Justifications

Nous avons testé plusieurs architectures pour déterminer la meilleure configuration.

- Nous avons choisi plusieurs réseaux que l'on a considérés comme étant insuffisamment complexes : *(10)*, *(16, 10)* et *(32, 16, 10)*.
  Ces réseaux sont très simples et ne possèdent pas la capacité d'apprendre des relations complexes. Nous les avons choisis pour observer les performances de modèles minimalistes et comprendre comment la profondeur et la taille des couches influencent les résultats.

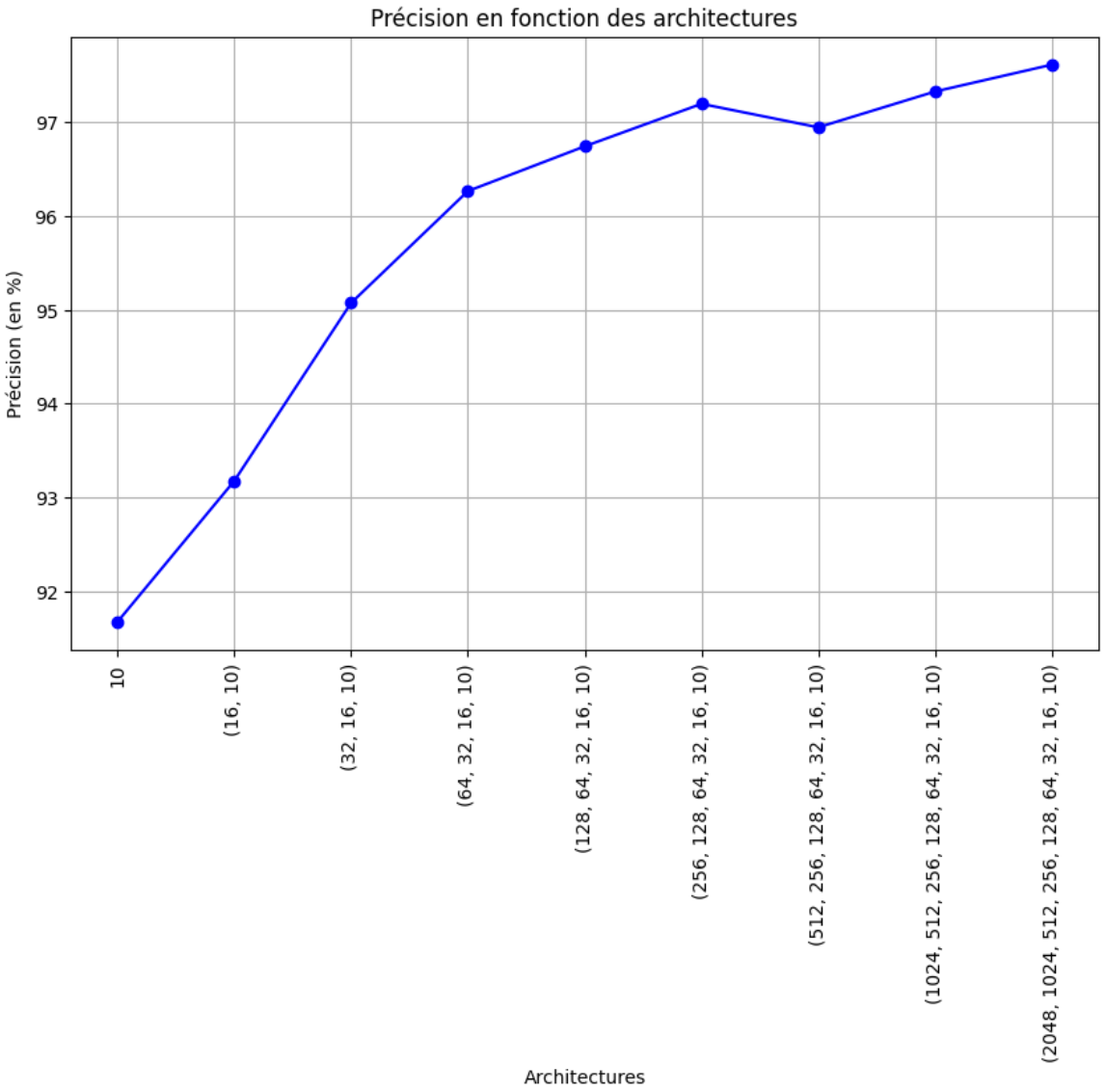
- Ensuite, nous avons programmé des réseaux avec des architectures modérément complexes : *(64, 32, 16, 10)*, *(128, 64, 32, 16, 10)* et *(256, 128, 64, 32, 16, 10)*.
  Ces réseaux, plus profonds, permettent une meilleure extraction des caractéristiques, notamment pour les relations significatives entre les pixels dans des images de chiffres manuscrits. Ces architectures sont adaptées pour capturer des relations plus complexes sans nécessiter des ressources excessives.
  Nous les avons choisis pour explorer le compromis entre capacité d'apprentissage et efficacité.

- Enfin, nous avons testé des architectures complexes et très profondes : *(512, 256, 128, 64, 32, 16, 10)*, *(1024, 512, 256, 128, 64, 32, 16, 10)*, et *(2048, 1024, 512, 256, 128, 64, 32, 16, 10)*.

Ces réseaux de grande envergure sont conçus pour des tâches exigeantes, exploitant pleinement les données complexes. Leur profondeur et leur largeur permettent d'apprendre des relations abstraites et de combiner des informations à plusieurs niveaux.
Cependant, ils présentent un risque de surajustement si les données ou les mécanismes de régularisation sont insuffisants.
Nous avons inclus ces architectures pour tester leurs limites et observer si elles apportent des gains significatifs par rapport à des architectures plus modestes.

III. Résultats

1. Evolution de la Précision selon l'Architecture.



Sur ce graphique, nous observons l'évolution de la précision en fonction de l'architecture choisie pour le réseau de neurones.

Globalement, plus le réseau est complexe, plus la précision augmente. Cependant, le gain de précision diminue progressivement à mesure que la complexité du réseau s'accroît. Par exemple, un réseau avec une seule couche cachée atteint une précision inférieure à 92 %, tandis qu'un réseau comportant 4 couches cachées dépasse les 96,5 %.
Par la suite, le gain stagne : un réseau avec 6 couches atteint environ 97,2 %, tandis qu'un réseau avec 9 couches n'améliore la précision que légèrement, atteignant un peu plus de 97,5 %.

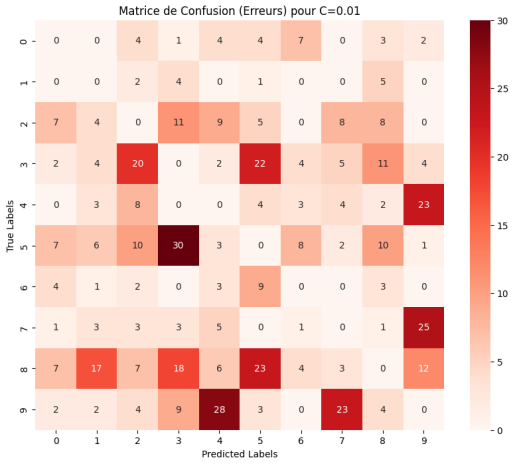
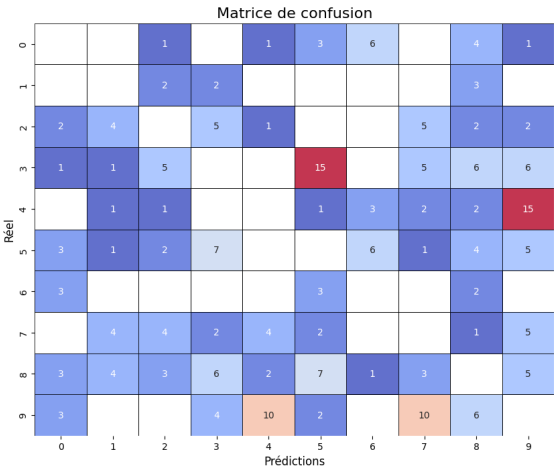
Cependant, cette amélioration marginale de la précision se fait au prix d'un temps de calcul beaucoup plus important. Les temps de construction des modèles augmentent de manière exponentielle avec la complexité du réseau.

2. Comparaison avec une SVM

Dans cette partie du rapport, nous comparons les performances du réseau de neurones en comparaison avec celles obtenues dans le premier rapport, où nous avions utilisé une **SVM** (Support Vector Machine) combinée à un **PCA** (Principal Component Analysis).

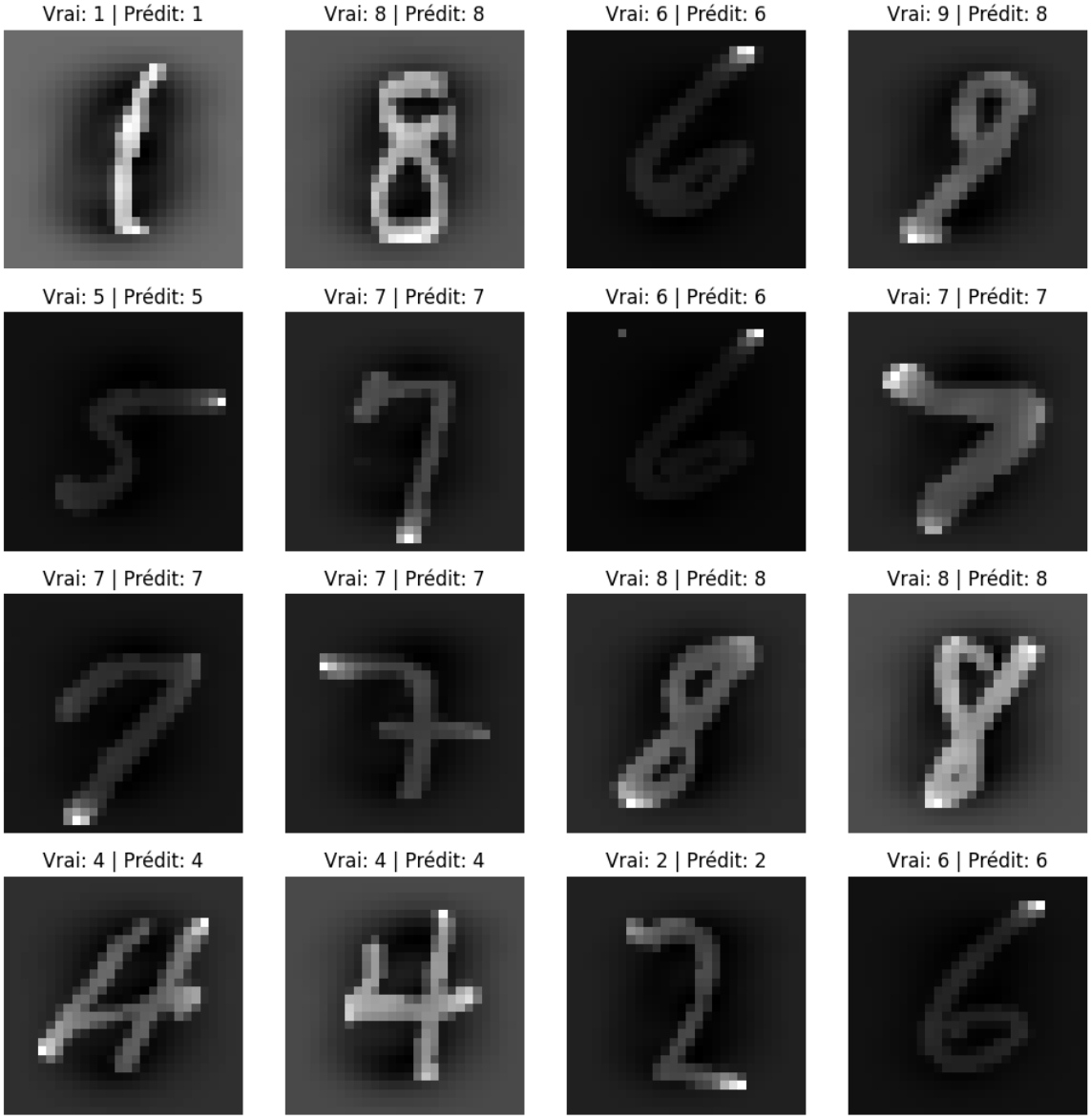
Dans le premier rapport, l'objectif était de prédire les chiffres manuscrits à partir des mêmes données, en utilisant une SVM comme algorithme de classification. Étant donné la grande dimensionnalité des images (784 dimensions correspondant aux pixels), un PCA a été utilisé en amont pour réduire la dimensionnalité.
Les performances de cette approche ont culminé à une précision maximale de **92 %**, après un ajustement des hyperparamètres de la SVM et une optimisation de la réduction dimensionnelle.

En utilisant un réseau de neurones, nous avons dépassé ces résultats. La meilleure architecture testée a atteint une précision de **97,6 %**, soit une augmentation de 5,6 % par rapport à la méthode basée sur SVM.

PCA & SVM	**Réseau de Neurones**
Précision : 92.6%	Précision : 97.2%
	

En comparant les deux matrices de confusion, nous pouvons voir que les deux modèles ont des difficultés pour différencier les même chiffres : 4<->9, 7->9 et 5<->3.
Cependant, les performances du réseau est meilleure, notamment pour la reconnaissance du 8 et du 3.

Conclusion



Le réseau de neurones s'est révélé plus performant pour ce problème, grâce à sa capacité à capturer des relations complexes entre les pixels des images sans nécessiter de transformation préalable des données. De plus, le réseau peut être ajusté à travers une large gamme d'architectures pour s'adapter aux besoins spécifiques, que ce soit en termes de précision ou de temps d'entraînement.

Parmi les différentes architectures de réseau de neurones testées, nous avons choisi de retenir l'architecture **(256, 128, 64, 32, 16, 10)**, qui a atteint une précision de **97,2 %**.
Lorsque nous comparons ces résultats à ceux obtenus dans notre premier rapport, où une approche basée sur une SVM (Support Vector Machine) combinée à un PCA (Principal Component Analysis) avait été utilisée, nous pouvons observer une amélioration significative des performances.
En effet, la SVM avait atteint une précision de **92,5 %**, soit une augmentation de **4,7 %** avec le réseau de neurones.

Ce projet ouvre la voie à de nombreuses optimisations et extensions possibles. Par exemple :

- Explorer d'autres fonctions d'activation ou optimisateurs pour améliorer la vitesse d'entraînement.
- Étendre le modèle à des données plus complexes, comme des images de dimensions supérieures ou en couleurs.
- Comparer les performances du réseau de neurones avec d'autres modèles avancés comme les **CNN (Convolutional Neural Networks)**, spécifiquement conçus pour les tâches de vision par ordinateur.

Ainsi, ce projet met en lumière l'efficacité des réseaux de neurones pour la reconnaissance d'images tout en soulignant l'importance du choix de l'architecture et des hyperparamètres pour obtenir un modèle performant.