Comparaison des performances des architectures MLP et CNN pour la classification des fichiers audio : étude sur la base de données ESC-10

EL AMRANI Wadie[1] BARSEGHYAN Anna [1]

[1] Université Paul Sabatier, Toulouse, France

wadie.el-amrani@univ-tlse3.fr
anna.barseghyan@univ-tlse3.fr

Résumé

Dans ce papier, nous proposons une méthode de classification sonore des fichiers audio de la base de données ESC-10. Nous transformons la classification des fichiers sonores en une classification d'images de spectrogrammes, et appliquons des méthodes de classification basées sur les perceptrons multicouches (MLP) et les convolutionnels neuronaux (CNN). Nous comparons les résultats pour évaluer l'efficacité de notre approche, nous avons obtenu une précision de 72% pour MLP et 85% pour CNN sur le jeu de donné de test.

Mots Clef

Apprentissage automatique, traitement de signal sonore, MLP, CNN.

1 Introduction

Le traitement du signal audio est un domaine en constante évolution, avec des applications variées dans les domaines de la reconnaissance vocale, de la classification musicale et de l'analyse acoustique. Dans ce contexte, l'apprentissage automatique offre des outils puissants pour extraire des caractéristiques pertinentes à partir de signaux audio complexes. Cependant, la plupart des approches traditionnelles se concentrent sur l'analyse directe des signaux audio, sans explorer d'autres représentations possibles. Dans ce projet, nous nous sommes intéressés à l'utilisation de spectrogrammes comme représentation visuelle pour la classification des sons audio. Les spectrogrammes offrent une représentation graphique de la fréquence et de l'intensité du signal audio, permettant ainsi une analyse visuelle de la structure sonore. Nous avons utilisé le framework PyTorch [6] pour développer des architectures de réseaux de neurones convolutifs (CNN) et de perceptrons multicouches (MLP) pour classifier les spectrogrammes issus de la base de données ESC-10 [7]. Les résultats obtenus montrent l'efficacité de cette approche pour la classification des sons audio, ouvrant ainsi des perspectives pour de futures applications dans le domaine de l'analyse audio.

2 Dataset

Le ESC-10 est une sélection de 10 classes issues de la base de données plus large ESC-50 [7], représentant trois groupes de sons généraux :

- *des sons transitoires/percutants, parfois avec des patterns temporels très significatifs (éternuement, aboiement de chien, tic-tac de montre),*
- *des événements sonores avec un contenu harmonique fort (bébé pleurant, coq chantant),*
- *des paysages sonores plus ou moins structurés (pluie, vagues de mer, craquement de feu, hélicoptère, tronçonneuse).*

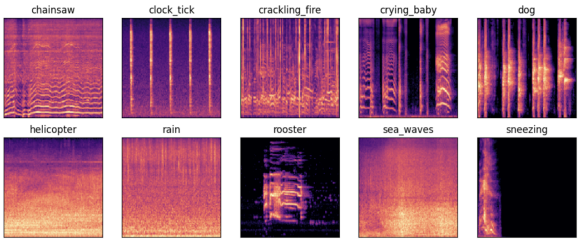

FIGURE 1 – Exemples de spectrogrames des fichiers sonores de la Dataset ESC-10

Nous utiliserons ce sous-ensemble car il devrait fournir un problème plus facile à commencer, et il a été initialement construit comme un jeu de données de preuve de concept. La base de données ESC-10 comprend 400 enregistrements de 10 classes. Chaque classe comprend 40 enregistrements, et chaque enregistrement est de 5 secondes.

2.1 Augmentation de donnés

Le manque de données est un gros problème pour utiliser les modèles d'apprentissage profond, comme les MLP et les CNN. Ce problème est très présent dans les domaines où il est difficile ou impossible de rassembler beaucoup de données. Le dataset ESC [7] pour la classification des sons environnementaux, en particulier l'ensemble ESC-10,

	Original	Augmented with TS	Augmented with PS1	Augmented with PS2	Combined
Train	32	128	128	128	416
Test	8	32	32	32	104

TABLE 1 – Nombre de fichiers audios pour chaque ensemble de données, y compris les données originales ainsi que les données augmentées avec Time Stretching (Étirement Temporel) - TS, Pitch Shifting (Décalage de Hauteur) - PS1 et PS2, ainsi que les données augmentées combinées [8]

contient seulement 40 enregistrements par classe. Cela limite l'entraînement des modèles et peut réduire leur efficacité. Plusieurs études montrent que de plus grands ensembles de données améliorent la performance des modèles [1, 2, 5]. Notre méthode s'inspire de celle de [8] et utilise des techniques spécifiques à l'audio comme l'étirement du temps et le changement de ton pour augmenter la taille des données.

Étirement Temporel (TS). *L'étirement temporel consiste à ralentir ou accélérer l'échantillon audio sans modifier la hauteur tonale. Nous avons appliqué cet étirement à chaque échantillon selon quatre facteurs : {0,81, 0,93, 1,07, 1,23}.*

Décalage de Hauteur (PS1). *Le décalage de hauteur modifie la hauteur tonale de l'échantillon audio tout en conservant la durée originale. Chaque échantillon a été modifié par quatre valeurs (en demi-tons) : {-2, -1, 1, 2}.*

Décalage de Hauteur (PS2). *Nous avons introduit un second ensemble de décalage de hauteur. Dans cet ensemble, chaque échantillon a été modifié par quatre valeurs plus importantes (en demi-tons) : {-3,5, -2,5, 2,5, 3,5}.*
Nous avions implementer ces differentes techniques en utilisant la librairie librosa

3 Architecture

Cette section décrit quatre architectures conçues pour cette tâche. Il s'agit de deux architectures MLP et de deux architectures CNN de différentes complexités. Le modèle MLP est caractérisé par des variations dans le nombre de couches cachées et l'utilisation de la normalisation des lots, tandis que les modèles CNN se distinguent par des différences dans le nombre de blocs convolutionnels et de couches de normalisation des lots, ce qui permet d'explorer diverses stratégies de traitement des données.

3.1 Architecture MLP

Les réseaux neuronaux denses, également connus sous le nom de perceptrons multicouches, sont une architecture fondamentale en apprentissage profond. Ils se composent de multiples couches de neurones interconnectés, où chaque neurone dans une couche est connecté à chaque neurone dans les couches adjacentes. La représentation mathématique d'un MLP peut être notée comme :

$$Z^{(0)} = X$$
$$Z^{(i)} = f^{(i)}(W^{(i)} \cdot Z^{(i-1)} + b^{(i)})$$
$$\hat{Y} = f^{(out)}(W^{(out)} \cdot Z^{(out-1)} + b^{(out)})$$

Dans la definition precedante $Z^{(0)}$ représente les valeurs d'entrée. $Z^{(i)}$ représente les valeurs de pré-activation de la couche i. $\hat{Y}$ représente la sortie prédite. $W^{(i)}$, $b^{(i)}$ sont respectivement la matrice de poids et le vecteur de biais de la couche i. $f^{(i)}$ est la fonction d'activation de la couche i. Nous avons testé plusieurs architectures MLP pour la tâche donnée. Nous parlerons de deux d'entre elles dans les sections suivantes.

Architecture MLP simple. *L'architecture représentée dans la figure 2 est composée de 2 couches entièrement connectées. La couche cachée compte 50 neurones et une activation ReLU lui a été appliquée. La couche de sortie comporte 10 neurones avec une activation softmax.*

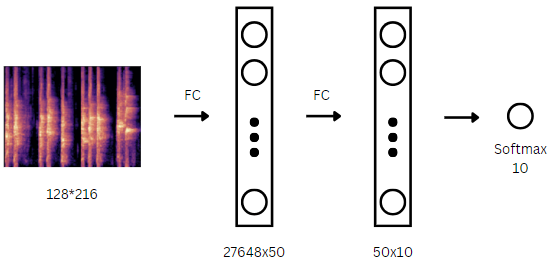

FIGURE 2 – Architecture MLP simple avec une couche cachée

Architecture MLP complex. *L'architecture illustrée sur la figure 3 comprend trois couches cachées avec une activation relu et une couche de sortie avec une activation softmax. Pour stabiliser les gradients et améliorer*

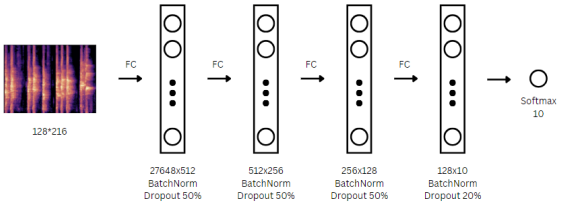

FIGURE 3 – Architecture MLP complex avec trois couches cachée

la vitesse d'entraînement, nous avons utilisé la Batch Normalisation[3].

$$BN(x) = \gamma \frac{x - \mu}{\sigma} + \beta$$

où :

— γ et β sont des paramètres apprenables

— μ et σ sont la moyenne et l'écart type des données d'entrée

*Pour améliorer la capacité du modèle à généraliser et réduire le surapprentissage, nous avons ajouté des couches de dropout[**?**]. Les couches de dropout fonctionnent en désactivant aléatoirement un pourcentage des neurones pendant la phase d'entraînement d'un réseau neuronal. Cela signifie que ces neurones ne contribuent pas à la propagation avant ni à la rétropropagation des gradients pendant cette étape.*

3.2 Architecture CNN

Le réseau neuronal convolutionnel (CNN) est un type de réseau neuronal conçu spécifiquement pour traiter des données structurées en grille, telles que des images. Les CNN sont particulièrement efficaces dans des tâches telles que la classification d'images. L'architecture CNN pour la classification d'images se compose de deux parties : l'extraction de caractéristiques réalisée à l'aide de couches convolutionnelles et la classification des représentations d'images effectuée par un MLP.
L'extraction des caractéristiques implique l'application d'une opération de convolution, où un filtre est déplacé sur l'image et le produit scalaire entre l'image et les poids du filtre est calculé. La définition mathématique de la convolution est donnée par :

$$(f \star g)[i,j] = \sum_{m=-\infty}^{\infty} \sum_{n=-\infty}^{\infty} f[m,n] \cdot g[i-m, j-n]$$

Cependant, dans les tâches d'optimisation où l'objectif est d'apprendre les poids optimaux du filtre pour la classification des images, il n'est pas nécessaire d'inverser le filtre lors de la convolution. Cela introduirait une complexité computationnelle inutile sans affecter le processus d'apprentissage. En pratique, les couches convolutionnelles utilisent la corrélation mathématique, définie comme suit :

$$(f \star g)[i,j] = \sum_{m=-\infty}^{\infty} \sum_{n=-\infty}^{\infty} f[m,n] \cdot g[i+m, j+n]$$

MaxPooling peut également être utilisé en conjonction avec la convolution. Agissant comme une forme de sous-échantillonnage, il réduit efficacement la dimensionnalité des cartes de caractéristiques tout en conservant les traits les plus saillants identifiés par les couches de convolution. Ce processus implique une fenêtre de regroupement qui parcourt la carte de caractéristiques, extrait la valeur maximale dans chaque fenêtre, et la transmet à la sortie de la couche de regroupement.
Nous avons testé plusieurs architectures CNN pour la tâche donnée. Nous parlerons de deux d'entre elles dans les sections suivantes.

Architecture CNN simple. *L'architecture simple du CNN est présentée sur la figure 4. Cette architecture comprend trois couches de convolution avec une taille de noyau de 3x3 et un nombre de filtres respectivement de 8, 16 et 32. Suivies d'une couche de MaxPooling de taille 2x2 et d'un réseau MLP avec une couche cachée de 50 neurones et une couche de sortie de 10 neurones. Des fonctions d'activation ReLU sont utilisées pour toutes les couches, à l'exception de la couche de sortie où nous avons une fonction d'activation softmax.*

Architecture CNN complexe. *Cette architecture comprend quatre couches de convolution, chacune utilisant un filtre de taille 3x3. Ces couches sont conçues avec respectivement 32, 64, 128 et 128 filtres, capturant progressivement des caractéristiques hiérarchiques. Après chaque opération de convolution, une couche de normalisation par lots est appliquée pour améliorer la stabilité et la vitesse de l'entraînement. Ensuite, une couche de max-pooling avec une taille de fenêtre de 2x2 est utilisée pour sous-échantillonner les cartes de caractéristiques, préservant ainsi les informations essentielles tout en réduisant la dimensionnalité. La représentation des caractéristiques résultante est ensuite alimentée dans une couche dense avec une taille cachée de 512 neurones, servant d'extracteur de caractéristiques de haut niveau. Enfin, la couche de sortie se compose de 10 neurones avec une fonction d'activation softmax, facilitant la classification à travers plusieurs classes.*

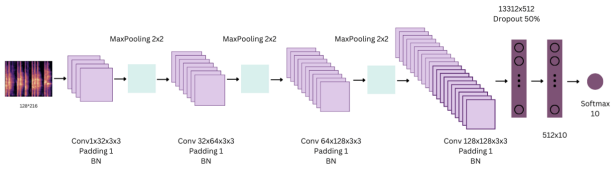

FIGURE 4 – Architecture CNN simple

4 Méthodologie

Dans cette étude, nous avons utilisé des modèles basés sur les architectures décrites dans la section 3. Nous avons entraîné ces modèles sur quatre types de données ESC-10. Ces types sont : les données originales, les données originales avec étirement temporel (ts), les données originales avec décalage de hauteur (ps), et les données originales avec ts et ps. Ces méthodes d'augmentation sont expliquées dans la section 2.1.

Entrainement. *Pour la classification de plusieurs classes, nous avons utilisé la perte d'entropie croisée. Nous aurions pu utiliser la log-vraisemblance négative. Mais, cela aurait nécessité une fonction d'activation softmax dans notre modèle. PyTorch gère déjà l'entropie croisée automatiquement.*

Nous avons testé différentes architectures de réseaux neuronaux. Nous avons changé le nombre de couches et de neurones. Nous avons ajouté une normalisation et un dropout pour les modèles MLP. Pour les CNN, nous avons ajusté la taille des noyaux et ajouté un pooling.
Une stratégie consistait à essayer d'obtenir un overfitting modéré initialement, puis à le réduire à l'aide de dropouts et de régularisations. Si un overfitting ne pouvait pas être obtenu, cela indiquait que la complexité de l'architecture pourrait ne pas être suffisante pour la tâche, comme ce fut le cas avec l'architecture MLP simple.
Les optimiseurs ADAM et SGD ont été utilisés pour l'entraînement, et les résultats ont montré qu'ADAM était plus performant que SGD.
Le pseudo algorithme d'entrainement est comme suivant :

Algorithm 1 Procédure d'entraînement pour modèles d'apprentissage profond

Require: modèle, chargeur d'entraînement, nombre d'époques

```
configurerAppareil()
initialiserOptimiseur()
initialiserFonctionDePerte()
for époque = 1 to nombre d'époques do
    for chaque lot in chargeur d'entraînement do
        préparerDonnées(lot)
        modèle.entrainer()
        sorties ← modèle(lot.entrées)
        perte ← calculerPerte(sorties, lot.cibles)
        effectuerRétropropagation(perte)
        mettreÀJourParamètresDuModèle()
        réinitialiserGradients()
    end for
    afficherProgrès(époque)
end for
afficherRésultatsD'entraînement()
```

Recherche sur Grille. *L'objectif d'un algorithme d'apprentissage est de trouver une fonction f qui minimise la fonction perte sur des échantillons ,dans notre cas,c'est un problème de classification de fichiers sonores, notre fonction f est un mapping entre les fichiers sonores et leurs vrai classes, ces algorithmes sont produits par l'optimisation d'un critere d'entrainement qui est la minimsation de la fonction cross entropie, une approche couramment utilisé est la recherche sur grille [4]. cette méthode consiste à définir une grille d'hyperpamaètres et à évaluer chaque combinaison.*

Evaluation. *Pour évaluer nos modèles, nous avons utilisé l'ensemble de test original, constant pour toutes les expériences. Cette approche permet de comparer efficacement les différentes architectures et techniques d'augmentation sonore. L'ensemble de test comprend 80 fichiers audio, répartis en 8 enregistrements pour chacune des 10 classes. Nous avons mesuré la performance des modèles à l'aide de la métrique de précision. Les détails et les résultats de ces évaluations sont discutés dans la section 5.*

5 Résultats

Ni les architectures MLP simples ni complexes n'ont donné de résultats satisfaisants. ***L'architecture MLP simple*** *n'a pas réussi à overfit, même avec un réglage hyperparamétrique, ce qui indique son incapacité à comprendre suffisamment l'ensemble de données. L'incapacité à overfit le modèle issu de cette architecture souligne son inadéquation à la complexité de l'ensemble de données. La meilleure précision obtenue par ce modèle était de 18 %, une amélioration marginale par rapport à une supposition aléatoire, étant donné que l'ensemble de données comprend 10 classes avec une supposition aléatoire donnant une chance de 10 % de précision.*
Dans le cas du ***architecture MLP complexe****, une précision maximale de 72 % a été atteinte, ce qui représente une amélioration significative par rapport à l'architecture simple. Cependant, cette amélioration, bien que notable, ne parvient toujours pas à fournir un résultat satisfaisant. Il est important de noter que cette amélioration s'est accompagnée d'une augmentation de dix fois du nombre de paramètres. Nos résultats montrent une amélioration considérable des performances avec l'utilisation des réseaux de neurones convolutionnels (CNN). Nous avons atteint une précision de 60 % avec un CNN simple et de 86 % avec un CNN plus complexe, surpassant significativement le modèle de perceptron multicouche (MLP). D'après nos constatations, il est évident que les architectures MLP sont généralement moins efficaces pour la classification d'images. Cela peut être attribué à plusieurs facteurs clés :*

1. ***Manque de conscience spatiale*** *: Les couches denses des MLP traitent chaque pixel de manière indépendante, sans prendre en compte les relations spatiales entre les pixels voisins, essentielles pour comprendre le contenu de l'image.*
2. ***Dimensionnalité élevée des données d'image*** *: Les images contiennent des milliers ou des millions de pixels, ce qui entraîne un grand nombre de connexions dans les réseaux neuronaux denses. Cela conduit à une prolifération de paramètres, rendant l'entraînement lent et coûteux en termes de calcul.*
3. ***Absence de structure hiérarchique*** *: Contrairement aux CNN, les MLP ne possèdent pas de structure hiérarchique conçue pour apprendre des représentations hiérarchiques des caractéristiques. Les CNN capturent les caractéristiques simples telles que les contours dans les couches inférieures et les caractéristiques complexes telles que les parties d'objets dans les couches supérieures, ce qui facilite une meilleure généralisation aux nouvelles images.*

Ces facteurs limitent collectivement l'efficacité des architectures MLP pour les tâches de classification d'images, mettant en évidence la supériorité des CNN dans la ges-

tion de la complexité spatiale et de la nature hiérarchique des données d'image.
Bien que ADAM ait donné de meilleurs résultats sur l'ensemble de test, la différence plus faible de précision entre les ensembles d'entraînement et de test lorsque nous utilisons SGD suggère que SGD était moins sujet au surapprentissage dans cette tâche.
*D'autre part, nous observons dans le tableau ci-dessus les résultats de précision obtenus en utilisant les meilleurs hyperparametres des architectures CNN et des MLP sur différents ensembles de données d'entraînement. Nous observons que l'ajout de techniques d'augmentation (**Ts**, **Ps**) conduit à une augmentation notable des performances des CNN et des MLP, ce qui se traduit par des taux de précision plus élevés. Cette observation souligne l'importance des techniques d'augmentation de données pour améliorer les performances des réseaux de neurones dans des tâches de classification.*

TABLE 2 – Résultats de précision en utilisant CNN et MLP sur les ensemble de données d'entraînements. **Sans-aug** : Sans augmentation ;

Setting	**CNN complexe**	**MLP complexe**
Sans-aug	83%/75%	70%/62%
Ts	89.8%/78.6%	72%/63.1%
Ps	93.8%/83.2%	85%/69%
Ts+Ps	99.75%/86.75%	86%/72%

TABLE 3 – Performance Comparison using Adam Optimizer on Different Batch Sizes

Model Type	Batch 8	Batch 16	Batch 32	Batch 64
Simple MLP	10% / 23%	10% / 23%	34% / 45%	25% / 35%
Complex MLP	83.36% / 71.25%	83.05% / 71.25%	83.83% / 72.50%	86.25% / 71.25%
Simple CNN	29.45% / 27.50%	87.27% / 60.00%	10.16% / 10.00%	91.88% / 72.50%
Complex CNN	99.61% / 83.75%	99.61% / 83.75%	99.69% / 86.25%	99.84% / 83.75%

TABLE 4 – Performance Comparison using SGD Optimizer on Different Batch Sizes

Model Type	Batch 8	Batch 16	Batch 32	Batch 64
Simple MLP	10% / 20%	10% / 20%	11% / 21%	10% / 20%
Complex MLP	72.11% / 63.75%	76.17% / 70.00%	75.47% / 67.50%	72.89% / 66.25%
Simple CNN	72.03% / 56.25%	59.92% / 47.50%	77.11% / 60.00%	72.81% / 57.50%
Complex CNN	84.45% / 78.75%	85.86% / 78.75%	83.52% / 77.50%	85.00% / 80.00%

Références

[1] Author, A.N. : Impact of dataset size and variety on the effectiveness of deep learning and transfer learning for plant disease classification. Journal of Plant Disease Prevention (2017)

[2] Cho, J., Lee, K., Shin, E., Choy, G., Do, S. : How much data is needed to train a medical image deep learning system to achieve necessary high accuracy ? Journal of Radiology (2015)

[3] Ioffe, S., Szegedy, C. : Batch normalization : Accelerating deep network training by reducing internal covariate shift. arXiv preprint arXiv :1502.03167 (2015)

[4] Liashchynskyi, P., Liashchynskyi, P. : Grid search, random search, genetic algorithm : A big comparison for nas. arXiv preprint arXiv :2004.07961 (2020)

[5] Motamedi, M., Sakharnykh, N., Kaldewey, T. : A data-centric approach for training deep neural networks with less data. Journal of Deep Learning (2017)

[6] Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., Lerer, A. : Automatic differentiation in pytorch. arXiv preprint arXiv :1502.05767 (2017)

[7] Piczak, K.J. : Esc : Dataset for environmental sound classification. Institute of Electronic Systems, Warsaw University of Technology (2015)

*[8] Salamon, J., Bello, J.P. : Deep convolutional neural networks and data augmentation for environmental sound classification. IEEE Signal Processing Letters **24**(3), 279–283 (2017). https ://doi.org/10.1109/LSP.2016.2641592*