M2 Systèmes Embarqués et Traitement de l'Information (SETI)

UE : D2 – Reconnaissance et interaction vocale

2018 / 2019

RAPPORT DU PROJET : DEVELOPPEMENT D'UN SYSTEME NEURONAL DE DETECTION DE COMMANDES VOCALES

Réalisé par :

CHGHAF Mohammed

FAN Zhuzhi

MENG Huaizhong

TAKALI Mohamed Selmi

Encadré par :

M. BARRAS Claude

Table des matières

Table de figures

Tableaux

Introduction :

Un système de commande vocale est un système informatique doté d'une interface d'entrée qui permet d'exécuter des tâches suite à un message vocal de l'utilisateur. Ce système est muni d'un programme de reconnaissance vocale qui permet d'analyser les sons qu'il capte et décide de la commande à exécuter.

Ces systèmes sont devenus presque une nécessité dans la plupart des objets connectés du marché technologique et sont utilisés dans plusieurs domaines. De l'assistance virtuelle à l'apport d'aide aux personnes handicapées, la précision du système de commande vocale est un critère déterminant pour la réussite des produits commercialisés. Etant donné que cette précision est souvent coûteuse en termes de temps de calcul et d'énergie consommée, le problème est souvent réduit à la détection d'un seul mot clé qui va permettre de lancer le calcul pour détecter la suite de la commande sur un serveur plus performant.

Notre projet s'inscrit dans ce cadre et consiste à développer un système neuronal de détection de commandes vocales en utilisant la librairie LibROSA pour l'analyse du signal et la détermination des paramètres acoustiques. La libraire PyTorch d'apprentissage profond et un corpus de commandes vocales de plus de 2 gigabits.

Notre objectif final va être le développement d'un système performant et simple capable de détecter le maximum possible de mots clés figurant dans le dataset. L'objectif sera atteint en comparant différentes architectures en termes de complexité et de performance.

1. Prétraitement et extraction des caractéristiques du Dataset :

1.1. Prétraitement du Dataset :

Contrairement à la programmation traditionnelle, en apprentissage automatique, l'analyse des données fournies (Dataset) est une étape essentielle afin de choisir les hyperparamètres et l'algorithme adéquats. Dans cette partie nous avons étudié quelques caractéristiques du corpus qui nous a été fourni.

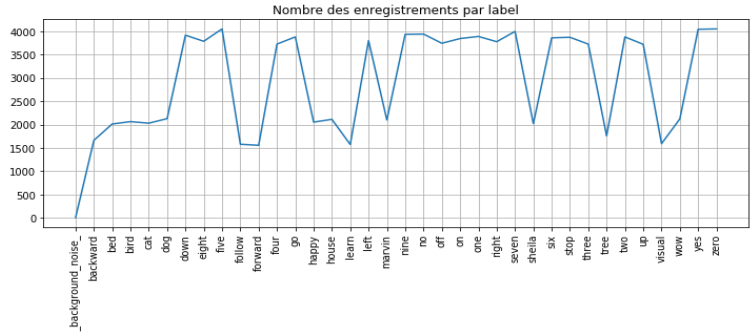

Figure 1: Nombre des enregistrements en fonction des labels.

Au total, la base d'exemples qui nous a été fournie est constituée de 105836 enregistrements répartis entre 35 labels. Pour chaque label le nombre d'enregistrements qui est fourni est souvent très différent comme ce qui est montré dans la Figure 1. On cite à titre d'exemple la différence entre le nombre d'enregistrements du label « backward » qui est de 1664 et du label « five » qui est de « 4052 ». On remarque par la suite, que ce nombre d'enregistrements qui diffère peut influencer fortement la précision du modèle vis-à-vis de certains labels.

En analysant davantage les enregistrements fournis on remarque qu'ils sont répartis selon différents lecteurs pour un même label. La Figure 2montre la forme de la transformation FFT de deux lecteurs pour le même label « backward ». Ainsi, et pour entrainer un modèle robuste, il faudra tenir en compte de cette remarque pour bien répartir le Dataset, entre base d'apprentissage, base de validation et base de test.

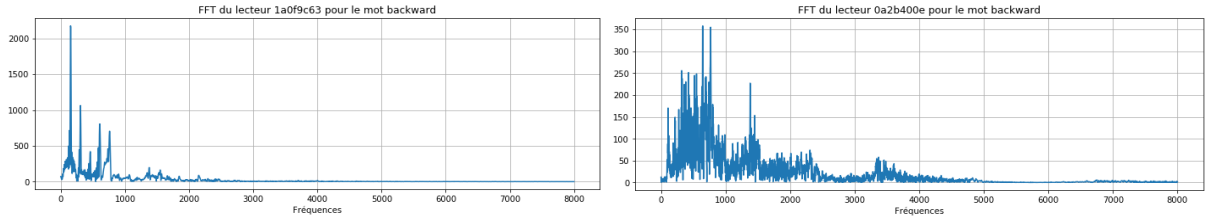

Figure 2: FFT de l'enregistrement de deux lecteurs prononçant le même label « backward »

Une analyse plus approfondie nous permet de remarquer qu'il existe des enregistrements où le mot n'est pas prononcé en entier ou que l'enregistrement n'a pas une durée de 1 seconde. Dans la figure 1.3, on illustre un cas où le mot « backward » n'est pas prononcé en entier et où l'enregistrement fait moins qu'une seconde. Le deuxième enregistrement a été arrêté avant que le lecteur ne prononce le mot et le troisième enregistrement représente le cas où le mot est bien prononcé dans l'enregistrement.

Nous avons noté qu'en total il y a 10435 enregistrements qui font moins d'une seconde et 95394 enregistrements qui font exactement 1s.

Par la suite nous tenons en compte ces remarques afin de compléter les enregistrements qui font moins d'une seconde par un bruit aléatoire. Nous augmentons la variabilité statistique de notre base d'apprentissage en ajoutant encore un pourcentage de bruit au signal obtenu à la fin. Les détails de

l'algorithme vont être présentés dans les parties qui suivent. La Figure 3 montre le résultat dans le domaine temporel après l'ajout de bruit aux enregistrements de la Figure 4.

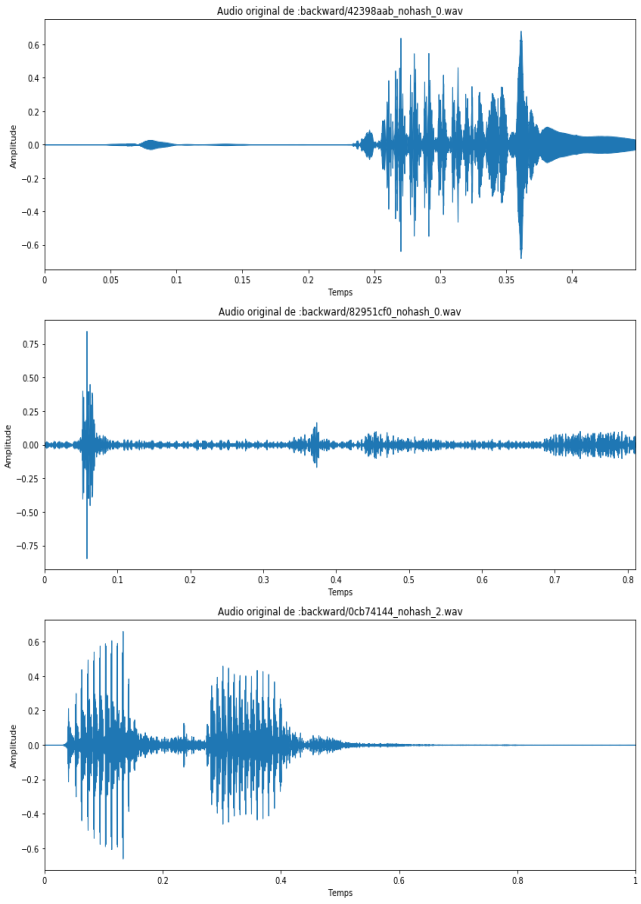

Figure 4: Forme de l'onde des différents enregistrements audio dans le domaine temporel (sans bruit)

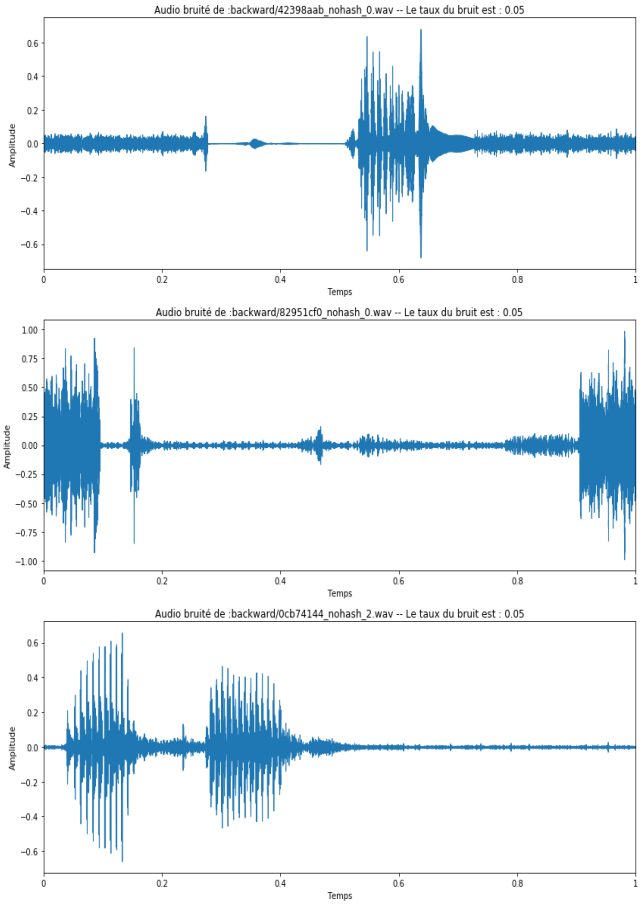

Figure 3: Forme de l'onde des différents enregistrements audio dans le domaine temporel (après l'ajout de bruit)

1.2. Extraction des caractéristiques :

Une première analyse nous a permis de constater que pour un même mot on peut avoir plusieurs prononciations différentes, l'algorithme qu'il faudra concevoir doit alors capturer les différentes caractéristiques d'un label sans tenir compte de la source du signal.

Nous allons alors découper le signal en plusieurs trames d'environ 20 à 30 ms toutes les 10 ms (Ce qui nous donne une fréquence d'échantillonnage de 16 KHz). Sur chacune de ces trames, on effectue une transformée de Fourier discrète permettant d'obtenir la répartition énergétique des fréquences qui composent le signal. En rassemblant ces spectres sur l'échelle du temps, on obtient le spectrogramme du signal. Dans notre exemple nous avons utilisé le logarithme des valeurs du spectrogramme.

Cette transformation permet de le rendre plus visible et plus proche de la façon avec laquelle l'oreille humaine entend les signaux. Elle permet également de deviner les mots prononcés à partir des formants et de la répartition du bruit.

Les caractéristiques acoustiques les plus couramment utilisées sont les Mel-Frequency Cepstral Coefficients (MFCC) notamment parce qu'ils prennent en compte la sensibilité de la perception humaine vis-à-vis des fréquences en décrivant de manière concise la forme générale de l'enveloppe spectrale et convient donc mieux à la reconnaissance des mots clés.

Nous représentons dans la Figure 5 une comparaison entre les deux labels « backward » et « five ». Nous comparons les deux enregistrements du point de vue temporel, puis du point de vue spectrogrammes et finalement nous comparons les deux filtres en échelle Mel et les coefficients MFCC obtenus.

Dans ce qui suit, nous allons utiliser les Mel-spectrogrammes puis les MFCC comme caractéristiques acoustiques pour entrainer notre système neuronal puis nous comparerons ses performances selon les hyperparamètres choisis.

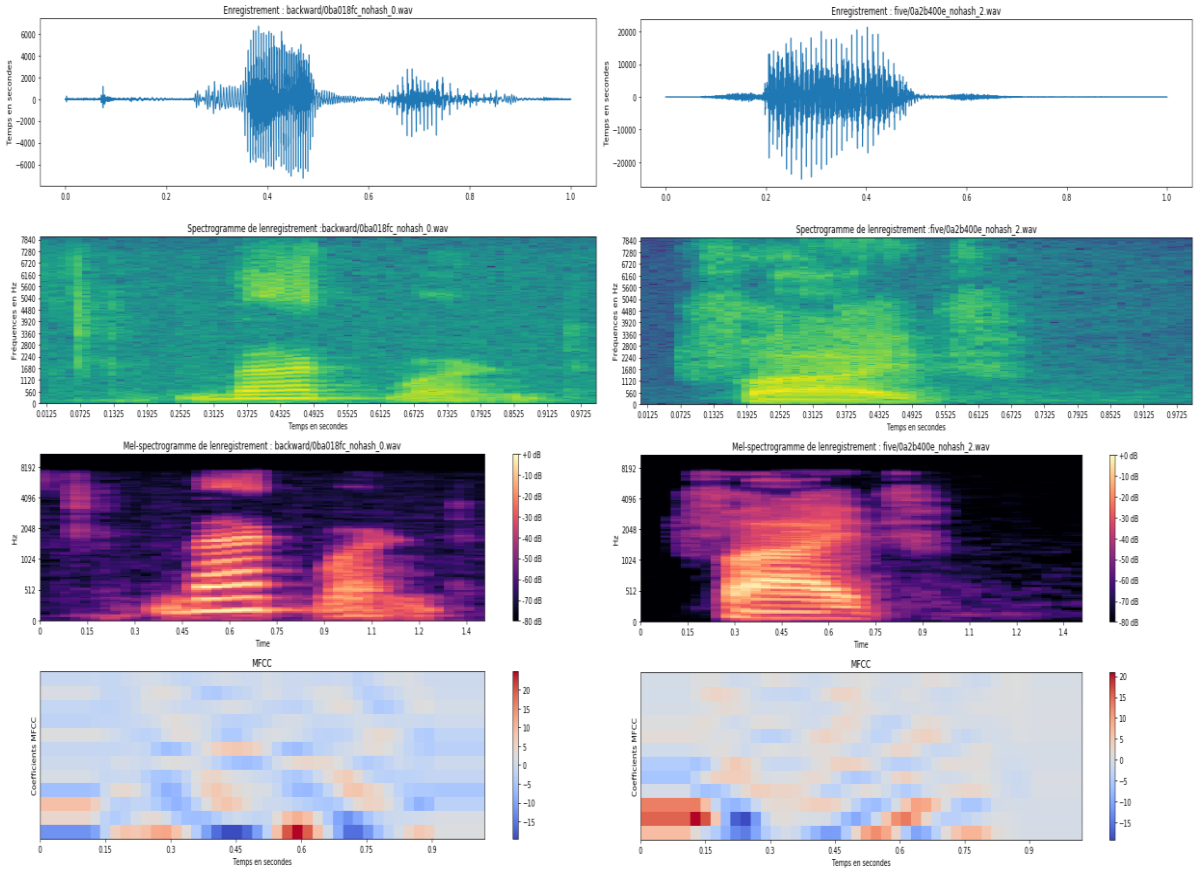

Figure 5: Comparaison entre la représentation du label « backward » et « five » : forme de l'onde dans le temps, spectrogrammes, échelle Mel et coefficients MFCC

1.3. Etapes de l'algorithme général du système et exemple de l'architecture du réseau de neurones utilisé :

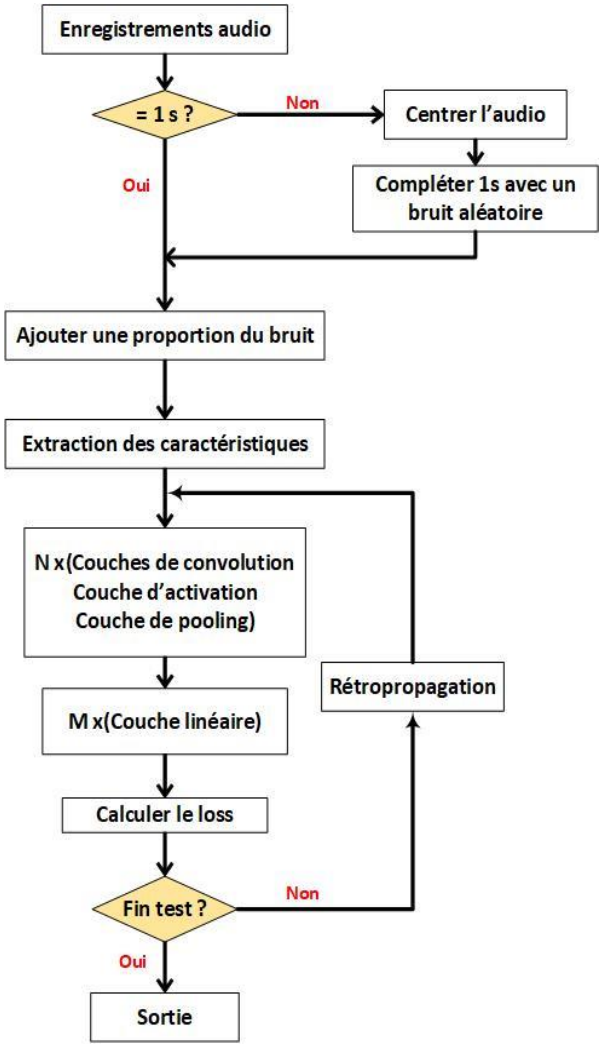

Figure 6: Diagramme de l'algorithme général utilisé

Comme pour les instruments de musique, les cordes vocales produisent un son composé d'une fréquence fondamentale et de plusieurs harmoniques.

Ce son est ensuite modulé par les différents organes : dents, lèvre, langue, palais, cavité nasale, pharynx, etc. en fonction de ce que l'on souhaite dire.

Le système à développer devra alors comprendre un processus d'extraction de caractéristiques pour détecter un message vocal sans prendre en compte la durée du message, sa source ou les déformations qui peuvent s'y ajouter.

Nous utiliserons par la suite un réseau neuronal convolutif pour traiter ces caractéristiques qui seront sous forme d'images 2-dimensions.

Pour cela, nous suivons les étapes décrits dans la Figure 6 en utilisant un réseau neuronal convolutif (CNN) compte tenu de leur efficacité en classification d'images.

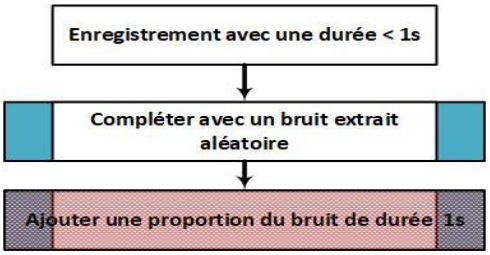

Figure 7: Processus de l'ajout du bruit.

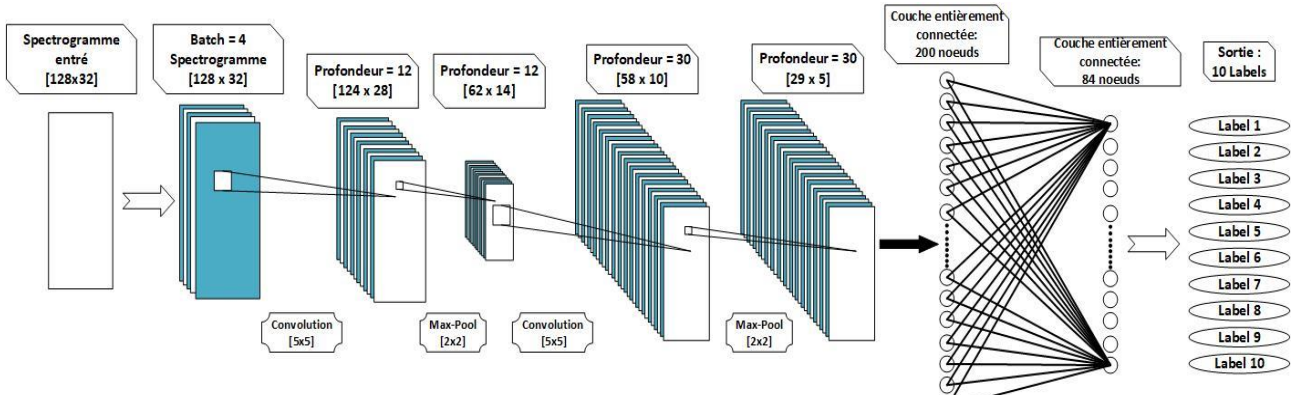

Figure 8: Exemple d'une des architectures des réseaux de neurones convolutifs utilisées.

2. Tests d'influence des différents paramètres sur la performance du système :

2.1. Effet du bruit sur la performance :

Dans un premier lieu nous avons regardé l'effet du bruit sur la précision de notre modèle avec 10% d'enregistrements en apprentissage, 5% en validation et 5% en test. Nous avons gardé un Learning_rate de 0.001 et un EPOCH de 10.

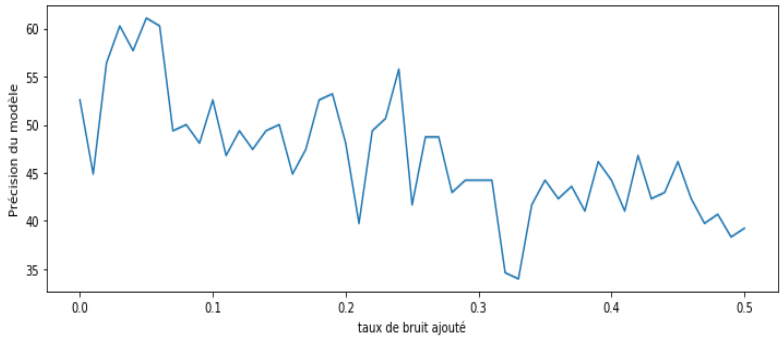

Figure 9: Evolution de la précision du modèle en fonction du taux du bruit fixé.

Après avoir effectué le test plusieurs fois, on a obtenu une précision du modèle maximale pour un bruit de 0.05 avec un taux de 61%. L'ajout de ce bruit nous permet d'augmenter la variabilité statistique mais aussi de remédier aux enregistrements où l'information n'est pas prononcée en complet ou n'est pas du tout prononcée. Nous avons gardé la valeur 0.05 en proportion de bruit pour la suite de nos tests.

2.2. Effet du Learning rate sur la performance :

Dans le domaine d'apprentissage automatique, l'algorithme d'optimisation le plus utilisé est celui du gradient stochastique. C'est cet algorithme qui a été utilisé pour mettre à jour des paramètres. Le taux d'apprentissage « Learning rate » a une influence importante sur la vitesse de convergence ainsi que le taux de précision. En général, un petit taux d'apprentissage diminue la vitesse de convergence mais augmente la précision du modèle.

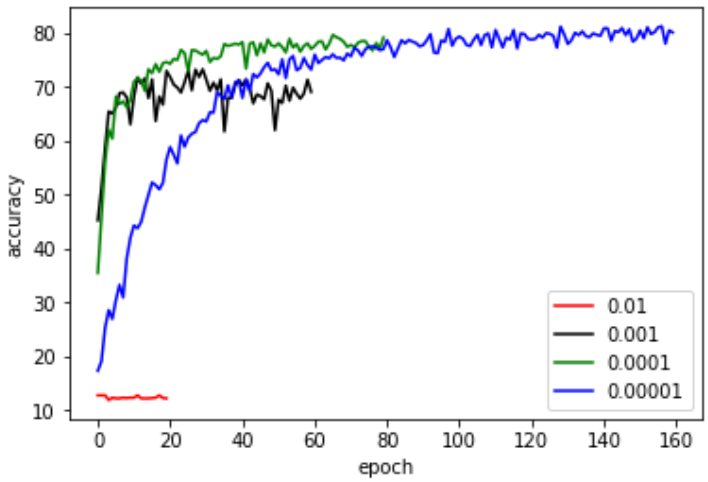

Figure 10: Evolution de la précision du modèle pour différents taux d'apprentissage « Learning rate » en fonction d'epochs

Nous avons effectué une série de tests pour trouver une valeur optimale du taux d'apprentissage. Un taux de 0,00001 permet d'obtenir les meilleurs résultats mais n'atteint une bonne précision qu'à partir de 100 répétitions « EPOCH ». Nous avons opté pour un taux d'apprentissage de 0,0001 qui parait un choix raisonnable en termes de performance et de vitesse de convergence.

2.3. Effet de la profondeur des couches CNN sur la performance :

De manière générale, le modèle peut apprendre davantage des informations que les couches CNN sont profondes (existence de plusieurs filtres). Cependant, si la base d'apprentissage n'est pas assez grande, alors que la profondeur des couches CNN est trop importante et que le nombre de répétitions est grand, le modèle devient « overfitting ». C'est-à-dire que le modèle va se concentrer sur le bruit ou sur des informations qui ne sont pas intéressantes.

On peut remarquer à partir de la Figure 11 que la profondeur des couches CNN n'influence pas beaucoup la précision du modèle.

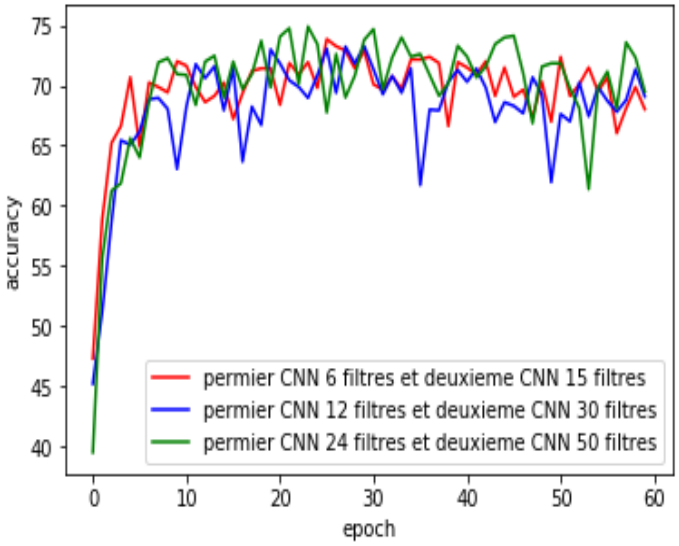

Figure 11: Evolution de la précision du modèle pour différentes profondeurs des couches CNN en fonction des répétitions « EPOCHs »

2.4. Influence du filtre Pooling sur la performance :

Dans cette partie, nous avons analysé l'effet de la taille du filtre Pooling sur la précision.

A partir des résultats obtenus, on remarque dans la Figure 12 que le taux de précision du modèle est plus stable pour des filtres de taille 3x3. D'autant plus, le filtre 3x3 atteint un meilleur taux de précision que celui du 5x5.

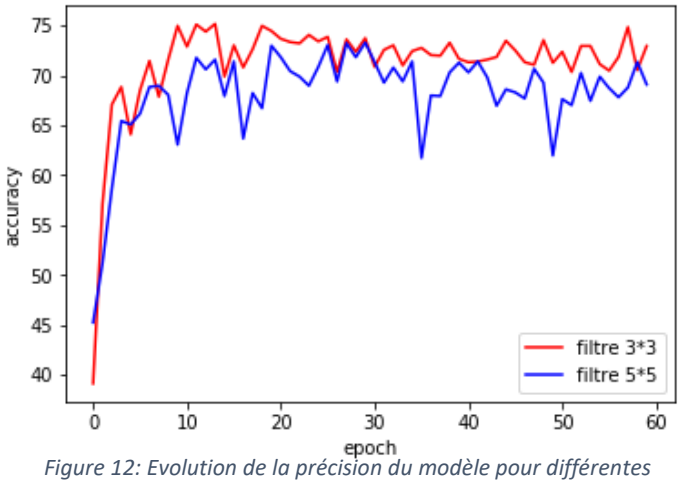

Figure 12: Evolution de la précision du modèle pour différentes dimensions des filtres Pooling en fonction des répétitions « EPOCHs »

2.5. Influence du pourcentage des données d'apprentissage sur la performance :

Dans cette partie on s'intéresse à l'influence du nombre des données entrées fournies au réseau pour l'apprentissage sur sa performance. Les données des entrées d'apprentissage, de validation et de test.

Pour nous assurer de la fiabilité du test, nous avons fixé les paramètres du système comme ce qui est présenté dans la Table 1.

Paramètre	Valeur
Nombre de labels	10
Proportion du bruit	0,2
Taille de batch	4
EPOCH	10
LEARNING_RATE	0,001
Couche CNN1	12x5x5
Couche CNN2	30x5x5
Couche linéaire 1	4350
Couche linéaire 2	200
Couche linéaire 3	84
Couche linéaire 4	10

Table 1 Paramètres fixés pour tester l'influence de proportion des données d'apprentissage

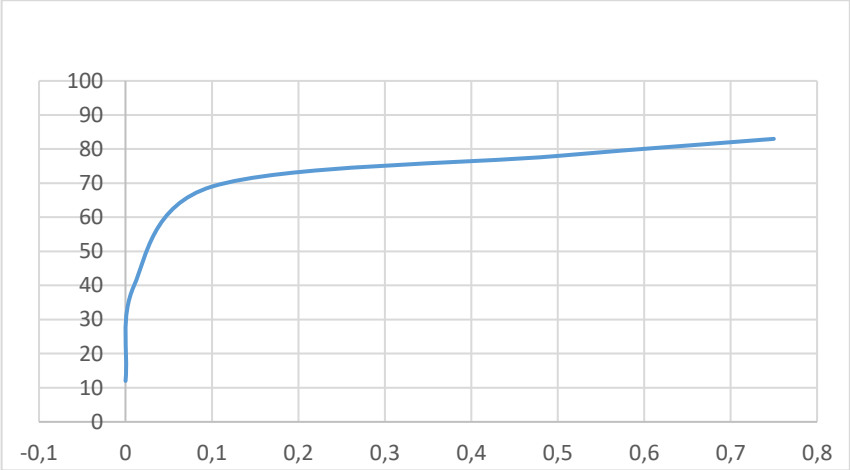

Figure 13 : Evolution du taux de reconnaissance en fonction des données fournies pour l'apprentissage

Pour un taux d'apprentissage, de validation et de test de 10 % on obtient une précision de 69%.

Avec un pourcentage de données d'apprentissage de 75%, de validation de 10% et de test de 10% on obtient une précision de 83%

2.6. Influence de la structure du réseau CNN sur la performance :

Pour nous assurer qu'uniquement la structure du réseau CNN qui va influencer les résultats, nous avons fixé les autres paramètres comme indiqué dans la Table 2. Ensuite, nous avons modifié le nombre des couches de convolution et des couches linéaires pour analyser leur influence sur les performances du modèle.

Paramètre	Valeur
Nombre de labels	10
Proportion de données d'apprentissage	0,1
Proportion de données du test	0,05
Taux de bruit	0,2
Taille du batch	4
Nombre d'epochs	15

Table 2 : Paramètres fixés pour tester l'influence de la structure du réseau CNN

En notant :

T_{n_i} : La taille de l'i-ème filtre de convolution

N_{n_i} : La profondeur de l'i-ème filtre de convolution

T_{l_i} : La taille de l'i-ème couche linéaire

Nous obtenus les résultats montrés dans la Table 3, avec les structures montrées dans la Table 4.

No. Test	Couches CNN	Couches linéaires	Taux de reconnaissance
1	1	4	0,59
2	1	5	0,63
3	1	6	0,64
4	2	4	0,68
5	2	5	0,67
6	2	6	0,75
7	3	4	0,73
8	3	5	0,77
9	3	6	0,75

Table 3: Taux de reconnaissance en fonction du nombre des couches CNN et des couches linéaires

No. Test	Nombre et tailles des couches de convolution						Nombre et taille des couches linéaires					
	T_{n_1}	N_{n_1}	T_{n_2}	N_{n_2}	T_{n_3}	N_{n_3}	T_{l_1}	T_{l_2}	T_{l_3}	T_{l_4}	T_{l_5}	T_{l_6}
1	5*5	12					4350	200	84	10		
2	5*5	12					4350	1000	100	50	10	
3	5*5	12					4350	1500	1000	100	50	10
4	5*5	12	5*5	30			4350	200	84	10		
5	5*5	12	5*5	30			4350	1000	100	50	10	
6	5*5	12	5*5	30			4350	1500	1000	100	50	10
7	5*5	12	5*5	30	2*2	60	1680	200	84	10		
8	5*5	12	5*5	30	2*2	60	1680	1000	100	50	10	
9	5*5	12	5*5	30	2*2	60	1680	1500	1000	100	50	10

Table 4: Informations détaillées sur les différentes couches utilisées dans chaque test

On remarque alors qu'en augmentant le nombre des couches de convolution on peut améliorer considérablement le taux de reconnaissance. Ceci est dû au fait que cette augmentation peut nous aider à mieux extraire des caractéristiques plus représentatives des enregistrements vocaux, et ainsi on arrive à distinguer les différents labels.

En outre, quand les caractéristiques extraites ne sont pas suffisantes, par exemple dans les tests numéro 1, 2 et 3, le nombre des couches linéaires n'a pas une grande influence sur le taux de reconnaissance.

En comparant les résultats de tests Numéro 6, 7, 8 et 9, quand le nombre de couches de convolution est plus ou égal à 2, l'augmentation du nombre de couches linéaires améliore la précision du modèle. C'est-à-dire qu'avec assez de caractéristiques, le taux de reconnaissance peut être amélioré en augmentant les calculs effectués, autrement dit, en augmentant le nombre des couches linéaires.

3. Analyse des performances d'autres modèles de système neuronal :

3.1. CNN avec extraction de caractéristiques MFCC

La MFCC (Mel Fréquence Cepstral Coefficients) est une extraction de caractéristique du signal développée autour de la FFT et DCT, ceci sur une échelle de Mel. Dans le MFCC, les bandes de fréquences sont équidistantes sur l'échelle Mel, ce qui se rapproche davantage de la réponse du système auditif humain que les bandes de fréquences à espacement linéaire utilisées dans le Cepstral normal. Cette déformation de fréquence peut permettre une meilleure représentation du son.

En considérant les tests effectués dans la partie 2.6, nous allons appliquer en entrée du système les caractéristiques MFCC extraites sans changer les autres paramètres. Nous nous baserons sur les tests numéro 1, 4 et 9, qui sont les plus représentatifs. La Table 4 montre les résultats obtenus.

No. Test	Nbre de couches convolution	Nbre de couches linéaire	Taux de reconnaissance	
			Sans MFCC	Avec MFCC
1	1	4	0,59	0,54
2	2	4	0,68	0,75
3	3	4	0,77	0,74

Table 5: Comparaison entre l'utilisation des spectrogrammes et des MFCC sur le taux de reconnaissance

A partir de ces résultats on déduit que : si le réseau utilisé ne pourra pas relever beaucoup de caractéristiques dans les entrées fournies au réseau (cas d'une seule couche de convolution), l'utilisation des caractéristiques MFCC ne va pas améliorer le modèle. Par contre si le modèle permet d'extraire suffisamment de « features » des entrées fournies (cas de deux couches de convolution), l'utilisation des caractéristiques MFCC apporte une nette amélioration au modèle utilisé. Mais si on sur-extrait des caractéristiques avec 3 couches de convolutions, à partir des caractéristiques MFCC déjà extraite, il y au risque de perdre des informations importantes qui pourraient influencer le taux de reconnaissance.

3.2. Utilisation des réseaux neuronaux profonds : DNN

Dans ce modèle de réseau de neurones on n'utilise que des couches entièrement connectées (pas de couches de convolution). Cette méthode nous permet de mieux conserver les informations originales dans les sons. Cependant, sans l'étape d'extraction des « features » (par convolution) et la réduction de dimension par couche « pooling », on a souvent besoin d'une puissance de calcul énorme.

Dans ce test, nous avons utilisé un DNN avec 7 couches linéaires (5 couches linéaires cachées), le nombre des nœuds de chaque couche se trouve est détaillé dans la Table 6.

	Couches d'entrée	Couches cachés					Couche de sortie
Nbres des nœuds	4096	3500	3000	2500	2000	1000	10

Table 6: Nombre des nœuds utilisés en chacune des couches entièrement connectées

À cause des nombreux couches linéaires, le réseau prend énormément de temps pour l'apprentissage. Dans notre test, le loss de l'apprentissage commence à converger après 100 epochs. La Figure 14 montre l'évolution du loss. Le taux de reconnaissance ne dépasse pas 70%, qui est moins que la performance obtenue avec des CNN (en gardant les mêmes paramètres).

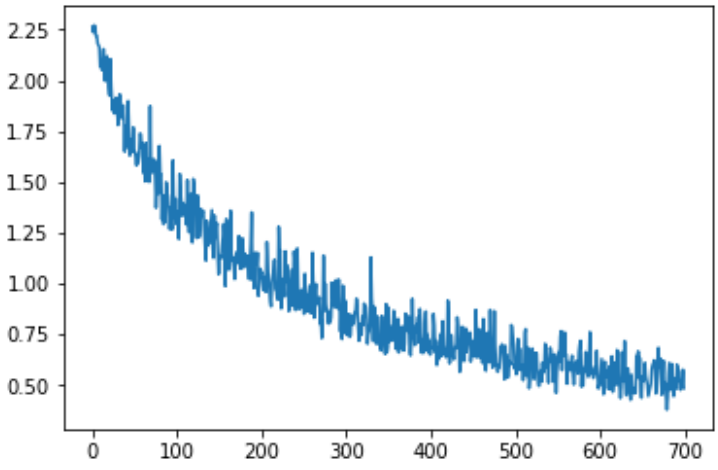

Figure 14: Evolution du loss en fonction des epochs

Ceci peut être interprété par le fait que sans extraction de caractéristiques par couche de convolution et sans extraction de caractéristiques du MFCC, le DNN a besoin plus de données d'apprentissage pour extraire des caractères par des itérations de l'algorithme du gradient stochastique. En considérant les temps de calcul très importants, nous n'avons pas augmenté davantage les couches du DNN.

3.3. Combinaison des réseaux de neurones convolutifs et des réseaux de neurones récurrents : CNN + RNN

Un réseau de neurones récurrent est constitué d'unités (neurones) interconnectés interagissant non-linéairement et pour lequel il existe au moins un cycle dans la structure. Les unités sont reliées par des arcs (synapses) qui possèdent un poids. La sortie d'un neurone est une combinaison non linéaire de ses entrées. Il nous permet donc de considérer des informations précédentes qui auraient des influences sur l'information instantanée. Avec RNN, on peut considérer des informations globales de chaque mot mais pas uniquement des informations locales présentés dans les spectrogrammes.

Dans notre modèle, avant d'entrer des données au réseau RNN, les enregistrements sont traités au préalable par le réseau CNN pour extraire des caractéristiques principales « features ». Dans la partie du réseau RNN, nous supposons que les caractères internes d'un mot ne sont influencés que par les caractères de la période précédente. On n'a employé qu'une couche cachée qui a pour rôle d'enregistrer les informations de la période précédente. Avant de sortir, une couche de softmax est utilisée pour calculer la probabilité d'appartenance d'un enregistrement à un label. La Figure 15 montre la structure utilisée, le tableau Table 7 montre le détail des différentes couches CNN et RNN utilisées.

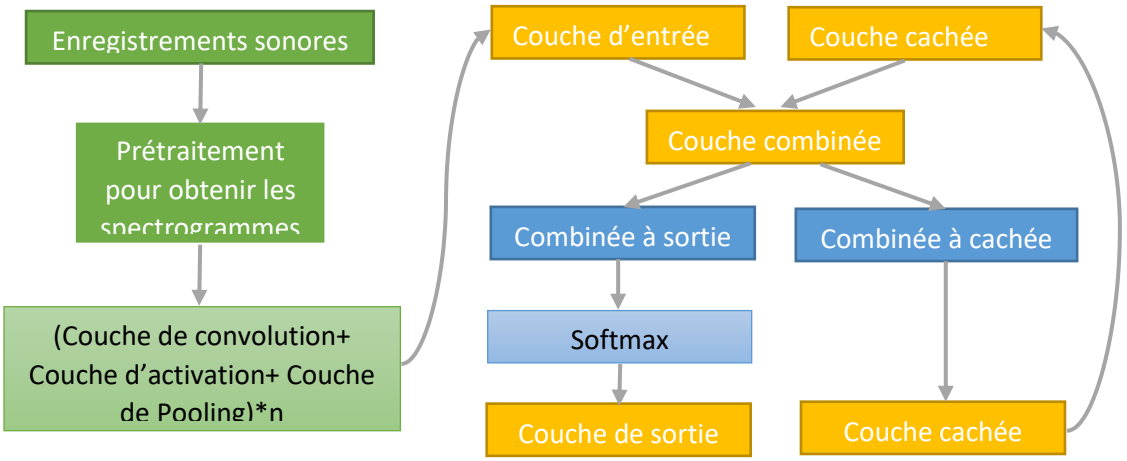

Figure 15: Architecture du réseau de neurones utilisé

Étape	Taille d'entrée	Opération	Taille de noyaux	Nombre de noyaux	Taille de sortie
1	1*128*32	Convolution	5*5	12	12*124*28
2	12*124*124	Activation			12*124*28
3	12*124*124	Pooling	2*2		12*62*14
4	12*62*62	Convolution	5*5	30	30*58*10
5	30*58*10	Activation			30*58*10
6	30*58*10	Pooling	2*2		30*29*5
7	30*29*5	Convolution	2*2	60	60*14*2
8	60*14*2	Activation			60*14*2
9	60*14*2	Convolution	14*1	120	120*1*2
10	120*1*2	Activation			120*1*2
11	120*1*2	Combiné avec couche caché			240*1*2

Table 7: Informations détaillées sur les couches CNN et RNN utilisées

Dans notre test, le loss de l'apprentissage commence à converger à partir de 5 epochs mais reste très élevé. La courbe du loss en fonction des epochs est montrée dans la Figure 16. Le taux de reconnaissance ne dépasse pas 12%, qui est une valeur très inférieure à celle obtenue pour le réseau CNN seul avec les mêmes paramètres utilisés pour l'apprentissage (un pourcentage de 10% pour l'ensemble d'apprentissage).

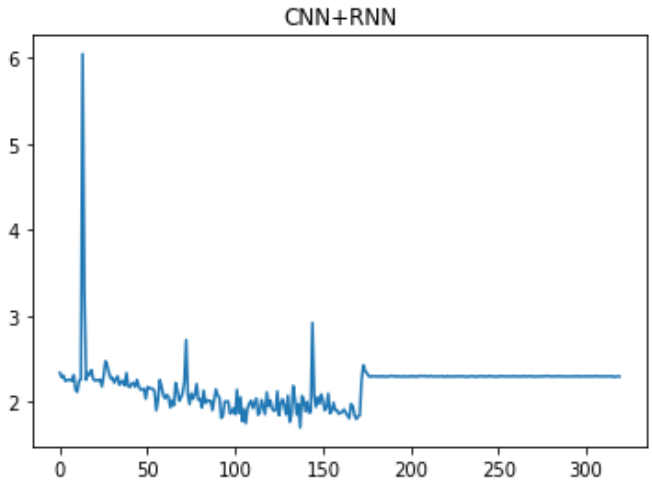

Figure 16 : Evolution du loss en fonction d'epochs pour l'architecture CNN + RNN

Ceci peut être interprété par le fait que les architectures RNN sont plutôt efficaces en traitement des phrases plus qu'en traitement des mots indépendants. Les informations fournies par le réseau CNN à l'entrée du réseau RNN ne sont pas suffisantes pour classifier correctement les labels sans l'utilisation d'un codage exact des informations, par exemple un encodage one-hot. Cette architecture ne sera pas utilisée par la suite de notre projet car l'erreur causée par le CNN ne pourra qu'augmenter en se propageant dans le réseau RNN.

4. Améliorations du modèle choisi :

4.1. Fixer le nombre des couches du CNN à 3 :

À partir des résultats des tests effectués, nous avons pu déduire des paramètres qui vont maximiser la performance du modèle. On cite à titre d'exemple le learning rate 0.0001, la taille du noyau 3×3, la proportion du bruit ajouté 0.05, une structure de 3 couches de CNN avec 3 couches de poolings et 4 couches entièrement connectées (après chaque couche entièrement connectée on ajoute un ReLU en fonction d'activation) et une base d'apprentissage de 60% du dataset initial, 20% pour la validation et 20% pour le test. Nous utilisons une matrice d'entrée (spectrogramme) de 32×128. Nous obtenons une précision totale de 90,5 % comme montré dans la Figure 17.

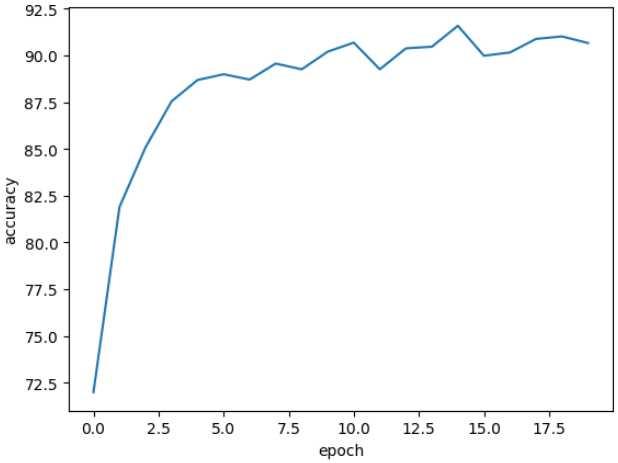

Figure 17: Evolution de la précision du modèle en fonction d'epochs

Précision	Les mots
85% - 89%	″stop″ ″seven″ ″bed″
90% - 94%	″three″ ″up″ ″sheila″ ″visual″ ″one″ ″forward″ ″down″

Table 8: Différents taux de reconnaissance pour chacun des 10 labels sélectionnés

Dans la suite, nous avons essayé de supprimer 2 couches de pooling. C'est-à-dire 3 couches CNN avec 1 pooling (dans premier CNN). La précision totale obtenue est de 88,7 % comme montré dans la Figure 18.

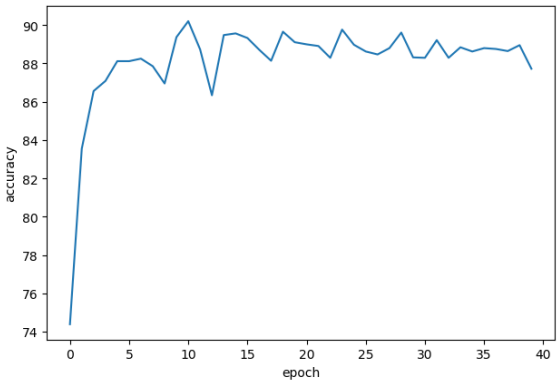

Figure 18: Evolution de la précision du modèle avec 1 pooling en fonction d'epochs

Précision	Les mots
75% - 79%	˝bed˝ ˝seven˝
85% - 89%	˝three˝ ˝visual˝ ˝down˝ ˝stop˝
90% - 94%	˝sheila˝ ˝up˝ ˝one˝ ˝forward˝

Table 9: Différents taux de reconnaissance pour chacun des 10 labels sélectionnés pour le modèle avec 1 pooling

4.2. Augmenter le nombre des couches CNN à 4 :

Nous allons ajouter une autre couche CNN. Maintenant, la première couche du CNN contient 12 noyaux de 3×3, la deuxième couche de CNN contient 30 noyaux de 3×3, la troisième couche de CNN contient 60 noyaux de 3×3, la quatrième couche de CNN contient 80 noyaux de 3×3.

La précision totale de 4 couches de CNN avec 1 pooling est 89,8 %. La précision totale de 4 couches de CNN avec 2 poolings est 91,6%.

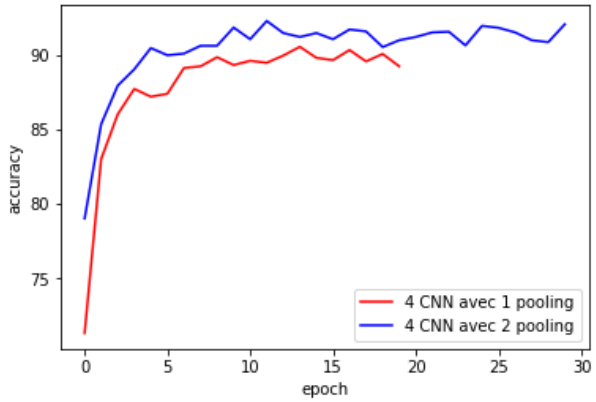

Figure 19: Evolution de la précision du modèle 4 couches du CNN

Précision	4 CNN 1 pooling	4 CNN 2 pooling
80% - 84%	″stop″ ″bed″	NaN
85% - 89%	″three″ ″forward″ ″sheila″ ″seven″	″stop″ ″sheila″ ″visual″ ″bed″
90% - 94%	″one″ ″down″ ″visual″	″forward″ ″three″ ″down″ ″up″ ″seven″
95% - 99%	″up″	″one″

Table 10: Différents taux de reconnaissance pour chacun des 10 labels sélectionnés pour le modèle 4 couches CNN (dans le cas 1 pooling et le cas 2 poolings)

4.3. Augmenter la dimension de la matrice d'entrée

Nous avons essayé d'augmenter la taille de la matrice d'entrée pour récupérer plus d'informations de l'audio, normalement le modèle pourra apprendre davantage. A l'origine on avait une matrice de taille 128×32, maintenant on considère une matrice de taille 128×64.

La précision totale avec 2 poolings est de 93,7%. Pour 3 poolings elle est de 94,3% et pour 4 poolings elle est de 94,4%

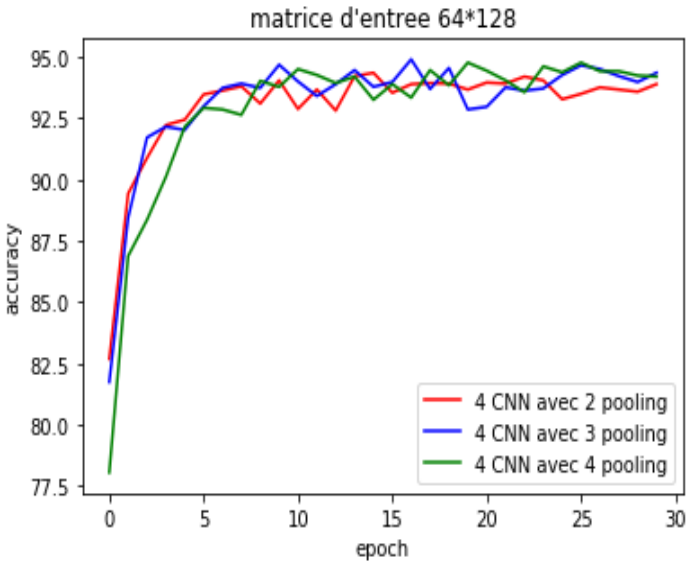

Figure 20: Evolution de la précision du modèle 4 couches du CNN pour une matrice de taille 128 x 64

Entre 3 et 4 poolings il n'y a pas une grande différence dans la performance globale, mais pour les 10 labels choisis le taux de reconnaissance est supérieur à 90% (contrairement à 4 poolings qui a 3 labels avec un taux de reconnaissance entre 85% et 90%) ceci veut dire que la variance pour le cas de 3 poolings est plus petite et que ce modèle est finalement le plus stable.

Précision	4 CNN 2 pooling	4 CNN 3 pooling	4 CNN 4 pooling
85% - 89%	″visual″	NaN	″visual″ ″forward″ ″stop″
90% - 94%	″three″ ″bed″ ″down″ ″one″ ″sheila″ ″stop″ ″seven″ ″forward″	″seven″ ″down″ ″bed″ ″stop″ ″visual″″forward″″three″	″sheila″ ″seven″ ″bed″ ″down″
95% - 99%	″up″	″sheila″ ″one″ ″up″	″three″ ″up″ ″one″

Table 11: Différents taux de reconnaissance pour chacun des 10 labels sélectionnés pour le modèle 4 couches CNN et une matrice 128x64 (dans le cas 2 poolings, 3 poolings et le cas 4 poolings)

4.4. Fixer le nombre de labels à identifier à 35 :

Dans cette partie, on va essayer de distinguer 35 mots. D'abord, on a essayé d'entraîner le modèle de 4 CNN avec 2 poolings (matrice d'entrée 64×128).

La précision totale est 86,2%.

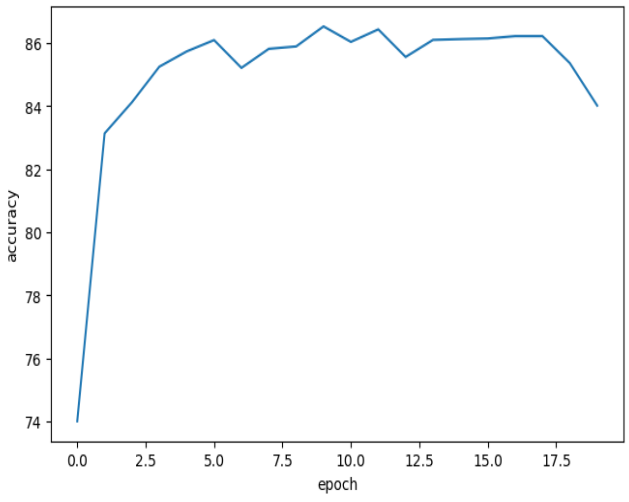

Figure 21: Evolution de la précision du modèle 4 couches du CNN et 2 poolings pour une matrice de taille 128 x 64.

Précision	4 CNN 2 pooling
70% - 74%	″three″ ″learn″ ″tree″ ″forward″ ″go″
75% - 79%	″bird″ ″off″ ″visual″ ″cat″ ″follow″ ″right″
80% - 84%	″four″ ″dog″ ″no″ ″left″ ″bed″ ″nine″ ″up″
85% - 89%	″eight″ ″seven″ ″on″ ″two″ ″stop″ ″one″ ″house″ ″happy″ ″wow″ ″five″ ″marvin″ ″backward″
90% - 94%	″zero″ ″yes″ ″sheila″ ″down″ ″six″

Table 12: Différents taux de reconnaissance pour chacun des 35 labels pour le modèle 4 couches CNN, 2 poolings et une matrice 128x64

Pour 35 sorties, le modèle devra être plus complexe. Alors deux idées se présentent pour le faire :

1) Ajouter une autre couche CNN
2) Ajouter plus de filtres dans chaque CNN

Finalement nous avons réalisé un filtre de 5 couches de CNN avec 3 pooling. Maintenant, la première couche de CNN contient 12 noyaux de 3×3, la deuxième couche de CNN contient 30 noyaux de 3×3, la troisième couche de CNN contient 60 noyaux de 3×3, la quatrième couche de CNN contient 80 noyaux de 3×3, la cinquième couche de CNN contient 120 noyaux de 3×3.

La précision totale est de 86,1 %

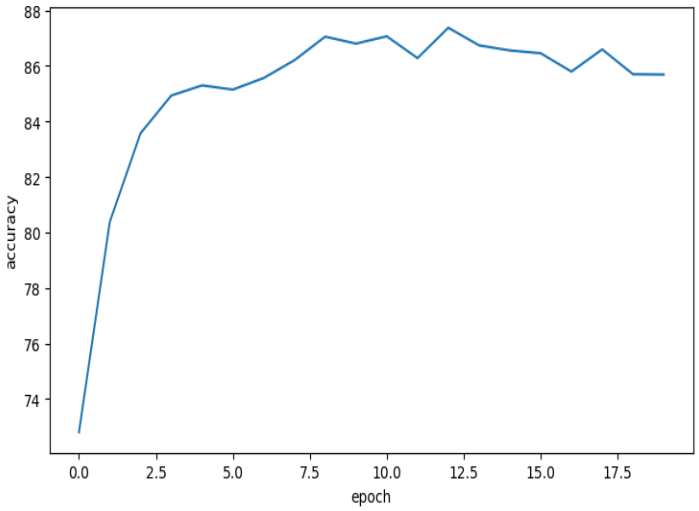

Figure 22: Evolution de la précision du modèle 5 couches du CNN et 3 poolings pour une matrice de taille 128 x 64

Précision	5 CNN 3 pooling
65% - 69%	˝learn˝ ˝two˝
70% - 74%	˝follow˝ ˝tree˝
75% - 79%	˝forward˝ ˝no˝
80% - 84%	˝marvin˝ ˝visual˝ ˝bed˝ ˝dog˝ ˝off˝ ˝go˝
85% - 89%	˝nine˝ ˝sheila˝ ˝down˝ ˝eight˝ ˝left˝ ˝on˝ ˝house˝ ˝one˝ ˝three˝ ˝bird˝ ˝cat˝ ˝four˝
90% - 94%	˝right˝ ˝wow˝ ˝five˝ ˝happy˝ ˝seven˝ ˝backward˝ ˝six˝ ˝stop˝ ˝up˝ ˝zero˝ ˝yes˝

Table 13: Différents taux de reconnaissance pour chacun des 35 labels pour le modèle 5 couches CNN avec 3 poolings et une matrice 128x64

Même si 5 couches de CNN avec 3 poolings a la même précision par rapport à 4 couches de CNN avec 2 pooling, mais pour 5 couches de CNN, il y a que 6 mots dont la précision est inférieure à 80% et il y a 11 mots dont la précision est supérieure à 90%. Et pour 4 couches de CNN, il y a 11 mots dont la précision est inférieure à 80% et il y a que 5 mots dont la précision est supérieure à 90%. Donc 5 couches de CNN avec 3 poolings est une meilleure structure par rapport à 4 couches de CNN avec 2 poolings.

Vu les contraintes de temps d'apprentissage pour 35 labels, nous n'avons pas pu améliorer davantage le modèle proposé.

5. Conclusion :

Ce projet nous a permis de mettre en œuvre nos compétences en PyTorch pour le deep learning. Nous avons réussi à concevoir un modèle qui peut reconnaître les 35 mots clés. Finalement, la précision sur 10 labels a atteint 94%, et la précision sur 35 labels 86.1%.

La plus grande partie du projet a été développée pour 10 labels. Au début du projet, nous avons construit un réseau de neurones qui a 2 couches CNN et 3 couches entièrement connectées. L'entrée de ce modèle sont les spectrogrammes des enregistrements fournis avec une taille de matrice de 128*32. Nous avons réalisé une série de tests sur ce réseau afin de voir l'influence des différents paramètres sur la performance de notre système pour pouvoir les fixer aux valeurs optimales par la suite.

Notre architecture principale repose sur un réseau CNN parce qu'il est le plus efficace pour le traitement des matrices (qui sont des images dans notre cas). Nous avons essayé d'utiliser les réseaux DNN et CNN avec MFCC. DNN convergeait très lentement. CNN avec MFCC donne des résultats assez satisfaisants, mais nous avons opté pour améliorer les résultats à partir des spectrogrammes.

Nous avons ajusté le nombre de couches de CNN et le nombre de couches de pooling. Pour trouver en fin une structure plus performante : 4 couches de CNN avec 2 couches de pooling qui permet d'atteindre une précision de 91.6%.

Après, nous avons augmenté la taille de matrice d'entrée à 64*128 (qui était à l'origine 32*128). Dans ce cas-là, la meilleure structure est 4 couches de CNN avec 3 couches de poolings, la précision est 94.3%

De même pour 35 mots, nous avons trouvé la structure suivante : 5 couches de CNN avec 3 couches de poolings et qui permet d'atteindre la précision est 86.1%.

Nous pensons possible d'améliorer le modèle proposé pour reconnaitre 35 labels en augmentant la complexité du modèle, mais les contraintes du temps d'apprentissage sont toujours une barrière.