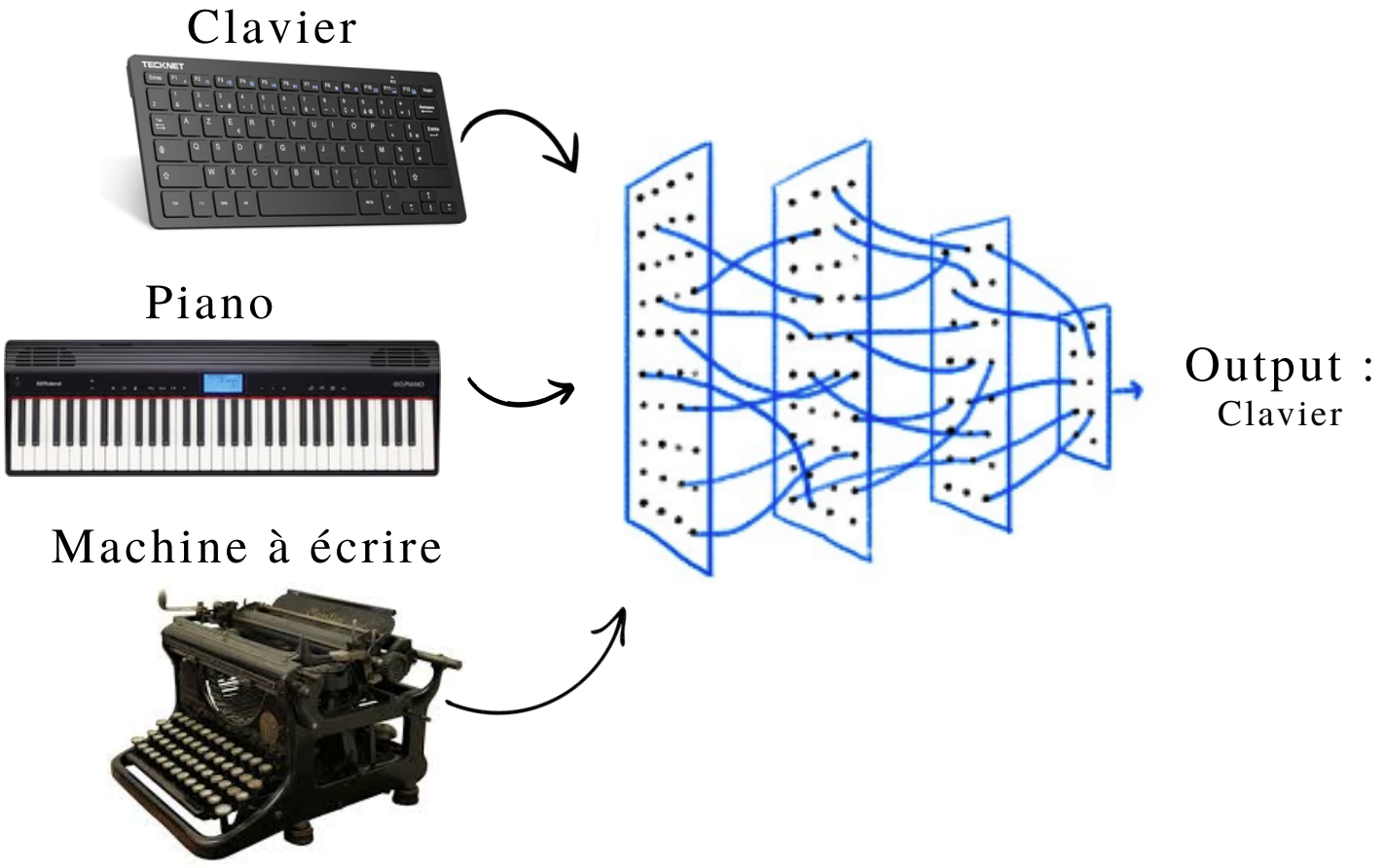
Bachelor 3 Intelligence artificielle et big data

RAPPORT DU PROJET ANNUEL

Nabil SARKER

Classification des images à l'aide d'un perceptron multi couches



SEPTEMBRE 2024

SOMMAIRE

1 - INTRODUCTION

Le perceptron multicouche, souvent abrégé PMC, est l'un des algorithmes de base des réseaux de neurones artificiels. Il est à la base de nombreux modèles d'apprentissage automatique, notamment les réseaux profonds utilisés pour résoudre des problèmes de classification, de reconnaissance de formes, et bien d'autres tâches complexes. Un perceptron multicouche est caractérisé par l'ajout de couches intermédiaires (couches cachées) entre les neurones d'entrée et de sortie, permettant au modèle de capturer des relations non linéaires et d'augmenter sa capacité d'apprentissage.

Le projet que j'ai entrepris consiste à implémenter un perceptron multicouche (PMC) pour classer des données issues de trois catégories distinctes : Piano, Clavier, et Machine à écrire. Ce rapport présente l'ensemble des étapes réalisées, de la constitution du jeu de données au prétraitement, en passant par l'implémentation du PMC, les différentes expérimentations sur les hyperparamètres, jusqu'à l'analyse détaillée des résultats obtenus.

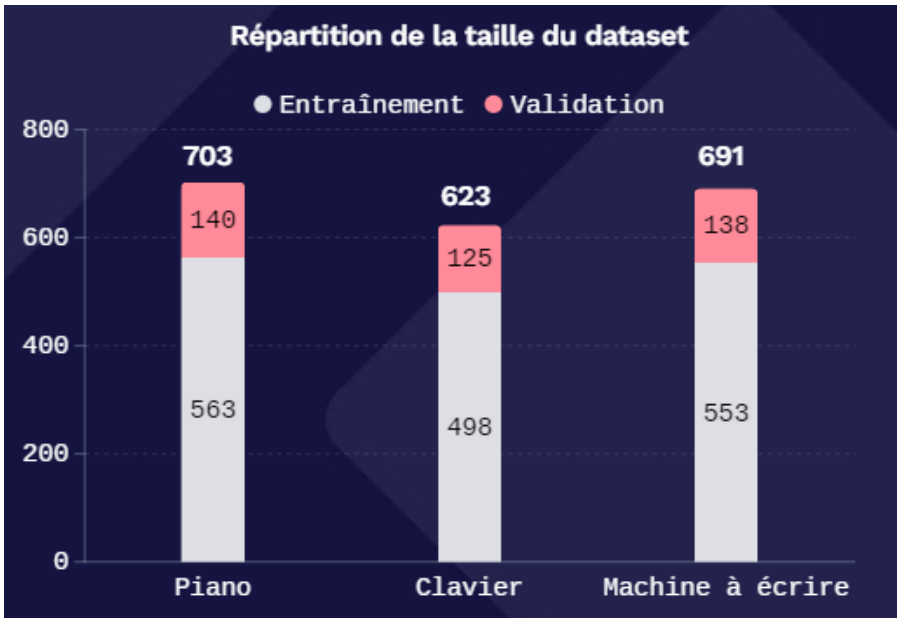
2 - CONSTITUTION DE DONNÉES

La première étape de ce projet a été la constitution du jeu de données. Celui-ci a été recueilli à l'aide de techniques de web scraping via l'extension Google Chrome "Visual Web Scraper". Cette méthode m'a permis d'extraire efficacement des images pertinentes sur les catégories Piano, Clavier et Machine à écrire à partir de différentes sources en ligne.

J'ai ensuite divisé les données en deux sous-ensembles : un ensemble d'entraînement et un ensemble de validation, afin de permettre une évaluation juste des performances du modèle. Les proportions des données d'entraînement et de validation sont listées ci-dessous :

- Piano : 563 données d'entraînement et 140 données de validation.
- Clavier : 498 données d'entraînement et 125 données de validation.
- Machine à écrire : 553 données d'entraînement et 138 données de validation.

Ces données sont bien équilibrées, ce qui va me permettre d'entraîner un modèle robuste sans biais majeur dans les catégories.



3 - PRÉTRAITEMENT DES DONNÉES

Le prétraitement est une étape cruciale avant d'alimenter les données dans le perceptron multicouche. L'absence de prétraitement adéquat peut entraîner de mauvaises performances du modèle, voire une incapacité à apprendre efficacement. Pour ce projet, plusieurs étapes de prétraitement ont été réalisées.

- Nettoyage des données : j’ai éliminé toutes les images anormales qui auraient pu fausser les résultats. Cette étape assure que le modèle reçoive uniquement des données (images) valides et exploitables.
- Normalisation des données : les données brutes, telles que recueillies, n'étaient pas sur une échelle uniforme. Par exemple, les valeurs des pixels dans une image peuvent varier de 0 à 255, tandis qu'une autre image variait entre 0 et 70. Cela pouvait déséquilibrer l'apprentissage du modèle, car des variables avec des amplitudes plus grandes auraient influencé davantage l'apprentissage. Ainsi, j’ai normalisé les données afin que toutes les variables aient une valeur comprise entre 0 et 1 et soient compatibles avec la fonction d’activation.

Ce prétraitement a permis de fournir des données cohérentes et équilibrées au modèle, facilitant ainsi l'entraînement et l'obtention de résultats optimaux.

4 - STRUCTURE DU PERCEPTRON MULTICOUCHE

Le perceptron multicouche est une extension du perceptron simple. Tandis que ce dernier ne possède qu'une seule couche de neurones (l'entrée étant directement connectée à la sortie), le perceptron multicouche ajoute une ou plusieurs couches cachées. Chaque neurone d'une couche est connecté à tous les neurones de la couche suivante, chaque connexion étant pondérée. Ces poids sont ajustés au cours de l'entraînement pour minimiser l'erreur entre la sortie prédite par le modèle et la valeur attendue.

Dans ce projet, plusieurs configurations de PMC ont été testées afin de trouver la meilleure structure possible. La version finale du PMC se compose de la manière suivante :

- Couche d'entrée : cette couche reçoit les caractéristiques du dataset. Elle contient un nombre de neurones égal au nombre de variables d'entrée (les caractéristiques des objets à classer).
- Couches cachées : j'ai expérimenté différentes configurations pour les couches cachées. Le modèle final contient une couche cachée avec 100 neurones dans la configuration B, qui a donné les meilleurs résultats. J'ai également testé un modèle plus complexe avec 3 couches cachées et 150 neurones, mais ce modèle n'a pas surpassé la version plus simple.
- Couche de sortie : la couche de sortie contient 3 neurones, car nous avons 3 classes à prédire (Piano, Clavier, Machine à écrire). Une fonction d'activation Sigmoïd est appliquée sur la sortie pour obtenir des probabilités de chaque classe.

5 - FONCTION D'ACTIVATION

La fonction d'activation utilisée dans les couches cachées de ce perceptron multicouche est la fonction sigmoïde. La fonction sigmoïde est largement utilisée dans les réseaux de neurones car elle a la propriété de comprimer les valeurs d'entrée, quelles qu'elles soient, dans un intervalle compris entre 0 et 1. Cela permet au réseau de modéliser des probabilités et de gérer les non-linéarités dans les données.

Voici la formule mathématique utilisée :

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

6 - ALGORITHME D'APPRENTISSAGE

L'algorithme d'apprentissage utilisé pour l'entraînement du modèle est la rétropropagation avec la méthode de stochastique gradient descente (SGD). La rétropropagation permet de calculer l'erreur pour chaque poids du réseau et d'ajuster ces poids en conséquence pour minimiser l'erreur totale du modèle.

```
def train(self, input_data, target, epochs=1000):
    self.losses = []
    for epoch in range(epochs):
        epoch_loss = 0
        for x, y in zip(input_data, target):
            x = x.reshape(1, -1)
            y = y.reshape(1, -1)
            hidden_output, output = self.forward(x)
            self.backward(x, y, hidden_output, output)
            epoch_loss += np.mean(np.square(y - output))
        self.losses.append(epoch_loss / len(input_data))
        #if epoch % 100 == 0:
        print(f"Epoch {epoch}, Loss: {self.losses[-1]}")
```

7 - EXPÉRIMENTATIONS ET RÉGLAGE DES HYPERPARAMÈTRES

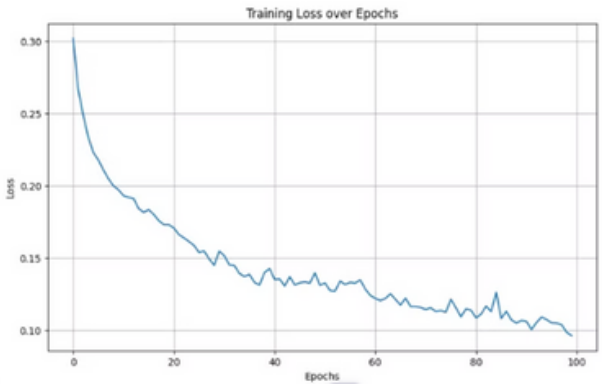
L'une des étapes les plus importantes dans l'implémentation d'un modèle de perceptron multicouche est le réglage des hyperparamètres. Ceux-ci incluent le taux d'apprentissage, le nombre d'époch, le nombre de neurones dans chaque couche cachée, ainsi que le nombre de couches cachées. J'ai mené plusieurs expérimentations en ajustant ces paramètres afin de trouver la meilleure configuration possible.

8 - EXPÉRIMENTATIONS RÉALISÉES

Quatre modèles principaux ont été testés avec différentes configurations d'hyperparamètres. Les résultats sont résumés dans le tableau ci-dessous :

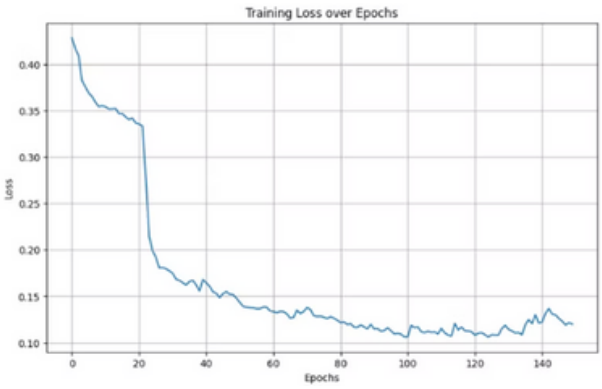
Modèle	Nb épochs	Learning rate	Nb neurones	Nb couches cachées	Accuracy	Temps
A	100	0.1	100	1	46%	9min
B	150	0.01	100	1	49%	11min
C	500	0.01	100	1	42%	38min
D	280	0.001	150	3	44%	1h25min

Je constate que le modèle B, avec 150 épochs, un taux d'apprentissage de 0,01 et 100 neurones dans une couche cachée, a obtenu la meilleure précision (49 %). Il présente également un bon équilibre entre précision et temps d'entraînement (11 minutes).



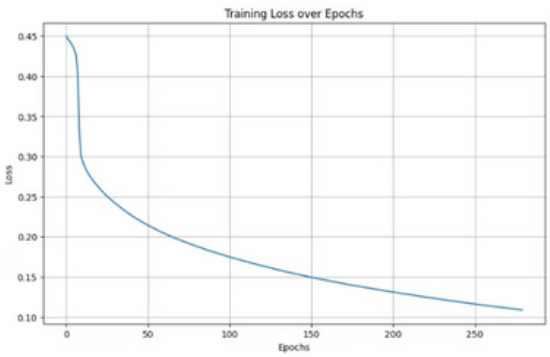
MODÈLE A

Nb épochs : 100
Learning rate : 0.1
Nb neurones : 100
Temps : 9 min



MODÈLE B

Nb épochs : 150
Learning rate : 0.01
Nb neurones : 100
Temps : 11 min



MODÈLE C

Nb époch : 500
Learning rate : 0.01
Nb neurones : 100
Temps : 38 min

MODÈLE D

Nb époch : 280
Learning rate : 0.001
Nb neurones : 150
Nb couches cachées : 3
Temps : 1h25min

9 - ANALYSE DES RÉSULTATS DES EXPÉRIMENTATIONS

Les résultats des expérimentations montrent que le choix des hyperparamètres a un impact significatif sur les performances du modèle. Plusieurs leçons peuvent être tirées de ces tests.

Le modèle A, bien qu'entraîné rapidement, n'a pas atteint une bonne précision (46 %). Le modèle C, avec un plus grand nombre d'époch, a souffert de surentraînement, ce qui a conduit à une diminution de la précision (42 %), malgré un temps d'entraînement plus long.
Le modèle D, bien qu'ayant ajouté deux couches cachées supplémentaires et 50 neurones de plus, n'a pas surpassé le modèle B. En fait, il a montré une précision inférieure (44 %) et un temps d'entraînement beaucoup plus long (1 heure 25 minutes). Cela montre que, dans certains cas, une augmentation de la complexité du modèle ne conduit pas nécessairement à de meilleures performances.

10 - IMPORTANCE DU TAUX D'APPRENTISSAGE

Le taux d'apprentissage est un facteur clé qui influence la vitesse de convergence du modèle et son aptitude à bien généraliser. Un taux trop élevé peut entraîner des oscillations ou empêcher la convergence, tandis qu'un taux trop faible peut ralentir le processus d'apprentissage ou mener à un sous-apprentissage.

Dans mon expérimentation, le modèle B, qui a un taux d'apprentissage de 0,01, a montré de meilleures performances avec une accuracy de 49 %. À l'inverse, le modèle A, avec un taux d'apprentissage de 0,1, a atteint un taux d'accuracy de 46 %, mais a convergé plus rapidement. Cela montre que, bien que des taux d'apprentissage plus élevés permettent d'obtenir des résultats rapidement, ils ne garantissent pas la meilleure performance possible.

11 - NOMBRE D'ÉPOCHS

Le nombre d'époch représente le nombre de fois que l'algorithme d'apprentissage parcourt l'ensemble des données d'entraînement. Une augmentation du nombre d'époch peut permettre d'améliorer la précision du modèle, mais elle risque également de provoquer un surapprentissage (overfitting), où le modèle devient trop spécifique aux données d'entraînement et perd sa capacité à généraliser sur de nouvelles données.

Dans ce projet, le modèle C, entraîné avec 500 époch, n'a pas obtenu de bonnes performances avec une accuracy de seulement 42 %. Ce résultat pourrait indiquer un phénomène de surapprentissage, où le modèle a mémorisé les données d'entraînement au lieu de généraliser correctement. Ainsi, il est essentiel de trouver un juste milieu entre un nombre d'époch suffisant pour que le modèle apprenne correctement et un nombre excessif qui entraînerait un surapprentissage.

12 - COMPLEXITÉ DU MODÈLE

J'ai également expérimenté avec des configurations plus complexes, notamment avec le modèle D, qui comporte trois couches cachées et 150 neurones dans chaque couche. Malgré cette complexité accrue, le modèle n'a pas surpassé le modèle B en termes de précision, et son temps d'entraînement a considérablement augmenté. Cela montre que l'ajout de couches cachées supplémentaires ou de neurones ne conduit pas nécessairement à une amélioration des performances. En fait, cela peut parfois entraîner une dégradation des performances en raison de l'augmentation de la complexité du modèle et des risques de surajustement.

13 - TEMPS D'ENTRAÎNEMENT

Le temps d'entraînement est un facteur critique, surtout lorsque l'on travaille avec des ensembles de données volumineux ou des structures de modèles complexes. Le modèle B a non seulement obtenu la meilleure précision, mais il a également maintenu un temps d'entraînement relativement court de 11 minutes. En comparaison, le modèle D a pris 1 heure et 25 minutes pour s'entraîner, sans pour autant offrir des performances supérieures. Cela montre l'importance d'équilibrer les performances et l'efficacité lors de la conception des structures de réseaux de neurones.

14 - COMPARAISON AVEC D'AUTRES MÉTHODES

Bien que le perceptron multicouche soit une méthode efficace pour les tâches de classification, il existe d'autres structures de réseaux de neurones qui peuvent offrir de meilleures performances pour certaines tâches spécifiques. Voici une alternative courante :

- Réseaux Neuronaux Convolutifs (CNN) : Les CNN sont souvent utilisés pour les tâches liées aux images ou aux séries temporelles. Ils sont particulièrement efficaces pour capturer les relations spatiales ou temporelles dans les données grâce à l'utilisation de convolutions qui extraient des caractéristiques locales.

15 - CONCLUSION

Ce projet a permis de démontrer la mise en œuvre et l'optimisation d'un perceptron multicouche pour une tâche de classification. J'ai pu tester plusieurs configurations de modèles en variant des hyperparamètres tels que le nombre d'époch, le taux d'apprentissage, et le nombre de neurones dans les couches cachées. La matrice de confusion m'a permis de visualiser avec quelle catégorie le modèle se trompe le plus. Le modèle B, avec 150 époch, un taux d'apprentissage de 0,01 et une seule couche cachée de 100 neurones, a obtenu les meilleures performances avec une accuracy de 49 % et un temps d'entraînement raisonnable de 11 minutes.

Bien que ce modèle ait montré de bonnes performances, il reste encore des marges d'amélioration. L'introduction de techniques de régularisation telles que le dropout pourrait prévenir le surapprentissage observé dans certaines configurations. De plus, l'utilisation de méthodes d'optimisation plus avancées, comme Adam pourrait également accélérer la convergence et améliorer les performances globales du modèle.