Rapport de TIPE Sup
Reconnaissance vocale lors d'appel d'urgence grâce à un réseau de neurones

Tran-Thuong Tien-Thinh, MPSIA, 2020-2021

Résumé

D'après les chiffres du ministère de la Santé, il y a eu plus de 31 millions d'appels d'urgence en 2019. Ces appels sont répartis sur 103 centres de plus en plus sollicités. Alors que les recommandations fixent un taux de 90% de réponses en moins de 60 secondes, seul 69% des appels étaient décrochés dans la minute.

Afin de répondre à ce problème, nous nous proposons d'étudier une intelligence artificielle capable de faire de la reconnaissance vocale, pour alléger le travail des opérateurs d'appel d'urgence.

Problématique retenue

INFORMATIQUE (Informatique Pratique)

Il s'agit de concevoir un réseau de neurones capable de reconnaitre la voix pour retranscrire ce qui est prononcé et de classer le niveau d'urgence d'un appel.

Objectif TIPE du candidat

1. Faire un réseau de neurones qui converge grâce à l'algorithme du gradient
2. Améliorer la vitesse d'entrainement grâce à des optimizers basés sur la descente de gradient
3. Essayer ce réseau sur la base de données du MNIST pour reconnaitre des chiffres
4. Utilisation du transfert d'apprentissage pour reconnaître le nom de la personne qui a été prononcé dans un audio

1 Un réseau de neuronne simple imitant l'opérateur XOR

La difficulté de l'opérateur XOR est qu'il n'effectue pas une classification linéaire entre les deux entrées :

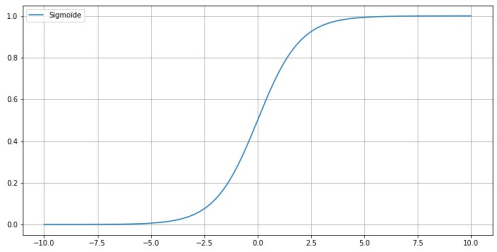
Tableau de l'opérateur XOR		
Entrée 1	Entrée 2	Sortie
0	0	0
1	0	1
0	1	1
1	1	0

1.1 Réseaux de neurones

Il est possible de démontrer que la taille minimum du réseau de neuronne est 2 entrées, 2 neuronnes cachées et 1 neuronne en sortie.

Pour améliorer la vitesse d'apprentissage il est possible d'utiliser des fonctions d'activation permettant de normaliser les sorties de chaque neuronne entre $[0, 1]$, comme la fonction *sigmoïde*.

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{1}$$



Courbe de la fonction d'activation sigmoïde

1.2 Apprentissage

Pour l'apprentissage des poids sur chaque neurone, il calculer l'erreur, et utiliser l'algorithme de rétropropagation. Un des moyen pour calculer la différence à appliquer au poids par rapport à l'erreur $\frac{d_{erreur}}{d_w}$ est la descente de gradient.

$$\frac{d_{erreur}}{d_{poids}} = \frac{d_{erreur}}{d_{\sigma(sortie)}} \times \frac{d_{\sigma(sortie)}}{d_{sortie}} \times \frac{d_{sortie1}}{d_{poids}} \tag{2}$$

$$\frac{d_{erreur}}{d_{poids}} = 2(\sigma(sortie) - sortie_{cible}) \times \sigma'(sortie) \times entree \tag{3}$$

1.3 Resultats

Le réseau de neurones indique bien les résultats associés au XOR :
Les entrées [0, 0] donnent en sortie 0.03356776609399469
Les entrées [1, 0] donnent en sortie 0.9295281916216991
Les entrées [0, 1] donnent en sortie 0.9295281849281336
Les entrées [1, 1] donnent en sortie 0.09395448088880454

Le temps d'apprentrissage semble cependant avoir été très long, l'apprentissage a pris près de 10 000 époques d'apprentissage avant de converger vers un résultat d'erreur inférieur à 10%.

2 Optimizers

Une des principale façon d'accélérer la vitesse d'apprentissage des neuronnes est l'utilisation des optimizers améliorant la descente de gradient simple.

2.1 Regroupement des données par paquet (batch)

Pour améliorer la vitesse d'apprentissage il est intéressant de grouper les données d'apprentissage par paquet (batch) et de mettre à jour le poid des neurones seulement après avoir évaluer la sortie pour chaque paquet de données. On réduit ainsi d'une part le nombre de mise à jour des poids, et on évite également l'influence de données abherrantes.

Afin de choisir rapidement par combien nous regroupons les données il est fortement conseiller de procéder par dichotomie en fonction du nombre de données que l'on possède avec des paquets de $\{64; 128; 256; 512\}$ choisi à chaque fois aléatoirement.

2.2 Moment

Le moment permet d'accentuer la modification des poids lorsque plusieurs données d'affilées génèrent les mêmes modification de poids. Cela permet de de faire varier le taux d'apprentissage (learning rate) en fonction des modifications passées.

Au lieu de modifier directement le poids des neurones avec la formule :

$$poids \leftarrow poids - taux_{apprentissage} \times d_{poids} \tag{4}$$

On prend en compte les poids passées avec la formule avec $\gamma = 0.5$ le taux de prise en compte du moment précédent :

$$d_{moment} \leftarrow \gamma * d_{moment} + taux_{apprentissage} * d_{poids} \tag{5}$$

$$poids \leftarrow poids - d_{moment} \tag{6}$$

Le moment permet également de sortir des problèmes possédant des solutions locales non optimales.

3 MNIST reconnaissance de chiffre écrit à la main

En utilisant tout ce que j'ai fait ci-dessus, j'ai créer un réseau de neuronne capable de reconnaitre des chiffres écrit à la main. Comme c'est un problème de classification avec 10 classes différentes $[|0; 9|]$, il est plus intéressant d'utiliser une fonction d'activation *softmax* plutôt que sigmoïde :

$$\sigma(X)_i = \frac{e^{X_i}}{\sum_{j=0}^{9} e^{X_j}} \quad \text{avec} \quad \text{X la matrice colonne des 10 sorties} \tag{7}$$

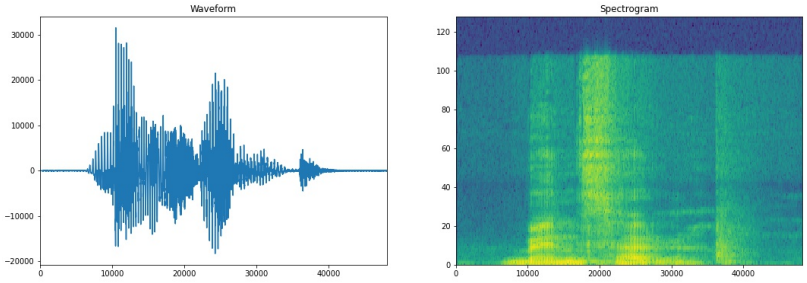
3.1 Résultat

Les résultats sont intérressant, plus de 95% de bonne réponses sur les données de test, mais il a fallu l'entrainer pendant près de 500 époques (50 minutes environ) d'apprentissage.

J'ai donc appliqué l'optimizer avec le moment décrit plus tôt sur les 50 premières époques et on remarque bien un apprentissage plus rapide lors des 10 premières époques.

4 Classification d'audio

Mon projet final est de classifier des séquende d'audio. J'ai donc essayé de faire un réseau de neurones capable de reconnaître le nom de mes professeurs {"Bensal", "Châteaux", "Gayout", "Le Grandic", "Mistler", "Schuschu"}. Pour cela j'ai décomposé l'audio avec la transformée de Fourier pour avoir un spectrogramme de celui-ci.



Un audio "Bensal" et sa décomposition *Short-time Fourier transform*

4.1 Transfert d'apprentissage

Cette section utilise la librairie Tensorflow pour la création et l'entrainement des réseaux de neurones.

J'ai entrainé un model sur une base de données de Google avec les mots {"down", "go", "left", "no", "right", "stop", "up", "yes"} afin de lui permettre de reconnaître des sons caractéristiques avec le plus données possible. Le model était ensuite entrainé, j'ai extrait de ce model les premières couches de neurones et j'ai fixé leurs poids. J'ai ensuite rajouté des neurones permettant à partir des sons caractéristiques reconnus de reconnaitre les noms qui sont prononcés.

Model: "Simple_audio"

Layer (type)	Output Shape	Param #
resizing (Resizing)	(None, 32, 32, 1)	0
normalization (Normalization	(None, 32, 32, 1)	3
conv2d (Conv2D)	(None, 30, 30, 32)	320
conv2d_1 (Conv2D)	(None, 28, 28, 64)	18496
max_pooling2d (MaxPooling2D)	(None, 14, 14, 64)	0
dropout (Dropout)	(None, 14, 14, 64)	0
flatten (Flatten)	(None, 12544)	0
dense (Dense)	(None, 128)	1605760
dropout_1 (Dropout)	(None, 128)	0
dense_1 (Dense)	(None, 8)	1032

Model: "Mon_Model"

Layer (type)	Output Shape	Param #
input_1 (InputLayer)	[(None, 124, 129, 1)]	0
resizing (Resizing)	(None, 32, 32, 1)	0
normalization (Normalization	(None, 32, 32, 1)	3
conv2d (Conv2D)	(None, 30, 30, 32)	320
conv2d_1 (Conv2D)	(None, 28, 28, 64)	18496
max_pooling2d (MaxPooling2D)	(None, 14, 14, 64)	0
dropout (Dropout)	(None, 14, 14, 64)	0
flatten (Flatten)	(None, 12544)	0
dense (Dense)	(None, 128)	1605760
dropout_1 (Dropout)	(None, 128)	0
dense_2 (Dense)	(None, 1024)	132096
dense_3 (Dense)	(None, 6)	6150

Total params: 1,762,825
Trainable params: 138,246
Non-trainable params: 1,624,579

Transfert des couches de *Simple audio* vers *Mon model*

4.2 Augmentation de données

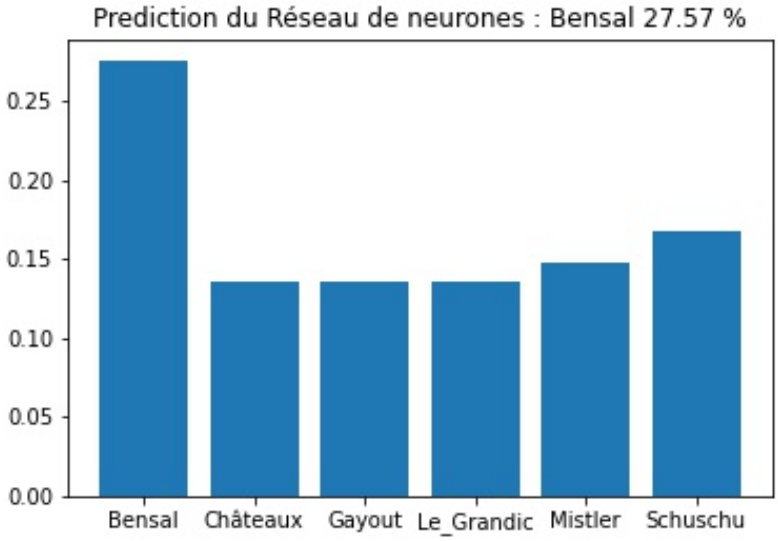
Je me suis enregistré Agathe et moi en train de dire le nom de nos 6 professeurs 8 fois chacun. Etant donné qu'il fallait beaucoup plus de données, j'ai fait de l'augmentation de données (petites modifications des données permettant d'avoir plus de données différentes) :

1. Ajout de bruit $\times 4$
2. Modulation de la voix (grave, aigu) $\times 4$
3. Modulation de la vitesse du son $\times 4$
4. Découpage du son $\times 4$

Ce qui me permet de me retrouver avec $4^4 = 256$ fois plus de données différentes. Ainsi je me retrouve avec $256 \times 8 \times 2 = 4096$ audios.

4.3 Résultat

Les résultat sont concluant, il y a plus de 80% de bonnes réponses lorsque c'est Agathe ou moi qui utilisons testons le réseau de neurones. Ce taux de exactitude est réduit en fonction de la voix et l'intonation de la personne qui teste le model.



Prediction pour un audio où "Bensal" a été prononcé

Annexes

Tous mes codes : github.com/tttienthinh/4Tipe