Rapport TP M2 RFA

Quentin ROBLEZ-DUMAS[1] Sabine MUZELLEC[1]

[1] Université Paul Sabatier

quentin.roblez-dumas@univ-tlse3.fr
sabine.muzellec@univ-tlse3.fr

Résumé

Le but de ce rapport est de comparer les performances de deux types de réseaux de neurones profonds pour la classification de fichiers audios. Les entrées sont converties en spectrogramme, une représentation temps/fréquence du signal qui peut être vue comme une image. L'objectif est d'associer le spectogramme d'entrée à son bon concept.

Mots Clef

Classification audio, CNN, MLP, Apprentissage Profond

Abstract

The aim of this report is to compare the performance of two types of deep neural networks for audio files classification. The inputs are converted to a frequency chart, a time / frequency representation of the signal that can be viewed as an image. The objective is to associate the input spectogram with its correct concept.

Keywords

Audio Classification, CNN, MLP, Deep Learning

1 Introduction

La classification d'image est une problématique majeure du domaine de l'apprentissage automatique.
A l'heure d'aujourd'hui, les modèles de type CNN (Convolutionnal Neural Network) se sont avérés les plus performants dans l'accomplissement de cette tâche [2].
Afin de nous conforter dans cette conclusion, nous avons essayé de comparer les performances d'un CNN avec celles d'un MLP (Multilayer Perceptron).

Les CNN [1] donnent un sens aux images grâce à un mécanisme composé de filtres et de couches de pooling.
Un filtre est une matrice de nombres randomisés. Dans un CNN, les filtres sont multipliés par rapport aux représentations matricielles de bouts d'une image, ce qui permet de la balayer efficacement pixel par pixel et d'obtenir la valeur moyenne de tous les pixels adjacents, pour ainsi détecter les caractéristiques les plus importantes.
Un MLP [3] est une classe de réseau neuronal artificiel à feedforward. Un MLP se compose d'au moins trois couches de noeuds : une couche d'entrée, une couche cachée et une couche de sortie. À l'exception des noeuds d'entrée, chaque noeud est un neurone qui utilise une fonction d'activation non linéaire. Un MLP utilise une technique d'apprentissage supervisé appelée rétropropagation pour l'apprentissage. Ses multiples couches et son activation non linéaire distinguent un MLP d'un perceptron linéaire. Il peut distinguer les données qui ne sont pas linéairement séparables.

1.1 Travaux antérieurs

Cet article fait suite aux 2 séances de TP du module reconnaissance de forme et apprentissage à travers lesquelles nous avons pu aborder succinctement les MLP et CNN. Plus précisément nous avons intégré un MLP afin de répondre à une problématique de classification d'image sur le dataset "Fashion MNIST" ainsi qu'un CNN visant à classifier des images de chat et de chien. Pour chacun de ces réseaux, nous avons jugé de leurs qualités en s'appuyant sur diverses métriques.

2 Description techiques

Le corpus est constitué de 10 concepts audios. Il est divisé en deux sous dossiers : le corpus d'entraînement contenant 320 fichiers et le corpus de test contenant 80 fichiers.
La transformation des fichiers audios en spectrogramme génère des images de taille 128*216.
Pour chaque modèle testé, nous evaluons ses performances sur deux métriques :

- En règle générale, avec les réseaux de neurones, nous cherchons à minimiser l'erreur. En tant que telle, la fonction objectif est souvent appelée fonction de coût ou fonction de perte et la valeur calculée par la fonction de perte est simplement appelée «loss».
- La précision (accuracy) est une métrique qui décrit généralement les performances du modèle dans toutes les classes. Elle est utile lorsque toutes les classes sont d'importance égale. Elle est calculée comme le rapport entre le nombre de prédictions correctes et le nombre total de prédictions.

Les performances sont illustrées par des graphiques indi-

quant l'évolution de la loss et de l'accuracy. Nous complétons ceci avec des matrices de confusion : résumé des résultats de prédictions sur un problème de classification. Les prédictions correctes et incorrectes sont mises en lumière et réparties par classe. Les résultats sont ainsi comparés avec les valeurs réelles.
Cette matrice permet de comprendre de quelle façon le modèle de classification est confus lorsqu'il effectue des prédictions. Ceci permet non seulement de savoir quelles sont les erreurs commises, mais surtout le type d'erreurs commises.
Durant les phases d'entraînement, nous pouvons jouer sur différents paramètres pour mesurer les performances :

- Une epoch est quand un ensemble de données entier est passé en avant et en arrière à travers le réseau neuronal seulement une seule fois.
- Le batch size est le nombre total d'exemples de formation présents dans un seul lot. C'est une division du dataset complet.

Afin de pouvoir reproduire les expérimentations proposées ci dessous et d'obtenir les mêmes résultats, nous avons fixé une seed de valeur : *1234*

3 MLP

3.1 Protocole expérimental

Afin de répondre à cette problématique de classification nous avons commencé par fixer certains paramètres que nous savons être adapté à ce genre de problème.
De cette façon, nous avons choisi d'utiliser la fonction de coût "cross-entropy" et la fonction d'activation, pour la dernière couche, "soft-max ", car ces dernières répondent à la classification multiclasses et notre système est constitué de 10 classes possibles en sortie.
On peut également établir, dans un même temps, que notre couche de sortie doit comprendre 10 neurones (un par classes).
En ce qui concerne les fonctions d'activation des couches cachées, les fonctions "sigmoid" et "ReLu" peuvent convenir avec une préférence pour la "ReLu" ayant des performances théoriques supérieures.
Pour l'architecture à proprement parlé, nous n'avons pas vraiment de connaissance à priori nous permettant de choisir une implémentation précise au détriment d'une autre. Nous avons donc testé de multiples possibilités. D'autre part nous avons procéder à un prétraitement des images envoyées en entrée de notre réseau. Nous avons donc effectué une normalisation ainsi qu'une standardisation,notre image est également mit sous la forme d'un vecteur (128*216), de nos images afin d'optimiser la précision de notre MLP.
Nous avons également opté pour une notation des labels sous la forme "one hot" approprié à une représentation multiclasse.

3.2 Résultats obtenus

N'ayant pas d'idée arrêtée sur l'architecture à adopter nous avons fait de très nombreux essai en modifiant de multiples paramètres tel que le nombre de couches cachées, le nombre de neurones de ces dernières, l'optimiseur, la taille des batchs, les fonctions d'activations. Nous ne sommes pas parvenus à déterminer une configuration obtenant des résultats significativement supérieur ceux d'une autre configuration.
Pour chacune de nos architectures, la précision évoluait entre 20 et 40 Nous présentons donc les résultats d'une de nos architectures ayant les paramètres suivant :

- Nombre de paramètres : 886,154
- Couche internes : 2 couches de 32 neurones
- Nombre d'epochs : 10
- Taille des batch : 32
- Optimizer : *adam*
- Fonctions d'activations internes : *ReLU*
- Fonction de sortie : *Softmax*
- Fonction de loss : *Cross Entropy*
- Random seed : 1234

La précision et la loss de notre modéle sur le jeu de test est alors la suivante : loss : 1.9729 - accuracy : 0.4000.
L'évolution des fonctions "loss" et "accuracy" durant l'entraînement est présentée sur la figure 1.

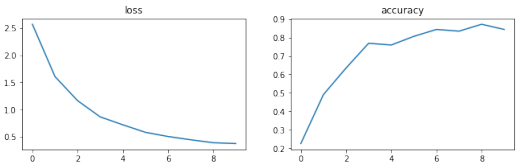

FIGURE 1 – Métriques pour notre MLP

Nous avons également réalisé la matrice de confusion de la figure 2.

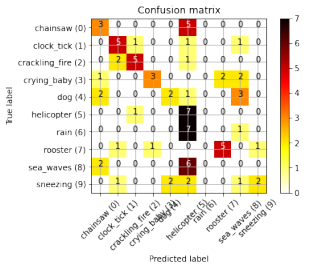

FIGURE 2 – Matrice de confusion de notre MLP

3.3 Interprétation

Nous avons pu constater à travers tous nos tests et au vu de la précision obtenue que l'architecture MLP est loin de convenir pour une tache de classification d'image.
En s'attardant sur la matrice de confusion, on constate que notre réseau à de grandes difficultés avec les bruits de pluie et de vagues qui classifie à tort en tant que bruit d'hélicoptère.

4 CNN

4.1 Protocole expérimental

Le premier modèle testé est un modèle relativement simple constitué de trois blocs suivi d'une couche dense. Chaque bloc contient une couche de convolution 2D (respectivement de taille 16, 32 et 64 et des filtres 3*3) suivie d'une fonction d'activation (relu) et une couche de MaxPooling 2D. La dernière couche dense, contenant 10 neurones est précédée d'une couche de flatten par soucis de comptabilité de taille de données.Les résultats obtenus sont visibles sur les graphiques 3 et 4. Ce modèle contient 247,306 paramètres.
En raison du nombre restreint d'image comprises dans le dataset d'entraînement, nous avons procédé à une augmentation des données pour essayer d'améliorer les performances de ce même modèle. Les résultats sont observables sur les graphiques 5 et 6.
Par la suite, notre travail s'est concentré sur les architectures testées. Les deux modèles présentées viennent de l'article [4].
Le modèle est assez commun : plusieurs couches convolutives sont suivies d'une couche de regroupement. Il contient 1,368,010 paramètres et composé de 3 blocs. Cependant, ces derniers sont plus complexes que les précédents : 3 couches de convolutions avant le MaxPooling 2D. Ces blocs sont suivies d'une couche de regroupement global.
Voici un bref aperçu du fonctionnement de cette couche. La dernière couche convolutive Conv2D (10, (1, 1)) produit 10 cartes d'entités correspondant à dix classes de sortie. Ensuite, la couche GlobalAveragePooling2D () calcule la moyenne spatiale de ces 10 cartes d'entités, ce qui signifie que sa sortie est juste un vecteur d'une longueur de 10. Une chose importante à noter : il n'y a pas de fonction d'activation appliquée à la sortie de la couche finale Conv2D (10, (1, 1)), car la sortie de cette couche doit d'abord passer par GlobalAveragePooling2D ().
Le prochain CNN, ALL-CNN-C, vient du même papier [4]. Ce modèle est très similaire au précédent. En réalité, la seule différence est que les couches convolutives avec une foulée de 2 sont utilisées à la place des couches de pooling max. Encore une fois, aucune fonction d'activation n'est utilisée immédiatement après la couche Conv2D (10, (1, 1)). Le modèle échouera à s'entraîner si une activation ReLU est utilisée immédiatement après cette couche.

Toutes les experimentations ont été effectuées avec les paramètres suivants :

- Nombre d'epochs : 10
- Taille des batch : 32
- Optimizer : *nadam*
- Fonctions d'activations internes : *ReLU*
- Fonction de sortie : *Softmax*
- Fonction de loss : *Cross Entropy*

4.2 Résultats obtenus

Les graphiques d'évolution des courbes sont donnés sur la figure 3 Pour compléter ces résultats, nous avons généré la

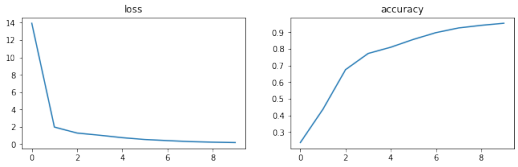

FIGURE 3 – Métriques pour le premier modèle

matrice de confusion associée aux prédictions (cf image 4)

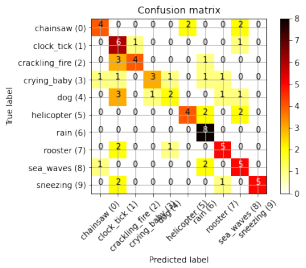

FIGURE 4 – Matrice de confusion pour le premier modèle

Les résultats suite à l'augmentation des données sont observables sur les graphiques 5 et 6.
Les résultats des expérimentations sur les architectures

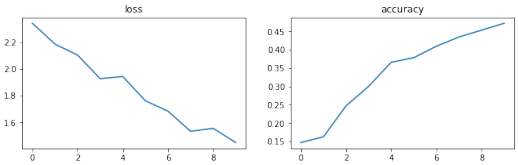

FIGURE 5 – Métriques pour le premier modèle suite à l'augmentation de données

présentées précédemment sont respectivement visibles sur les graphiques 7, 8 Les performances de chaque sont résumées dans le tableau 1

4.3 Interprétation

L'analyse du tableau 1 nous permet de mettre en évidence le fait que le premier modèle, malgré sa simplicité est le plus prerformant selon les deux métriques choisies. L'augmentation de données n'a pas révélé de resultats concluants. Nous supposons que la modification de spectrogramme fausse l'information contenue. Il serait donc plus judicieux d'augmenter les données en modifiant les sons directement plutôt que leur transformation en image.
Plus le modèle est complexe, plus l'entraînement est long.
Pour tous les entraînements, nous observons un comportement nominal pour l'évolution de la loss et de l'accuracy :

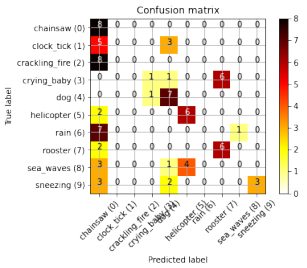

FIGURE 6 – Matrice de confusion pour le premier modèle suite à l'augmentation de données

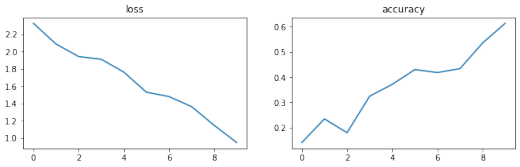

FIGURE 7 – Métriques pour le modèle ConvPool-CNN-C

la première baisse tandis que la seconde augmente.
L'analyse des matrices de confusion met en évidence les classes 8 et 9 qui semblent un peu plus dures à identifier.
Les deux derniers modèles testés sont relativement équivalents. Celui sans couche de pooling est légèrement meilleur. Cela est certainement du à la nature des données d'entrées.
Pour conclure sur cette partie, plus le modèle est simple (peu de paramètres), plus celui ci est rapide à entraîner. Dans notre cas, il est également plus performant. Nous observons un léger phénomène de spécialisation : la précision à la fin de l'entraînement est plus élévée que celle obtenue à l'évaluation.

Conclusion

À travers cet article nous avons mit en évidence le fait qu'un MLP est moins performant qu'un CNN en ce qui concerne la classification de sons et d'images.
On constate que même avec moins de paramètres un CNN reste plus performant qu'un MLP.
Cela pourrait s'expliquer par des problèmes de changement d'échelle ou de translation que rencontre naturellement un MLP lors de classification d'image.
Malgré le fait que notre CNN soit plus performant que notre MLP, celui-ci n'a pas des performances comparable à celui entraîné sur ImageNet [**2**]. Cela peut s'expliquer par le fait que dans notre problème il s'agit de classification de spectrogrammes et donc de sons. C'est pourquoi afin d'obtenir de meilleurs résultats pour cette tache nous recommandons l'utilisation de RNN (recurrent neural network).

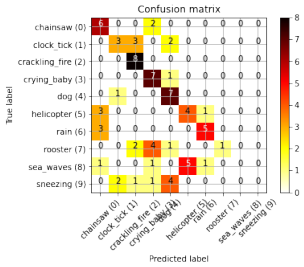

FIGURE 8 – Matrice de confusion pour le modèle ConvPool-CNN-C

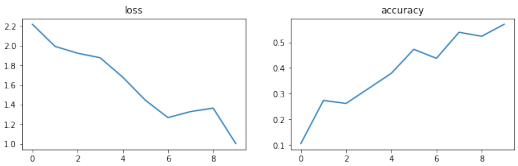

FIGURE 9 – Métriques pour le modèle All-CNN

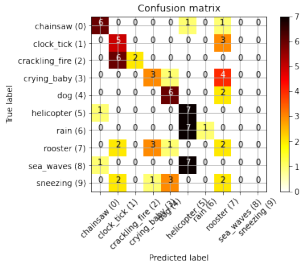

FIGURE 10 – Matrice de confusion pour le modèle All-CNN

	Modèle 1	DA	CNN1	CNN2
Time/epoch	13ms	70ms	377ms	188ms
Loss	2.1090	12.7566	4.1519	4.1398
Test Accuracy	0.6625	0.3875	0.5125	0.4000
Accuracy en début d'entrainement	0.23	0.15	0.0625	0.1250
Accuracy en fin d'entrainement	0.95	0.50	0.4648	0.5938

TABLE 1 – Performances des modèles : DA = Data Augmentation, CNN1 = ConvPool-CNN, CNN2 = All CNN

Références

[1] S. HERSHEY et al. “CNN architectures for large-scale audio classification”. In : *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. ISSN : 2379-190X. Mar. 2017, p. 131-135. DOI : 10.1109/ICASSP.2017.7952132.

[2] Alex KRIZHEVSKY, Ilya SUTSKEVER et Geoffrey E HINTON. “ImageNet Classification with Deep Convolutional Neural Networks”. In : *Advances in Neural Information Processing Systems 25*. Sous la dir. de F. PEREIRA et al. Curran Associates, Inc., 2012, p. 1097-1105. URL : http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf (visité le 22/10/2020).

[3] D. W. RUCK et al. “The multilayer perceptron as an approximation to a Bayes optimal discriminant function”. In : *IEEE Transactions on Neural Networks* 1.4 (déc. 1990). Conference Name : IEEE Transactions on Neural Networks, p. 296-298. ISSN : 1941-0093. DOI : 10.1109/72.80266.

[4] Jost Tobias SPRINGENBERG et al. “Striving for Simplicity : The All Convolutional Net”. In : *arXiv :1412.6806 [cs]* (avr. 2015). arXiv : 1412.6806. URL : http://arxiv.org/abs/1412.6806 (visité le 22/10/2020).