Réseaux de neurones profonds pour la classification audio avec MLP et CNN

Narimene CHOUIAL[1] Meriem DAHMANI[2]

Université Toulouse III – Paul Sabatier, Toulouse, France

narimene.chouial@univ-tlse3.fr
meriem.dahmani@univ-tlse3.fr

Résumé

Les avancées récentes dans le domaine de l'apprentissage automatique ont suscité un intérêt croissant pour de nombreux problèmes de classification, en particulier pour les données sous forme d'images, de vidéos et de fichiers audio. L'un des problèmes de classification les plus importants consiste à classifier des sons et à prédire la catégorie de ce son. Certaines des applications où un tel modèle de classification peut être appliqué dans le monde réel sont les systèmes de sécurité, la classification de clips musicaux pour identifier le genre de la musique, la classification de différents sons environnementaux, la détection et la vérification de locuteurs. La classification audio est la tâche d'analyser différents signaux audios. Pour cela, nous implémentons sur PyTorch un MLP et un CNN, ainsi qu'une fonction d'entraînement. Nous comparons ensuite nos deux réseaux en faisant varier les paramètres d'entraînement afin d'estimer au mieux les performances de chacun.

Mots Clef

MLP, CNN, Performance, Classification, Sons, Spectrogramme, Comparaison, Précision.

1 Introduction

Les sons contiennent des informations riches et aident les gens à percevoir les environnements qui les entourent. Les gens sont capables de reconnaître des sons complexes et de filtrer les informations pertinentes. De cette manière, le bruit inutile est éliminé et les informations brutes sont distillées. Aujourd'hui, les capteurs peuvent facilement collecter des tonnes de données audios brutes ; cependant, leur traitement pour obtenir des informations significatives reste ardu.

La majorité de la recherche en classification audio se concentre sur une tâche de classification spécifique pour obtenir une grande précision. Cependant, en raison de la complexité des données audio, diverses techniques doivent être utilisées pour analyser les données. Pour classer efficacement les données audio, les caractéristiques doivent être extraites de l'échantillon audio. Trois techniques largement utilisées pour la recherche en classification audio sont le coefficient cepstral de la fréquence de Mel (MFCC), le taux de croisement nul (ZCR) et le codage prédictif linéaire (LPC) [2, 3]. Les MFCC ont été utilisés pour l'extraction de caractéristiques afin d'améliorer la reconnaissance de locuteur. Après cela, la machine à vecteurs de support (SVM) et le modèle de mélange gaussien (GMM) sont utilisés pour effectuer les classifications [3]. Les MLPs (Multi-Layer Perceptrons) peuvent être utilisés pour la classification audio en tant que classificateurs binaires ou multi-classes. Ils ont une structure de réseau de neurones artificiels (ANN) avec une ou plusieurs couches cachées. Pour la classification audio, les MLPs peuvent être entraînés avec des caractéristiques extraites à partir des fichiers audio, telles que les MFCC ou les caractéristiques temporelles et spectrales. Les ils peuvent également être utilisés avec des réseaux de neurones convolutifs (CNN) pour améliorer la précision de la classification.

Récemment, les réseaux de neurones convolutionnels (CNN) ont été utilisés pour apprendre automatiquement des représentations de caractéristiques à partir de données complexes [3]. Cette technique universelle a été appliquée dans de nombreux domaines pour remplacer les conceptions de fonctions ad hoc et a raccourci une période de développement décennale à quelques mois.

2 Description des données et des paramètres

2.1 Le corpus

Pour tester les performances, nous utiliserons un corpus contenant des enregistrements audios répartis en 10 catégories.[1] Le but est donc d'entraîner un modèle capable de ranger un enregistrement dans une des 10 catégories.

2.2 L'apprentissage

Pour entraîner les modèles, nous faisons une fonction d'entraînement utilisant la "Cross Entropy Loss" et pouvant utiliser deux optimiseurs : Adam ou SGD. Nous lui donnons 320 enregistrements d'entraînement, et 80 enregistrements de test. Nous allons ensuite créer des groupes aléatoires d'enregistrements, appelés "batch". Pour chaque groupe, nous allons passer les enregistrements dans le modèle, afin de calculer les nouveaux poids de notre réseau. Nous détaillerons les paramètres que nous ferons varier dans chaque sous-partie.

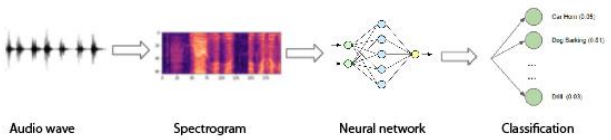

Figure 1 – Cycle d'apprentissage

3 Description des modèles

3.1 MLP

Définition. Le perceptron multicouche (multilayer perceptron, noté MLP), est un réseau de neurones organisé en couches [4]. Une couche est un groupe de neurones qui n'ont pas de liens les uns avec les autres. Un MLP contient au minimum deux couches : une couche cachée et une couche de sortie, mais on peut ajouter autant de couches cachées que l'on veut. Chaque couche cachée interagit avec la couche précédente seulement, ce qui nous donne pour la k^{ième} couche cachée :

$$h^{(k)}(x) = g(b^{(k)} + W^{(k)} h^{(k-1)}(x))$$

où $h^{(k)}$ est la k^{ième} couche cachée, g(x) la fonction d'activation, $b^{(k)}$ le biais et $W^{(k)}$ la matrice de poids. On initialise $h^{(0)}(x)$ = x. Une fonction d'activation différente peut être appliquée sur la couche de sortie.

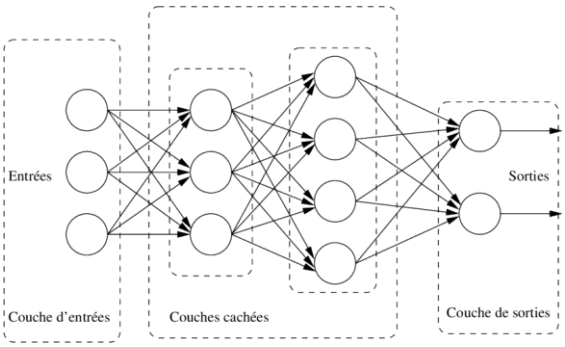

Figure 2 - Représentation d'un MLP

Notre implémentation. Nous avons implémenté une classe avec différents paramètres pour personnaliser le nombre de couches et leurs valeurs. Il est possible de personnaliser le nombre de couches cachées, ainsi que leur nombre de neurones. On peut aussi changer la fonction d'activation, et décider où avoir un BatchNorm.

3.2 CNN

Définition. Un réseau de convolution (convolutional neural network, noté CNN) est un type de réseau de neurones créé au départ pour traiter des images. Le principe du CNN est assez semblable à celui du MLP : on retrouve une couche de sortie et une ou plusieurs couche(s) cachée(s) [3]. Dans les couches cachées nous avons plusieurs couches de convolution, suivies de couches d'activations, du "pooling" qui n'est autre que du sous-échantillonnage, des couches de normalisation qui servent à accélérer et éventuellement, améliorer l'apprentissage, et des couches "fully connected" pour faire la classification. Afin de mieux visualiser ce qu'est un CNN, une représentation graphique est présentée en Figure 2.

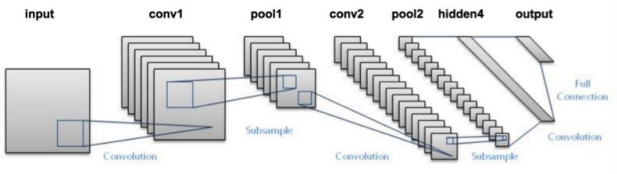

Figure 3 - Représentation d'un CNN

Notre implémentation. Nous avons implémenté une classe avec différents paramètres pour personnaliser le nombre de couches et leur valeur. Il est possible de personnaliser la taille du noyau, le nombre de neurones dans une couche cachée, le nombre de convolutions ainsi que la taille de leurs canaux en sortie. On peut aussi changer la fonction d'activation.

4 Résultats

4.1 MLP

Notre MLP de base dispose d'une couche cachée de 50 neurones, et a également une fonction d'activation ReLu. Cet MLP contient 1382960 paramètres. Nous l'avons entraîné avec des groupes de 32 "batchs". Voici certains tests que nous avons réalisés, avec 20 epochs, l'optimiseur "Adam" (sauf contre-indication) et un degré d'apprentissage de 0.0001 :

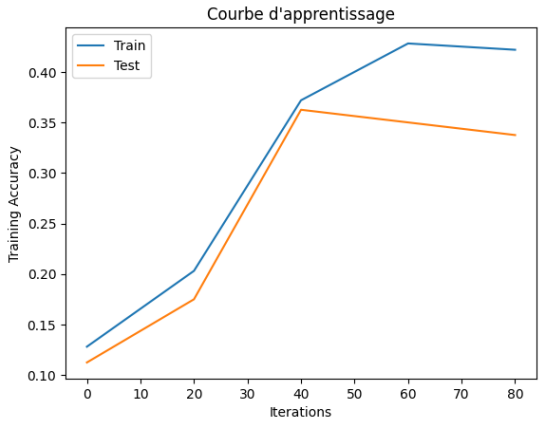

Figure 4 – Courbe de précision avant l'amélioration MLP

Modification	Préc.Train	Préc.Test
Modèle de base	0.33	0.25
Optimiseur SGD	0.19	0.1
Activation Sigmoid	0.47	0.45
num_hidden = 1000 (problème de overfitting)	***0.82***	***0.55***
BatchNorm1D	***0.89***	***0.67***

Table 1 : Résultats obtenus avec différentes versions d'un modèle MLP

Le meilleur résultat est obtenu avec le modèle de base auquel on a augmenté le nombre de neurones dans les couches cachées de 50 à 1000. Nous avons donc deux couches séparées par une couche de BatchNorm afin d'éviter le problème d'overfitting. On peut voir sur la Figure 5 les courbes représentant la précision du modèle en fonction des itérations pour les données d'entraînements et les données de tests, dans la version du MLP donnant les meilleurs résultats.

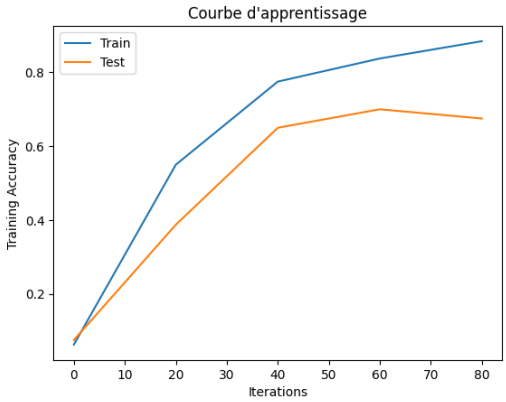

Figure 5 - Meilleur résultat obtenu pour notre modèle MLP

Malgré les améliorations effectuées, ce modèle reste non performant pour cette tâche de classification. On a décidé de passer à un modèle avec une architecture plus compliqué : Réseau neuronal convolutif (CNN).

4.2 CNN

Notre CNN de base dispose de trois couches de convolutions avec des noyaux de taille 3x3 et à 8, 16 et 32 canaux en sortie respectivement, un pooling qui s'applique après chaque couche de convolution ainsi que deux couches "fully connected" fc1 et fc2. fc1 à 50 neurones et fc2 10 neurones. Notre modèle a également une fonction d'activation ReLu appliquée après chaque couche de convolution et après fc1. Notre CNN de base contient 697648 paramètres, soit plus de 2 fois moins que le MLP. Nous l'avons entraîné avec des groupes de 32 "batchs". Voici certains tests que nous avons réalisés.

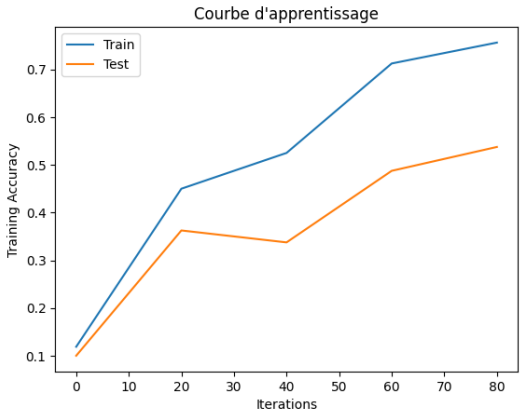

Figure 6 – Courbe de précision avant l'amélioration CNN

Modification	Préc.Train	Préc.Test
Modèle de base	0.77	0.56
Optimiseur SGD	0.61	0.5
Noyaux 5X5	0.77	0.46
couche conv(32,64) + batchNorm2D	0.95	0.80
couche conv(64, 128) + batchNorm2D	0.94	0.775
couche caché (500)	1	0.81
num_epochs = 20	1	0.83
Noyaux 9X9	0.98	0.73

Table 2 : Résultats obtenus avec différentes versions d'un modèle CNN

Le meilleur résultat est obtenu avec le modèle de base auquel on a ajouté deux couches de convolution (32, 64) et (64, 128) et on a modifié le nombre de neurones dans la couche cachée entièrement connectée (500 au lieu de 50) et on a augmenté le nombre d'epochs à 200, cela améliore la précision du train mais produit un problème d'overfitting. Pour régler ce problème on a utilisé une "BatchNorm" après chaque couche de convolution, ce qui a amélioré la pression du test et la performance de notre modèle.

La Figure 7 nous montre les courbes représentant la précision du modèle en fonction des itérations pour les données d'entraînements et les données de tests dans la version du CNN donnant les meilleurs résultats. On observe un léger sur-apprentissage.

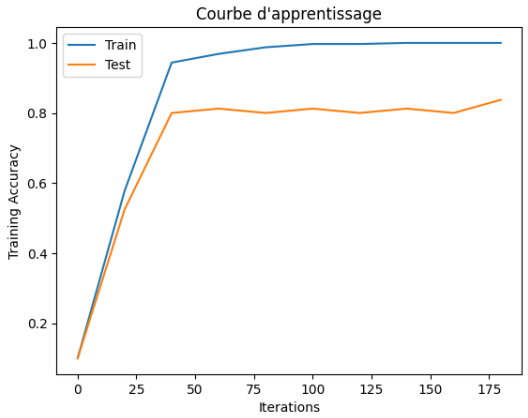

Figure 7 - Meilleur résultat obtenu pour notre modèle CNN

4.3 Analyse

On voit que malgré les améliorations apportées, le modèle CNN reste meilleur que le modèle MLP pour la classification d'évènements sonores. Cela peut venir du fait que CNN a été créé pour la classification d'image et que l'on classe les évènements sonores grâce à leur spectrogramme. De plus, on voit que notre MLP contient plus du double de nombre de paramètres que notre CNN, cela vient du fait que MLP est entièrement connecté. En effet, chaque nœud est connecté à tous les autres nœuds de la couche suivante et de la couche précédente. Ce surplus de paramètre entraîne des redondances qui causent des difficultés lors de l'entraînement. Notre CNN, quant à lui, ne contient que les deux dernières couches entièrement connectées, ce qui réduit le nombre de paramètres et améliore l'apprentissage.

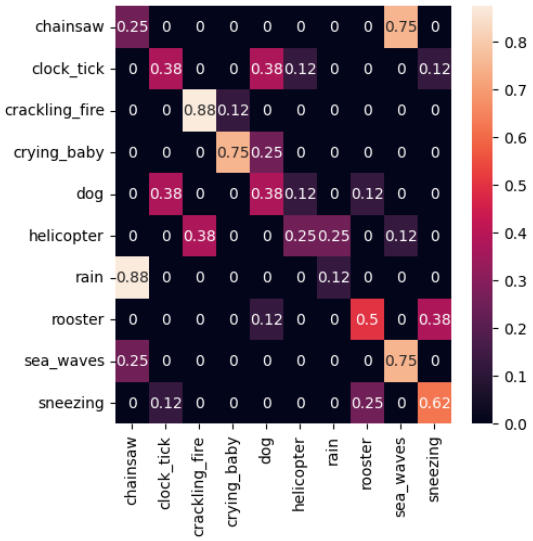

Figure 8 – Matrice de confusion MLP

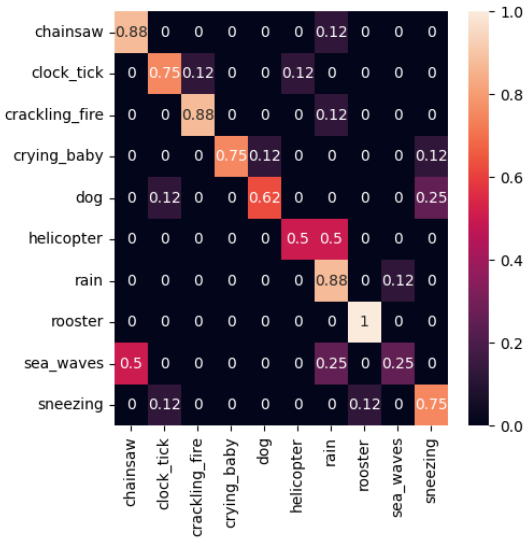

Figure 9 – Matrice de confusion CNN

5 Conclusion

Après avoir implémenté nos deux réseaux et avoir essayé plusieurs architectures différentes afin d'améliorer les résultats, nous pouvons en déduire que le CNN donne de meilleurs résultats pour la classification d'évènements sonores que le MLP.

Si on souhaite obtenir de meilleurs résultats, il faudrait envisager d'utiliser un jeu de données plus grand, ou bien utiliser la technique de data augmentation pour augmenter la taille et la diversité de l'ensemble de données d'entraînement. On peut éventuellement utiliser un autre type de réseau de neurones (RNN par exemple) qui ne se sert pas du spectrogramme.

Annexe

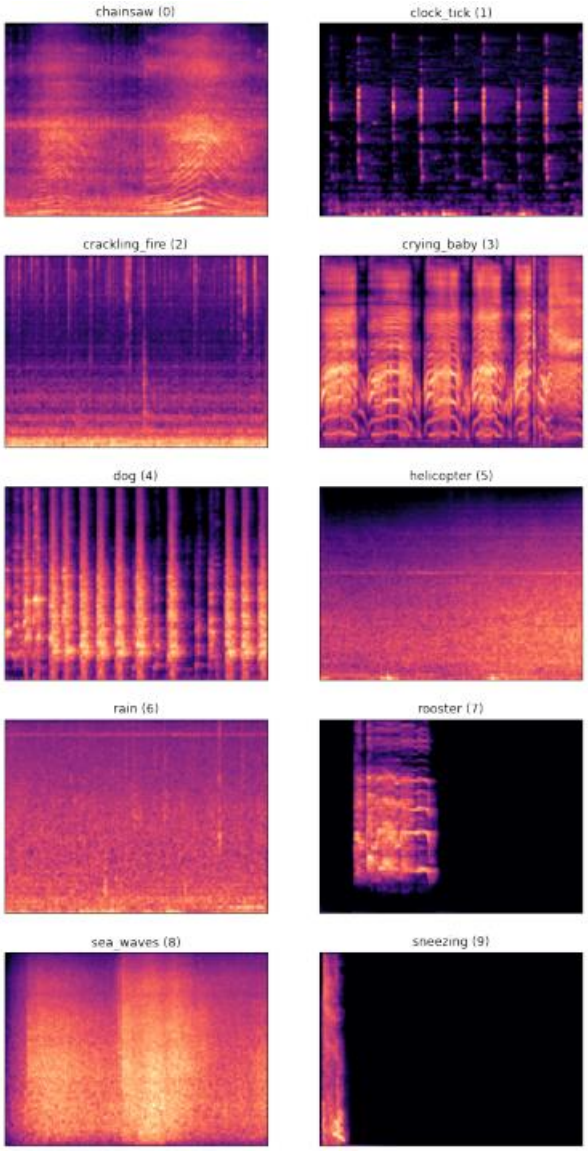

Figure 10 - Spectrogrammes des différentes catégories d'enregistrement

Bibliographie

[1] Pellegrini, T. : Audio dataset (Oct 2018), https://www.irit.fr/Thomas.Pellegrini/ens/M2RFA/dataset.zip

[2] Audio Classification: Environmental sounds classification, *Dept. of Computing Science (Multimedia) University of Alberta*

[3] Apprentissage profond pour la classification audio : comparaison entre MLP, RNN et CNN : *Ali El Moussawi, Fahed Abdallah et Hazem Hajj*

[4] Audio Classification: Environmental sounds classification *document (hal.science)*