Rapport PLDAC - Commande vocale

Réalisé par
Sarah ENG
Zhile ZHANG
Etudiantes en M1 Informatique - parcours DAC à Sorbonne Université

Date
Soutenance orale
le *16 mai 2024*

Enseignant encadrant : Olivier Schwander

Table des matières

1 Introduction

1.1 Contexte

La classification audio est un des problèmes classiques de l'apprentissage automatique, et est notamment derrière la reconnaissance vocale. Concrètement, un des cas d'application est les voitures autonomes où la classification audio est utilisée de diverses façons. En effet, elle peut évidemment reconnaître les commandes vocales des passagers, pour définir la navigation vers une destination spécifique ou bien pour ajuster les paramètres de la voiture tels que la température, mais elle joue également un rôle crucial dans certaines situations atypiques, aussi bien dans la détection de signaux externes (sirènes des véhicules prioritaires, klaxons, piétons) que de signaux internes (bruits anormaux dans le fonctionnement de la voiture, sons à l'intérieur de la voiture pour estimer le confort des passagers).

Durant ce projet, nous allons nous pencher sur le cas le plus classique de classification audio, en étudiant quels traitements et quels modèles semblent être adaptés à la classification de courtes directives audio.

1.2 Données

Tout le long du projet, les données que nous utilisons proviennent du dataset Google Speech Commands V2. Ce jeu de données est uniquement composé de 105829 fichiers audio ne durant qu'une seconde, et comporte 35 labels différents : "Yes", "No", "Up", "Down", "Left", "Right", "On", "Off", "Stop", "Go", "Zero", "One", "Two", "Three", "Four", "Five", "Six", "Seven", "Eight", "Nine", "Bed", "Bird", "Cat", "Dog", "Happy", "House", "Marvin", "Sheila", "Tree", "Wow", "Backward", "Forward", "Follow", "Learn", "Visual". Par soucis de reproductibilité, les échantillons utilisés pour la validation et le test sont déjà définis.

Les données et les découpages sont accessibles par la librairie de Python :

$$torchaudio.datasets.SPEECHCOMMANDS$$

1.3 Documentations

Nous avons, tout d'abord, lu divers articles autour de ce sujet, pour nous introduire aux études qui ont déjà été réalisées, et pour découvrir quelles techniques peuvent être utilisées pour répondre à ce problème.

Certains documents proposent un nouveau modèle, comme Audio Spectrogram Transformer (AST) [Gong et al., 2021] qui est basé uniquement sur l'attention, ou bien MatchboxNet [Majumdar et Ginsburg, 2020] qui est un réseau de neurones complexe dont la particularité est de réduire le nombre d'hyperparamètres tout en conservant une bonne précision et robustesse. D'autres, étudient des aspects et techniques particulières pour améliorer la classification audio, comme la détection de mots-clés (Keyword Spotting KWS) [Zhang et al., 2018] qui est souvent utilisée pour la reconnaissance vocale notamment sur les appareils ayant peu de mémoire et de capacité de calcul comme les téléphones, ou encore la création d'une bibliothèque permettant de convertir des modèles en mode non continu, en modèles en mode continu [Rybakov et al., 2020]. En effet, les modèles en mode non continu reçoivent toute la séquence d'entrée puis renvoie le résultat de classification alors que ceux en mode continu ne reçoivent qu'une partie de la séquence d'entrée et la classe de manière progressive, ce qui est plus approprié dans certains cas de classification audio où la séquence est infinie.

La seule particularité que tous les documents partagent est le fait qu'ils justifient la performance de leur approche en la testant sur différents modèles de l'état de l'art.

La plupart d'entre eux réalisent leurs tests en utilisant les Google Speech Commands Datasets, et évaluent les performances en fonction de la précision obtenue. Lors des prétraitements des données, ils transforment l'audio en MFCC (Mel-frequency cepstral coefficients) et augmentent le volume des données : le Time Shift et les masques font partie des techniques les plus communes dans ces études. Enfin, dans la majorité des documents, on retrouve parmi les modèles utilisés en tant que comparaisons, des réseaux de neurones classiques comme des DNNs, CNNs, et RNNs.

2 Expériences

2.1 Protocole expérimental

Voici le protocole expérimental par défaut sur nos données audio :

- **Collecte des données** : On récupère les données du dataset Google Speech Commands V2.

- **Séparation des ensembles de données** : On sépare les données en 3 ensembles pour l'entraînement, la validation et le test du modèle, selon les découpages fournis.

- **Prétraitement des données** : Pour chaque fichier audio, on récupère le signal, on ajuste la taille du signal pour qu'il dure bien une seconde (c'est une précaution prise pour s'assurer que tous les exemples aient la même taille), et on récupère les MFCC correspondants.

- **Augmentation des données** : Cette étape est facultative si on ne souhaite entraîner le modèle que sur les données originales. Sinon, on applique les techniques choisies d'augmentation des données sur tous les exemples, ce qui crée de nouveaux exemples qui sont ajoutés à l'ensemble des données d'entraînement.

- **Entraînement et validation du modèle** : On entraîne le modèle sur les données d'entraînement et on utilise les données de validation pour ajuster le modèle.

- **Evaluation du modèle** : Pour évaluer la performance du modèle, la métrique qu'on choisit est la précision.

2.2 Augmentations des données

Les techniques d'augmentation de données permettent d'augmenter le volume de données utilisées pour l'apprentissage, avec de légères différences entre elles. Cette étape devrait améliorer l'accuracy en permettant au modèle de mieux déterminer les caractéristiques discriminantes, et de rendre plus robuste le modèle en l'empêchant de sur-apprendre. Voici les quelques techniques que nous avons étudiées dans ce projet :

- l'ajout de **bruit blanc ("White noise")**, qui est un son continu homogène, c'est-à-dire qu'il y a la même intensité pour chaque fréquence.

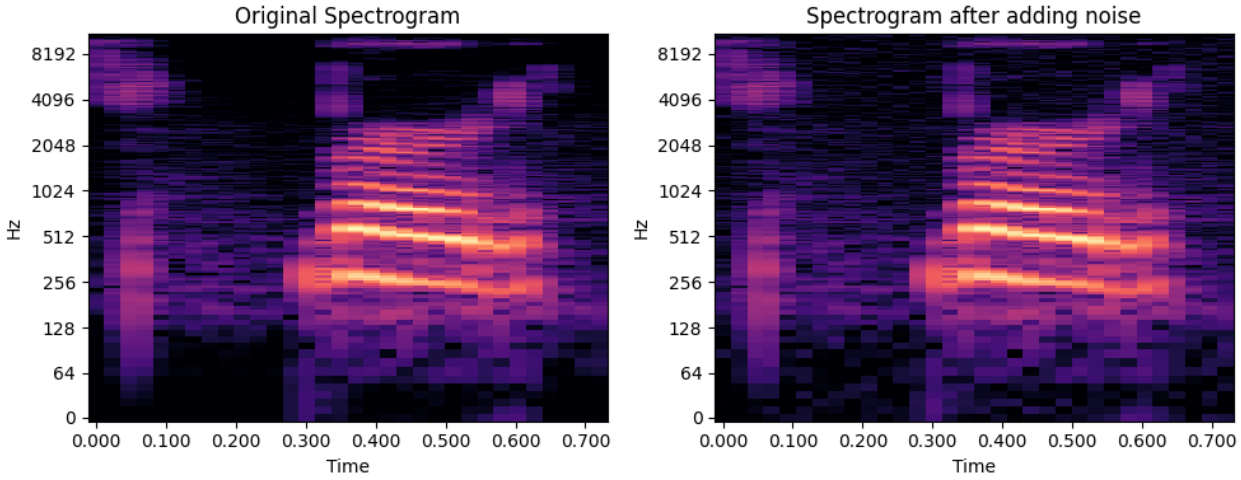


Figure 1 – Noise

- un **décalage temporel**, qu'on appelle **"Time Shift"**, permet de représenter des échantillons dans lequel l'emplacement de la commande est différent (elle peut être au début, au milieu, ou à la fin). Ce décalage effectue un roulement, c'est-à-dire que la partie du signal qui sort de la durée de l'échantillon à cause du décalage vers la droite, est réinjectée au début de l'audio.

- l'utilisation de **masques temporelle et fréquentiel**, plus connu sous les noms **"Time Mask" et "Frequency Mask"**, permet de symboliser des problèmes techniques. Par exemple, le Time Mask masque un instant dans l'échantillon, ce qui peut arriver en réalité si l'équipement a eu un problème et n'a rien pu capter pendant un moment, tandis

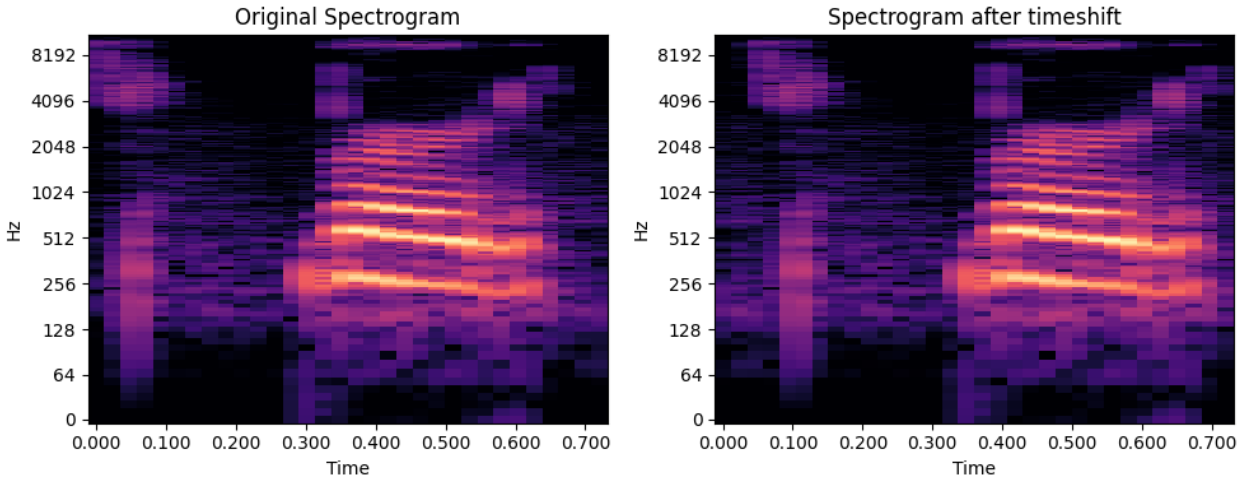


Figure 2 – Time Shift

que le Frequency Mask masque une fréquence durant toute la durée de l'échantillon, ce qui montre le cas où un équipement n'est pas adapté ou défectieux et n'arrive pas à capter certaines fréquences d'un signal.

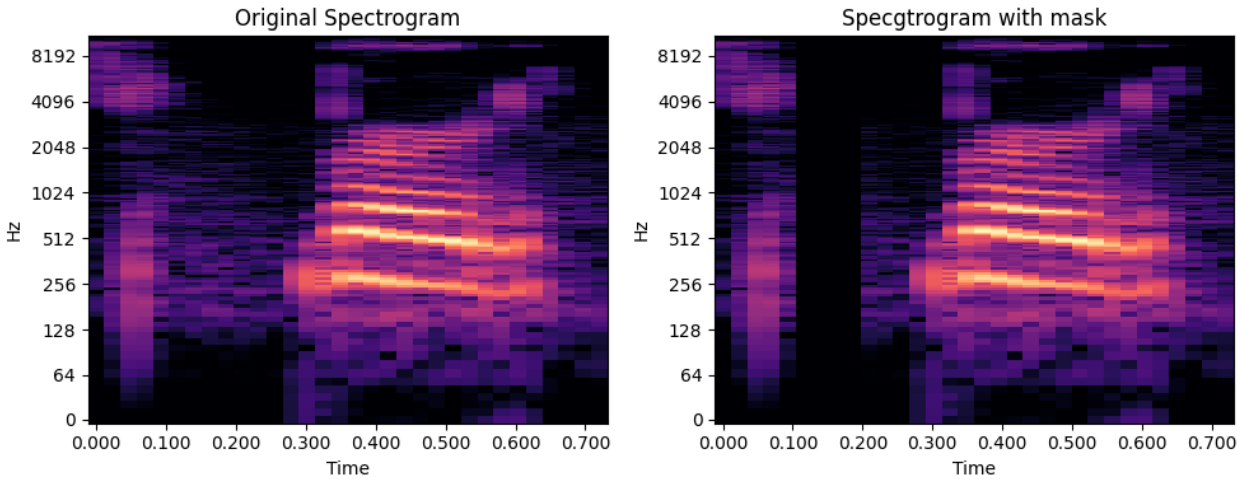


Figure 3 – Mask

- le **"Pitch Shift"** permet de changer la fréquence du signal, ce qui est, concrètement, un changement de voix.

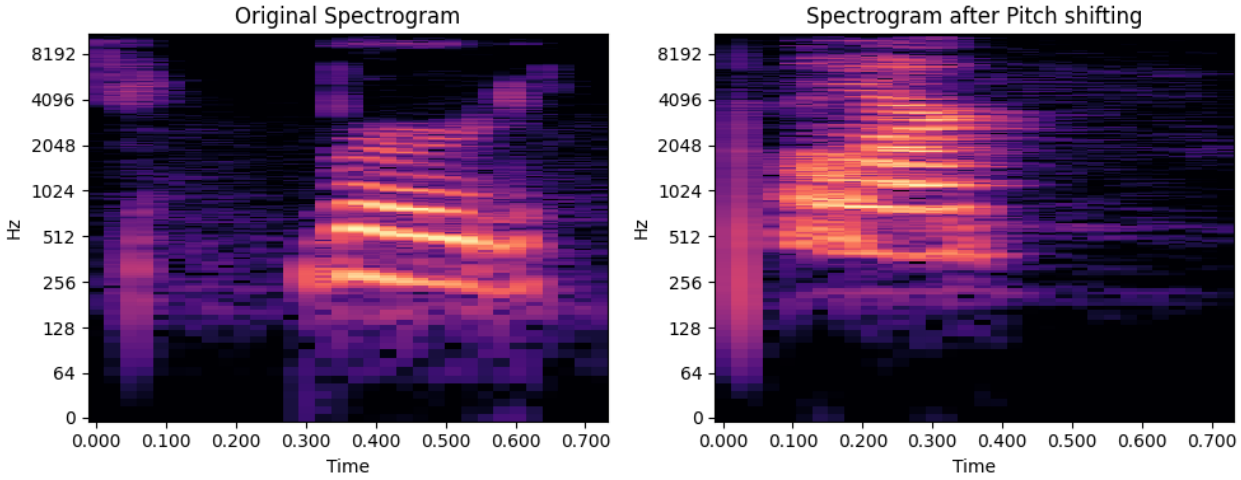


Figure 4 – Pitch Shift

- un **changement de vitesse ("Speed Change")** modifie l'audio pour que la commande soit énoncée plus rapidement ou plus lentement.

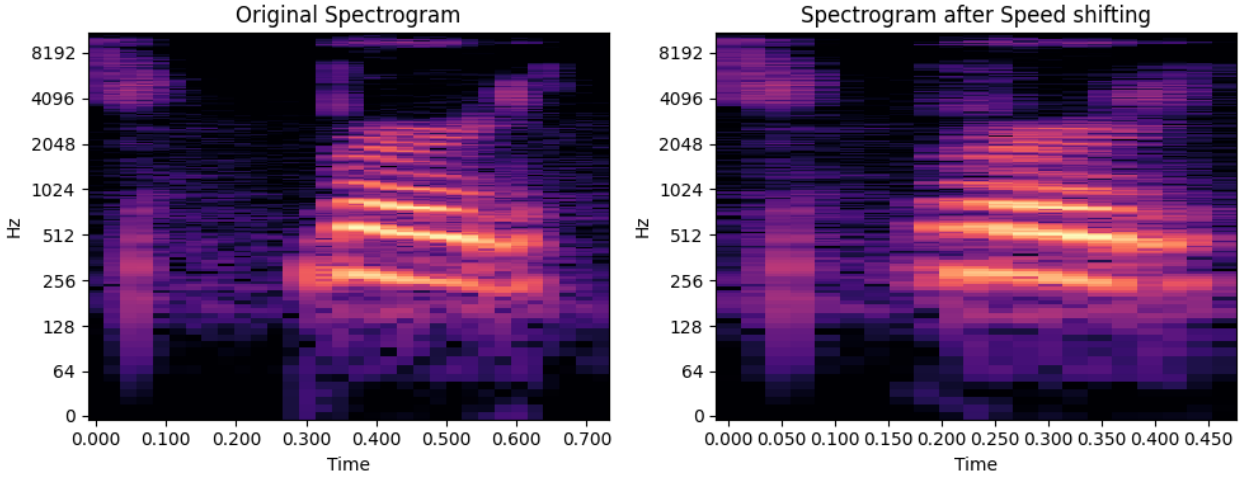


Figure 5 – Speed Change

2.3 Modèles

Dans cette étude, nous avons développé un modèle de réseau de neurones convolutif (CNN), un modèle combinant un CNN et un réseau de neurones à longue et courte mémoire bi-directionnel (Bi-LSTM) et un modèle simplement Multi-Layer Perceptron (MLP) pour la classification audio.

2.3.1 MLP

Le modèle MLP est un réseau de neurones entièrement connecté simple mais efficace, voici une explication détaillée du modèle :

Couches entièrement connectées (Fully Connected Layers) :

— **Première couche (fc1) :** La couche d'entrée, qui reçoit des données avec 338 caractéristiques et les traite à travers 500 neurones. Cette couche vise à capturer les relations linéaires et les caractéristiques initiales des données entrantes.

— **Deuxième couche (fc2) :** La couche de sortie, qui mappe les sorties de la première couche vers 35 nœuds de sortie, chaque nœud correspondant à une étiquette de classe.

Fonctions d'activation :

— **Première fonction d'activation :** ReLU

— **Deuxième fonction d'activation :** Softmax

2.3.2 CNN

Pour le modèle CNN, grâce à ses couches convolutionnelles, le CNN peut extraire automatiquement des caractéristiques importantes des données vocales, telles que le rythme, la tonalité et l'intensité du son, en utilisant des données transformées en coefficients cepstraux de fréquences de Mel (MFCC). Voici une explication détaillée de l'architecture du modèle :

Architecture du Modèle CNN

Couche d'entrée :

— **Dimensions :** $(13, 26, 1)$

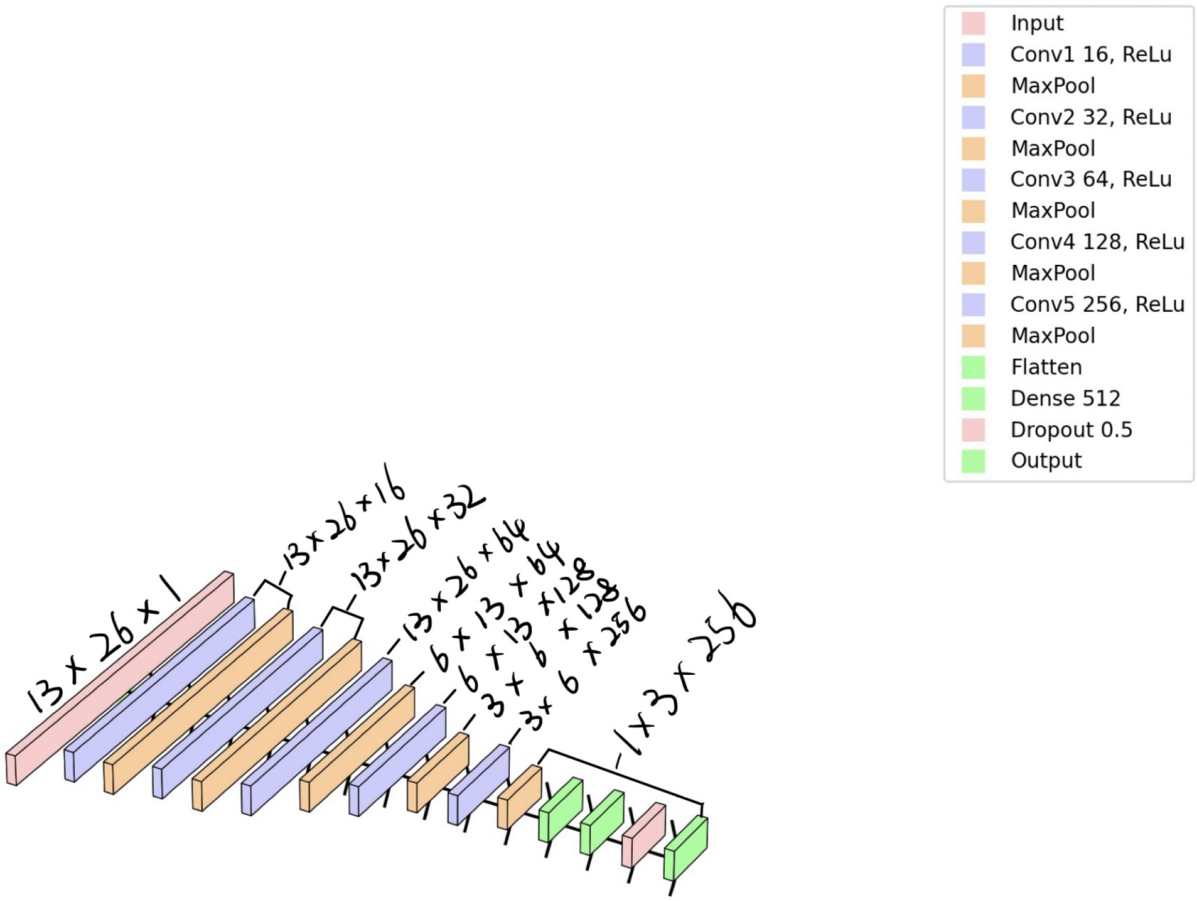


Figure 6 – L'architecture CNN

— **Description :** La couche d'entrée reçoit les données MFCC, où chaque échantillon a des dimensions de 13×26, représentant 13 coefficients MFCC pour chaque segment de série temporelle. Le format de ces données est semblable à une image en niveaux de gris, où 1 représente un canal monochrome.

Couches de Convolution et de Pooling :

1. **Première couche de convolution (Conv2D) :**
   — **Nombre de filtres :** 16
   — **Taille du noyau :** 3×3
   — **Fonction d'activation :** ReLU
   — **Padding :** 'same'
   — **Dimensions de sortie :** Conservées à $(13, 26, 16)$ car le padding 'same' et une stride de 1×1 sont utilisés pour garder les dimensions de sortie identiques à

celles de l'entrée.

— **Utilité :** Extraction des caractéristiques primaires des données MFCC, telles que les bords et les motifs texturaux simples.

2. **Première couche de pooling (MaxPooling2D) :**
   — **Taille du pooling :** 1×1
   — **Dimensions de sortie :** Maintenues à $(13, 26, 16)$. Cette couche de pooling ne réduit pas réellement les dimensions spatiales des données, mais peut aider à prévenir le surajustement.
3. **Deuxième couche de convolution (Conv2D) :**
   — **Nombre de filtres, Taille du noyau, Fonction d'activation :** 32, 3×3, ReLu.
   — **Dimensions de sortie :** Encore utilisées avec un padding 'same', les dimensions de sortie restent $(13, 26, 32)$.
   — **Utilité :** Extraction des caractéristiques plus complexes telles que les variations de rythme et de tonalité dans l'audio.
4. **Deuxième couche de pooling (MaxPooling2D) :**
   — **Taille du pooling :** 2×2
   — **Dimensions de sortie :** Comme la stride est généralement égale à la taille du pooling, les dimensions sont réduites de moitié à $(6, 13, 32)$.
   — **Utilité :** Réduction de la dimensionnalité des caractéristiques, augmentation de la capacité d'abstraction du modèle, et diminution de la complexité des calculs.
5. **Autres couches de convolution et de pooling .** Augmentation progressive du nombre de filtres (64, 128, 256), chaque couche suivie d'une opération de pooling 2×2, réduisant progressivement les dimensions spatiales des cartes de caractéristiques tout en augmentant leur profondeur et leur complexité.

Couches Entièrement Connectées et Couche de Sortie :

— **Flatten :** Aplatit la sortie de la dernière couche de pooling en un vecteur unidimensionnel pour servir d'entrée aux couches entièrement connectées.

— **Dense :** Première couche dense avec 512 unités, utilisant la fonction d'activation ReLU, vise à intégrer toutes les caractéristiques apprises par les couches convolutives

précédentes.

— **Dropout :** Élimine aléatoirement 50% des nœuds pour réduire le surajustement et améliorer la capacité de généralisation du modèle.

— **Couche de sortie :** Utilise la fonction d'activation softmax pour retourner la probabilité prévue pour chaque catégorie de classe (ici, 35 classes).

Compilation du Modèle :

— **Optimiseur :** Adam

— **Fonction de perte :** Entropie croisée catégorique

— **Métrique d'évaluation :** Précision (accuracy)

2.3.3 Bi-LSTM

Le modèle Bi-LSTM est spécialement conçu pour l'analyse de données séquentielles telles que l'audio ou les séries temporelles. Cette combinaison de CNN et de RNN permet de capturer efficacement les caractéristiques locales des données séquentielles (via les couches CNN) ainsi que les dépendances à longue distance (via les couches Bi-LSTM). Voici un schéma d'architecture et une explication détaillée du modèle :

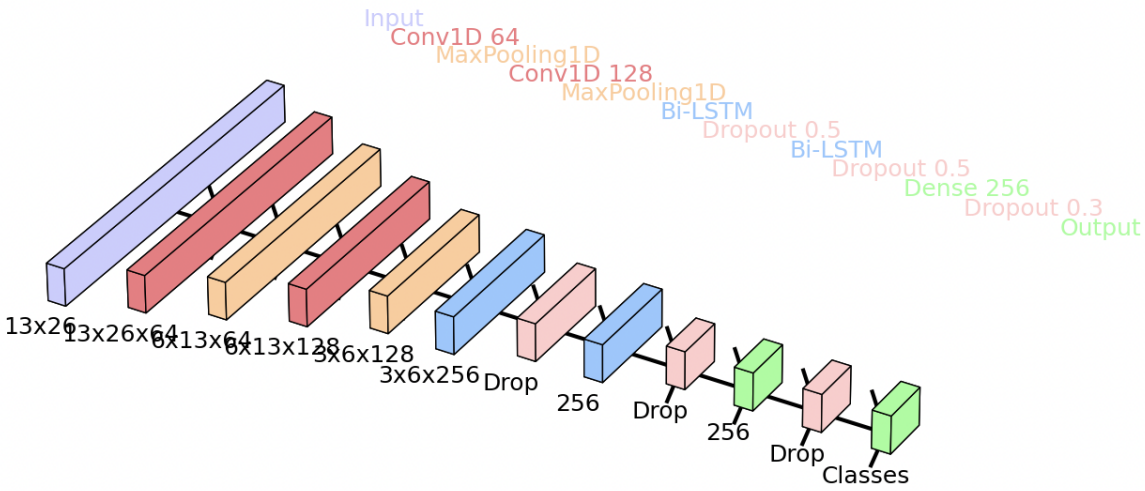


Figure 7 – L'architecture Bi-LSTM

Couche d'entrée :

— **Dimensions :** $(13, 26)$

— **Description :** La couche d'entrée reçoit les données MFCC, où chaque échantillon a des dimensions de 13×26.

Couches de convolution et de pooling $2\times$(Conv1D + MaxPooling1D) :

— **Nombre de filtres :** 64 et 128

— **Taille du noyau :** 5 et 3

— **Fonction d'activation :** ReLU

— **Padding :** 'same'

— **Taille du pooling :** 2×2

— **Dimensions de sortie :** les dimensions sont réduites de moitié à $(3, 6, 128)$

— **Utilité :** Couches de convolution visent à capturer les dépendances et caractéristiques locales dans les données d'entrée (telles que les bords, les tendances, etc.), aidant le modèle à extraire des informations utiles à partir des données brutes. Couches de pooling réduisent la dimensionnalité des données séquentielles, diminuant ainsi les calculs ultérieurs tout en aidant à extraire des caractéristiques plus robustes et à réduire le risque de surajustement.

Couches Bi-LSTM bidirectionnelles :

1. **Première couche Bi-LSTM :**
   — **Nombre d'unités :** 256
   — **Return sequences :** True, maintient l'intégrité de la séquence de sortie, fournissant une entrée de séquence complète à la couche Bi-LSTM suivante.
   — **Utilité :** Le Bi-LSTM traite les séquences d'entrée dans les deux directions, capturant ainsi les dépendances à la fois dans le sens positif et négatif du temps.
2. **Couche de Dropout :**
   — **Taux :** 0.5
3. **Deuxième couche Bi-LSTM :**
   — **Unité :** Ne retourne pas de séquences, sort directement la représentation finale des caractéristiques.

Couches entièrement connectées et couche de sortie.

3 Résultats

3.1 Performances et temps d'exécution

	CNN		Bi-LSTM		MLP	
	accuracy	**time**	**accuracy**	**time**	**accuracy**	**time**
Sans Augmentation	88.98%	34m	73.39%	22m	72.67%	2m
Avec Bruit	89.30%	62m	75.46%	44m	73.28%	3m
Avec TimeShift	90.04%	61m	75.41%	43m	74.88%	3m
Avec Mask	89.33%	72m	74.38%	44m	72.68%	3m
Avec PitchShift	89.12%	66m	75.05%	44m	73.54%	3m
Avec SpeedChange	89.16%	65m	74.63%	44m	71.95%	3m
Avec Mask et TimeShift	89.66%	88m	75.22%	75m	74.88%	3m
Avec Mask et PitchShift	89.87%	92m	76.25%	75m	72.98%	3m
Avec Bruit et SpeedChange	89.50%	82m	75.57%	66m	74.37%	3m
Avec All	89.59%	168m	77.92%	107m	75.62%	5m

Tableau 1 – Tableau des scores et temps(minute) pour les modèles CNN, Bi-LSTM et MLP

D'après ce tableau, nous pouvons constater les performances de base, c.à.d sans aucune augmentation de données, le CNN affiche les meilleures performances avec une précision de 88.98 %. Les précisions des modèles Bi-LSTM et MLP sont respectivement de 73.39 % et 72.67 %.

Pour les éffets des techniques d'augmentation des données, nous analysons et résumons chaque méthode d'augmentation de données individuellement :

1. **Ajout de bruit :** L'ajout de bruit a un impact positif sur la précision de tous les modèles, augmentant la précision du CNN à 89.30 %, celle du Bi-LSTM à 75.46 %, et celle du MLP à 73.28 %.
2. **Décalage temporel (TimeShift) :** Cette technique montre l'un des impacts les plus significatifs pour tous les modèles. La précision de chaque modèle s'est améliorée de près de 2%.
3. **Ajout de masquage :** Cette technique améliore principalement la précision du Bi-LSTM et CNN, atteignant 74.38% et 89.33 %. L'impact sur le MLP est minimal.
4. **Modification de la hauteur (PitchShift) :** Cette méthode a un léger effet positif

sur le CNN et le MLP.

5. **Changement de vitesse (SpeedChange) :** Cette technique augmente la précision du CNN à 89.16 %, mais a un léger impact négatif sur le MLP.
6. **Techniques d'augmentation combinées (Mask et TimeShift, Mask et PitchShift, Bruit et SpeedChange) :** Ces combinaisons offrent généralement une amélioration de précision plus stable. En particulier, la combinaison de masquage et de modification de la hauteur montre l'augmentation de précision la plus significative pour tous les modèles.
7. **Utilisation combinée de toutes les techniques (All) :** Lorsque toutes les techniques sont appliquées ensemble, le Bi-LSTM montre l'augmentation de précision la plus significative, atteignant 77.92 %.

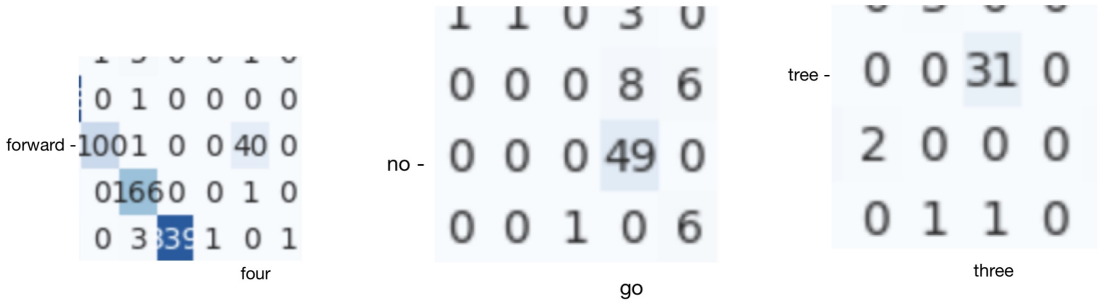
En comparant les effets des différentes techniques d'augmentation de données, nous constatons que la plupart d'entre elles améliorent la précision du modèle, ce qui confirme essentiellement nos hypothèses. En particulier, le décalage temporel et la combinaison de multiples techniques sont particulièrement efficaces. En termes de temps, nous observons que le modèle CNN nécessite beaucoup plus de temps que les deux autres modèles, mais le retour sur investissement est considérablement élevé, avec des taux de précision bien au-delà de ceux des autres modèles, certains atteignant même des taux de précision supérieurs à 90 %.

Il est important de mentionner que le modèle MLP très simple que nous avons créé, utilisant seulement deux couches de connexion, produit des résultats similaires à ceux du modèle Bi-LSTM plus complexe. De plus, le modèle MLP a utilisé beaucoup moins de temps que le modèle Bi-LSTM, ce qui suggère que notre approche pour l'architecture du modèle Bi-LSTM est probablement mauvaise. Les raisons de cela pourraient être la combinaison des couches CNN et Bi-LSTM n'a pas été très efficace, le nombre, la taille ou le type des couches CNN n'ont pas réussi à capturer suffisamment les caractéristiques locales essentielles dans les spectrogrammes. Par conséquent, les informations fournies aux couches Bi-LSTM pourraient ne pas être suffisantes pour un apprentissage efficace des séquences temporelles. De plus, le choix des paramètres pour les couches Bi-LSTM n'a pas été correctement adapté pour gérer les dépendances temporelles spécifiques des données audio.

Il existe également de nombreuses raisons pour lesquelles le modèle CNN a montré de meilleures performances. Par exemple, CNN peut efficacement apprendre les motifs locaux dans les spectrogrammes audio grâce à ses couches de convolution, tels que les variations d'intensité à des fréquences spécifiques, les formes du spectre, etc. De plus, l'empilement avec des couches de pooling permet d'extraire progressivement des caractéristiques audio plus complexes à partir de simples bords et textures, ce qui améliore considérablement la capacité du modèle à reconnaître les caractéristiques audio pertinentes.

3.2 Erreurs de classification

Ce qui est intéressant, c'est que nous pouvons identifier les catégories où les taux d'erreur sont les plus élevés en utilisant des matrices de confusion. Nous avons sélectionné les trois exemples suivants :



(a) forward-four-noise **(b)** no-go-timeshift **(c)** tree-three-speed

Figure 8 – Exemples de Mal classé

Ces figures représentent les catégories qui ont le plus d'erreurs dans les résultats des tests. Nous pouvons constater que ces erreurs sont effectivement difficiles à identifier. Par exemple, les mots « tree » et « three » qui se prononcent de manière très similaire, surtout après avoir modifié la vitesse de l'audio en accélérant le rythme, ces deux mots semblent presque identiques phonétiquement. De plus, « forward » et « four », où la prononciation de « four » correspond à la première partie de celle de « forward ». Ainsi, lorsque nous ajoutons du bruit blanc qui peut justement brouiller la partie finale de la prononciation de « forward », il devient facile de commettre des erreurs lors de l'entraînement du modèle.

3.3 Comparaison avec un modèle complexe

Table 7: *Comparing AST and SOTA models on ESC-50 and Speech Commands. "-S" and "-P" denotes model trained without and with additional audio data, respectively.*

	ESC-50	Speech Commands V2 (35 classes)
SOTA-S	86.5 [33]	97.4 [34]
SOTA-P	94.7 [7]	97.7 [35]
AST-S	88.7±0.7	**98.11±0.05**
AST-P	**95.6±0.4**	97.88±0.03

Figure 9 – L'évaluation du modèle AST sur le jeu de données Speech Commands V2[Gong et al., 2021]

Nous pouvons comparer les résultats de test de notre meilleur modèle avec ceux d'un modèle mentionné dans l'un des articles que nous avons lus, prenons par exemple un modèle basé uniquement sur des mécanismes d'attention, car dans cet article, le jeu de données utilisé ainsi que les découpage sont les mêmes. Nous pouvons constater que sans surprise, notre modèle est surpassé. Pour notre modèle, il s'agit encore seulement d'un modèle CNN ou d'un modèle RNN de base et typique, sans même intégrer le concept de mécanisme d'attention. Et notre prétraitement des données et la pré-formation de notre modèles sont assez faciles. Cependant, nous sommes également heureux que la précision de notre modèle puisse fluctuer autour de 90 %, ce qui en termes de précision est un bon début.

4 Conclusion

La classification de courtes directives audio est un problème classique d'apprentissage automatique, dont les solutions sont utilisées par de nombreux objets connectés pour permettre une interaction avec les utilisateurs. Dans ce projet, nous avons étudié la performance de plusieurs réseaux de neurones de l'état de l'art (MLP, CNN, Bi-LSTM), ainsi que l'impact de différentes techniques d'augmentation des données. D'après nos analyses, le CNN a les meilleurs performances avec une précision aux alentours de 90% dans tous nos tests, et les techniques d'augmentation de données améliorent en général la précision des modèles. Les modèles Bi-LSTM et MLP ont des performances similaires, malgré le fait que le MLP est plus simple et plus rapide que le Bi-LSTM. De plus, malgré les performances remarquables du CNN, son temps d'exécution est le plus conséquent. Des erreurs de classification fréquentes entre certaines classes sont acceptables, car elles sont liées au fait que leur prononciation se ressemblent énormément, et qu'avec la modification de l'échantillon par les techniques d'augmentation de données, les mots sont difficilement distinguables. Enfin, le modèle AST est meilleur que notre CNN, mais notre modèle et protocole expérimental sont simples et ne prennent par en compte les mécanismes d'attention.

Références

Gong, Y., Chung, Y.-A., & Glass, J. (2021). AST : Audio Spectrogram Transformer. https://arxiv.org/abs/2104.01778

Majumdar, S., & Ginsburg, B. (2020). MatchboxNet : 1D Time-Channel Separable Convolutional Neural Network Architecture for Speech Commands Recognition. https://doi.org/10.21437/interspeech.2020-1058

Rybakov, O., Kononenko, N., Subrahmanya, N., Visontai, M., & Laurenzo, S. (2020). Streaming Keyword Spotting on Mobile Devices. https://doi.org/10.21437/interspeech.2020-1003

Zhang, Y., Suda, N., Lai, L., & Chandra, V. (2018). Hello Edge : Keyword Spotting on Microcontrollers. https://arxiv.org/abs/1711.07128