Comparaison des performances de MLP et de CNN pour la classification sonores

$OzgurDogan^1$ Bernadette Falon[2]

[1] Université Paul Sabatier
[2] Université Paul Sabatier

ozgur.dogan@univ-tlse3.fr
adrress2

Résumé

Cet article présente une comparaison des performances d'un Multilayer Perceptron (MLP) et d'un Convolutional Neural Network (CNN) pour la classification d'événements sonores. La tâche est réalisée sur le jeu de données ESC-10 (Environmental Sound Classification), composé de 10 classes d'événements sonores. En raison du manque de données, une augmentation de données est appliquée pour améliorer les performances des modèles. Nous détaillons les architectures des modèles, les méthodes de prétraitement des données telles que la transformation en spectrogramme, ainsi que les techniques utilisées pour améliorer les performances des modèles, telles que l'ajout de couches de régularisation et d'optimiseurs spécifiques. Les performances des deux réseaux sont comparées en termes de précision de classification, mettant en évidence les avantages et les inconvénients de chaque approche.

Mots Clef

classification, CNN, MLP, ESC-10, optimisation.

Abstract

This article presents a comparison of the performance of a Multilayer Perceptron (MLP) and a Convolutional Neural Network (CNN) for the classification of sound events. The task is carried out on the ESC-10 dataset (Environmental Sound Classification), consisting of 10 classes of sound events. Due to the lack of data, data augmentation is applied to improve model performance. We detail the architectures of the models, data preprocessing methods such as transformation into spectrograms, as well as techniques used to enhance model performance, such as the addition of regularization layers and specific optimizers. The performances of the two networks are compared in terms of classification accuracy, highlighting the advantages and disadvantages of each approach.

Keywords

classification, CNN, MLP, ESC-10, optimization.

1 Introduction

Le traitement automatique des signaux sonores est une tâche cruciale dans de nombreux domaines, tels que la surveillance environnementale, la sécurité publique, et la réalité augmentée.
L'ESC-10[1] est une sélection de 10 classes issues d'un ensemble de données plus vaste, représentant trois groupes généraux de sons : des sons transitoires/percussifs, parfois avec des motifs temporels très significatifs (éternuement, aboiement de chien, tic-tac d'horloge), des événements sonores avec un contenu harmonique prononcé (bébé qui pleure, coq qui chante), et des bruits/paysages sonores plus ou moins structurés (pluie, vagues de la mer, crépitement du feu, hélicoptère, tronçonneuse). Ce sous-ensemble devrait fournir un problème plus simple pour commencer, et il a été initialement construit comme un ensemble de données de preuve de concept. La tâche de classifier des sons à partir d'un ensemble restreint de classes, une tâche triviale du point de vue humain, fixe la barre très haute pour la précision attendue des systèmes de reconnaissance sonore automatique. Par conséquent, ce sous-ensemble présente un problème légèrement différent à aborder que l'ensemble de données complet ESC-50. Les différences entre les classes sont beaucoup plus prononcées, avec une ambiguïté limitée, cela peut favoriser un type différent d'approches d'apprentissage automatique.
Dans ce travail, nous comparons deux architectures de réseaux neuronaux pour ce problème, un MLP et un CNN, sur la tâche de classification d'événements sonores en utilisant le jeu de données ESC-10. En plus de comparer différents modèles, nous étudions également l'utilisation de l'augmentation des données audio, car elle est essentielle pour l'entraînement des réseaux.

2 Description des données

Les données utilisées dans ce projet consistent en un ensemble de spectrogrammes représentant des fichiers audio, étiquetés avec leur classe correspondante. Chaque spectrogramme est une image de taille variable, généralement de dimensions 128×216, représentant les fréquences et le temps. Les classes comprennent différentes catégo-

ries d'événements sonores, comme la tronçonneuse (chainsaw), tic-tac d'une horloge(clock_tick), pleurs de bébé (crying_baby), chien(dog), hélicoptère (helicopter), pluie (rain), coq (rooster), bruit des vagues (sea_waves), éternuement (sneezing)

3 Paramètres Extraits

Les paramètres extraits des données comprennent les spectrogrammes eux-mêmes, ainsi que leurs étiquettes de classe correspondantes. Les spectrogrammes sont convertis en tenseurs et normalisés avant d'être utilisés comme entrée pour les modèles d'apprentissage profond. Les étiquettes de classe sont également converties en tenseurs correspondants.

4 Les Modèles

Nous avons utilisé 4 modèles différents comprenant deux architectures distinctes : un modèle de base MLP et un modèle de base CNN, ainsi que leurs versions améliorées. Cet article se concentre principalement sur les versions améliorées et les améliorations apportées à ces modèles.
Pour les deux modèles améliorés de MLP et de CNN, nous avons initialisé les poids avec l'initialisation de Xavier. De plus, nous avons appliqué la normalisation par batch à chaque couche pour assurer une meilleure stabilité et une convergence plus rapide lors de l'entraînement.

4.1 Initialisation de Xavier

Cette méthode vise à maintenir la variance des pondérations et des activations cohérente entre les différentes couches, afin que l'information puisse circuler de manière fluide dans le réseau. L'initialisation Xavier attribue les pondérations à partir d'une distribution uniforme ou normale avec une moyenne nulle et une variance spécifique qui dépend du nombre d'unités d'entrée et de sortie de chaque couche. Cette méthode a une effet positive pour la performance pour nos modèles.

4.2 MLP (Perceptron Multi-Couche)

1. Couche d'entrée :
   La couche d'entrée est implicitement définie par la taille des spectrogrammes en entrée, qui sont des représentations visuelles des données audio. Chaque spectrogramme est une image 2D.
2. Couches cachées :
   — La première couche cachée est une couche linéaire (nn.Linear) qui prend en entrée les données aplaties des spectrogrammes et les passe à travers une fonction d'activation ReLU (F.relu). Cette couche réduit la dimensionnalité des données.
   — La deuxième couche cachée est également une couche linéaire avec une fonction d'activation ReLU. Elle réduit progressivement la dimensionnalité jusqu'à atteindre le nombre de classes de sortie.
3. Couche de sortie : La couche de sortie est une couche linéaire qui produit des scores pour chaque classe de sortie. Ces scores sont ensuite convertis en probabilités en utilisant une fonction d'activation softmax, fournissant ainsi les prédictions finales du modèle.
4. Régularisation : Le modèle utilise également une technique de régularisation appelée dropout (nn.Dropout) pour réduire le surapprentissage. Pendant l'entraînement, certains neurones sont désactivés de manière aléatoire pour éviter qu'ils ne deviennent trop dépendants les uns des autres.

4.3 CNN (Réseau de Neurones Convolutif)

Le modèle CNN est un type de réseau de neurones profonds qui utilise des couches de convolution pour extraire des motifs et des caractéristiques spatiales des données en entrée. Dans ce modèle, nous avons une architecture à quatre couches de convolution suivies de couches de pooling, et deux couches entièrement connectées pour la classification finale.

1. Couches de convolution :
   — Le modèle commence par une série de couches de convolution (nn.Conv2d) avec une fonction d'activation ReLU (F.relu) pour extraire des caractéristiques des spectrogrammes d'entrée. Chaque couche de convolution utilise un filtre pour calculer les activations à travers l'image.
   — Les couches de convolution sont suivies de couches de pooling (nn.MaxPool2d) qui réduisent la taille spatiale des activations et réduisent la quantité de paramètres dans le modèle.
2. Couches entièrement connectées : Une fois que les caractéristiques ont été extraites par les couches de convolution et de pooling, elles sont aplaties et passées à travers des couches entièrement connectées (nn.Linear) pour la classification finale. Ces couches prennent les caractéristiques extraites et les transforment en scores de classe.
3. Couche de sortie : Comme pour le modèle MLP, la couche de sortie est une couche linéaire qui produit des scores pour chaque classe de sortie. Ces scores sont ensuite convertis en probabilités en utilisant une fonction d'activation softmax pour obtenir les prédictions finales.
4. Régularisation : Le modèle utilise également une couche de dropout (nn.Dropout) pour régulariser l'apprentissage et réduire le surapprentissage. Cela aide à améliorer la généralisation du modèle sur de nouvelles données.

5 Traitement de données

5.1 Préparation des données

La préparation des données est une étape essentielle pour obtenir de bons résultats en apprentissage automatique, nécessitant des données de qualité et bien organisées.
Nous préparons les données en suivant plusieurs étapes :

- **Chargement du corpus :** Les fichiers audio sont téléchargés localement et organisés par classe. Nous utilisons la fonction load_dataset pour charger les données.
- **Conversion en spectrogrammes :** Les fichiers audio sont convertis en spectrogrammes à l'aide de la bibliothèque librosa, permettant une représentation temps/fréquence adaptée aux modèles d'apprentissage profond.
- **Répartition des données :** Les spectrogrammes sont divisés en ensembles d'apprentissage et de test pour évaluer la capacité de généralisation des modèles.
- **Visualisation des données :** Quelques exemples de spectrogrammes sont affichés avec leur classe associée, facilitant la compréhension des caractéristiques des différentes classes et la vérification de la préparation des données.

5.2 Augmentation de données

L'augmentation de données est une stratégie cruciale pour enrichir un ensemble de données et améliorer les performances des modèles d'apprentissage automatique. Dans ce contexte, l'augmentation de données implique l'application de transformations diverses aux enregistrements audio existants afin de créer de nouvelles variations qui conservent la même sémantique mais présentent des différences subtiles.
Nous avons implémenté plusieurs techniques d'augmentation de données audio[2], notamment :

- **Ajout de Bruit Blanc :** Cette technique consiste à ajouter du bruit blanc gaussien au signal audio, simulant les bruits de fond courants.
- **Étirement Temporel :** Le signal audio est étiré ou compressé dans le temps, modifiant ainsi sa durée sans altérer sa hauteur tonale.
- **Transposition de Hauteur :** Cette technique permet de transposer la hauteur tonale du signal audio vers le haut ou vers le bas en un nombre spécifié de demi-tons.
- **Gain Aléatoire :** Un gain aléatoire est appliqué au signal audio pour simuler les variations de volume.
- **Inversion de Polarité :** La polarité du signal audio est inversée, ce qui entraîne une inversion de phase du signal.

Nous avons intégré ces transformations dans notre pipeline d'augmentation de données, en les appliquant de manière aléatoire à environ un tiers des enregistrements audio du corpus d'apprentissage. Cela garantit une diversité suffisante dans notre ensemble de données, améliorant ainsi la capacité des modèles à généraliser à de nouvelles données. Après l'augmentation, les nouvelles données sont intégrées à l'ensemble de données d'origine, augmentant ainsi sa taille et sa variabilité. Cette approche contribue à réduire le surajustement et à améliorer les performances des modèles sur de nouveaux enregistrements audio. Nous pourrions ainsi faire une augmentation sur les images de spectre mais ça pourrait poser la risque d'éloigner de son pertinent.

6 Résultats

en raison de divers facteurs tels que l'initialisation des poids, les méthodes d'entraînement aléatoire et d'autres paramètres d'hyperparamètres, nous avons observé des variations dans les performances des modèles.

6.1 MLP

Même si MLP n'est pas le plus adapté pour classification d'image nous avons obtenu une assez bonne performance, il était plus rapide que CNN pour l'entraînement. En utilisant $l'optima$ pour régler les hyperparamètres. Au final, nous avons reussi à avoir une précision jusqu'à $\%89$ mais au moyenne c'était autour de $\%60$ donc nous allons présenter ce modèle.

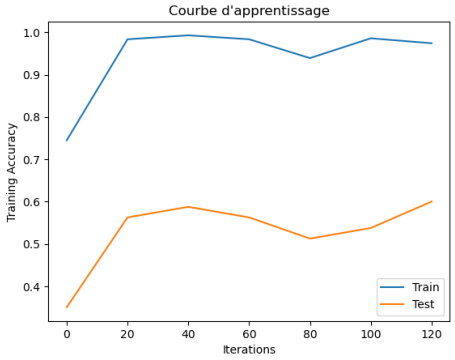

FIGURE 1 – Précision Train : 0.97, Test : 0.6

Malgré les bonnes performances sur l'ensemble d'entraînement, le MLP a rencontré des difficultés à généraliser les données, comme le montre la différence de précision entre l'ensemble d'entraînement et l'ensemble de test.
Malgré les amélioration effectuées, ce modele n'est pas assez performant pour bien distinguer les sons.

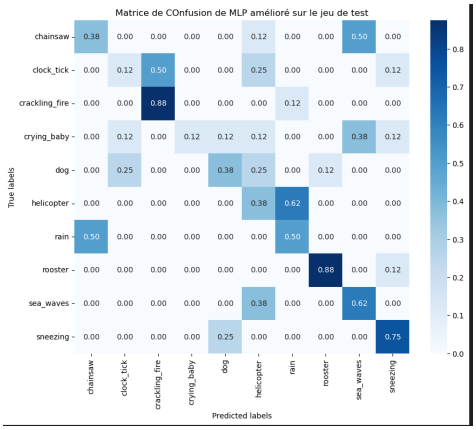

FIGURE 2 – Précision Train : 0.97, Test : 0.6

6.2 CNN

Pour le CNN nous avons eu des performance mieux et plus pertinent que MLP même si elle est plus gourmand en fonction de performance et

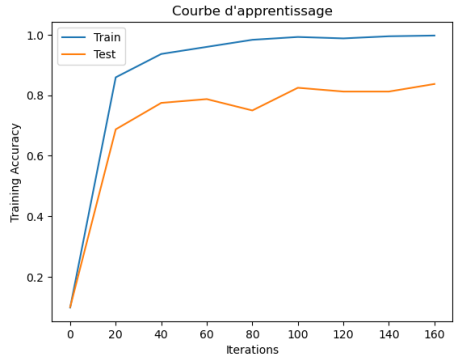

FIGURE 3 – Précision Train : 0.99, Test : 0.84

Nous remarquons bien qu'il est satisfait et il classe bien les sons mais quand même il a quelques problemes entre les classes "helicopter" et "dog".

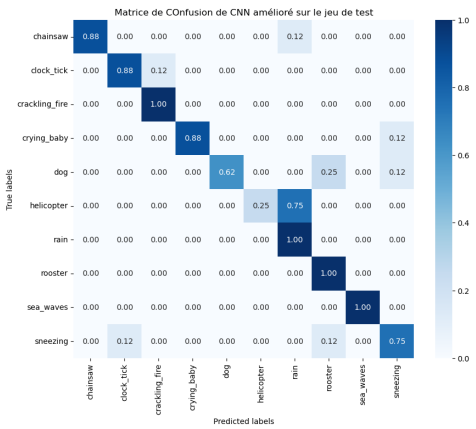

FIGURE 4 – Précision Train : 0.97, Test : 0.6

7 Conclusion

Après avoir implémenté nos deux réseaux et exploré différentes architectures pour optimiser leurs performances, nous constatons que le CNN surpasse le MLP dans la classification des images de spectres d'événements sonores. Mais aussi en trouvant les bonne paramètres il est toujours possible d'améliorer la performance de ces 2 modèles. Cependant, pour approfondir cette tâche, il pourrait être intéressant d'adopter une approche alternative consistant à classer les images en fonction du son correspondant. Cette approche pourrait offrir des informations supplémentaires et améliorer notre compréhension des relations entre les caractéristiques audio et visuelles des événements sonores.

Annexe

Nous voudrions remercier à Thomas Pellegrini et Philippe Muler pour leur cours de Apprentissage Automatique.

Références

[1] Karol J. Piczak, *ESC : Dataset for Environmental Sound Classification*

[2] Valerio Velardo, *Audio Data Augmentation Tutorial,* 2022.
`https://github.com/musikalkemist/audioDataAugmentationTutorial/blob/main/3/dataaugmentation.py`