

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 plus de 31 millions d'appels d'urgence par an. 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 reconnaître la voix, retranscrire ce qui est dit 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'entraînement grâce à des optimizers basés sur la descente de gradient
3. Essayer ce réseau sur la base de données du MNIST pour reconnaître des chiffres
4. Utiliser le transfert d'apprentissage pour reconnaître le nom de la personne qui a été prononcé dans un audio

1 Un réseau de neurones imitant l'opérateur XOR

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

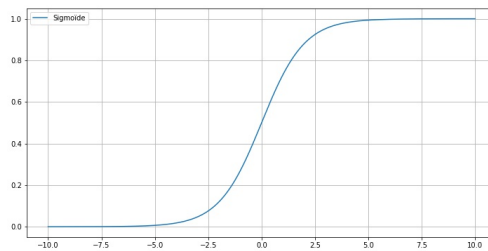
Tableau de l'opérateur XOR		
Entrée 1	Entrée 2	Sortie
0	0	0
1	0	1
0	1	1
1	1	0

1.1 Réseaux de neurones

Il est possible de démontrer que la taille minimum du réseau de neurone est 2 entrées, 2 neurones cachés et 1 neurone en sortie.

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

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



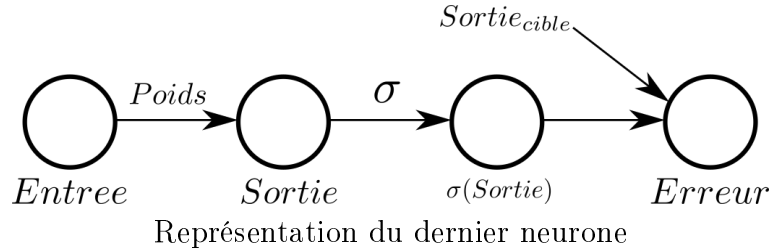
Courbe de la fonction d'activation sigmoïde

1.2 Apprentissage

L'apprentissage des poids sur chaque neurone, se fait en calculant l'erreur, et en utilisant la rétropropagation. On peut calculer la différence à appliquer au poids par rapport à l'erreur $\frac{d_{\text{erreur}}}{d_w}$ grâce à la descente de gradient.

$$\frac{d_{\text{erreur}}}{d_{\text{poids}}} = \frac{d_{\text{erreur}}}{d_{\sigma(\text{sortie})}} \times \frac{d_{\sigma(\text{sortie})}}{d_{\text{sortie}}} \times \frac{d_{\text{sortie}}}{d_{\text{poids}}} \quad (2)$$

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



1.3 Resultats

Le réseau de neurones indique bien les résultats associés au XOR :

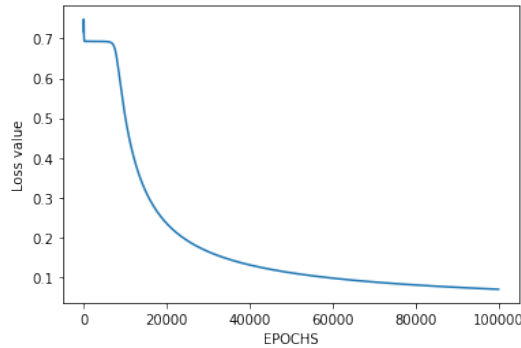
Les entrées [0, 0] donnent en sortie 0.03356776609399469

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'apprentissage est cependant très long, il a pris près de 100 000 époques avant de converger vers un résultat d'erreur inférieur à 10%.



Courbe d'erreur d'apprentissage du XOR

2 Optimizers

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

2.1 Regroupement des données par paquet (batch)

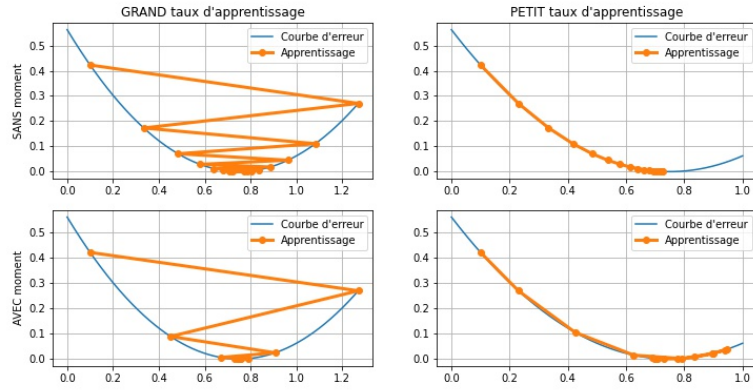
Pour améliorer la vitesse d'apprentissage il est intéressant de grouper les données par paquet (batch) et de mettre à jour le poids des neurones

après avoir évalué la sortie pour toutes les données d'un paquet. On réduit le nombre de mises à jour des poids, et on évite l'impact des données aberrantes.

Afin de choisir rapidement la taille des paquets il est conseillé de procéder par dichotomie avec des paquets aléatoires de $\{64; 128; 256; 512\}$ données.

2.2 Moment

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



Courbe d'erreur d'apprentissage d'un neurone sans puis avec moment

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

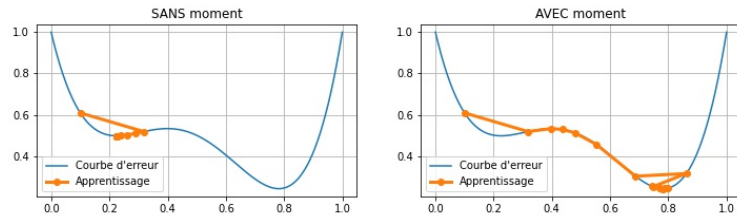
$$poids \leftarrow poids - taux_{apprentissage} \times d_{poids} \quad (4)$$

On prend en compte les poids passés avec $\gamma = 0.5$ le taux de considération du moment précédent :

$$d_{moment} \leftarrow \gamma * d_{moment} + taux_{apprentissage} * d_{poids} \quad (5)$$

$$poids \leftarrow poids - d_{moment} \quad (6)$$

Le moment permet également de résoudre des problèmes possédants des solutions locales non optimales au lieu d'y rester bloquer.



Courbe d'erreur comportant un minimum local

3 MNIST reconnaissance de chiffres écrits à la main

Comme c'est un problème de classification avec 10 classes différentes $[[0; 9]]$, il est plus intéressant d'utiliser la fonction d'activation *softmax* :

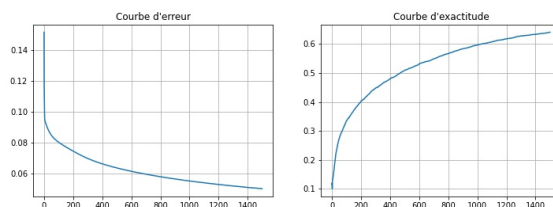
$$\sigma(X)_i = \frac{e^{X_i}}{\sum_{j=0}^9 e^{X_j}} \quad \text{avec } X \text{ la matrice colonne des 10 sorties} \quad (7)$$

3.1 Résultat



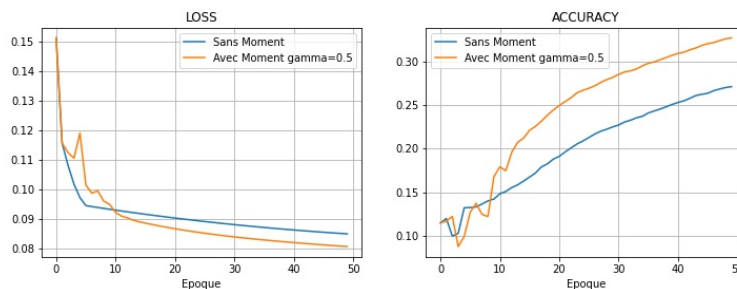
Prediction : 7 bonnes réponses sur 10

On obtient plus de 63% de bonne réponses sur les données de test, mais il a fallu l'entraîner 1 500 époques (50 minutes environ) d'apprentissage.



Courbes d'apprentissage

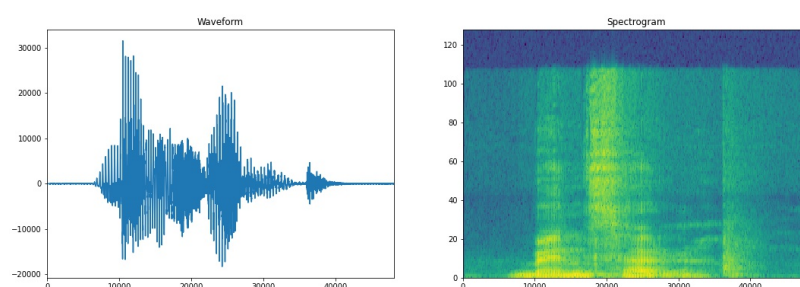
J'ai donc appliqué l'optimizer avec le moment sur les 50 premières époques. On remarque un apprentissage plus rapide au début de l'apprentissage.



Amélioration de l'apprentissage du réseau de neurones grâce au moment

4 Classification d'audio

Mon but est de classifier des séquences audio. J'ai donc essayé de faire un réseau de neurones capable de reconnaître les noms de mes professeurs {"Bensal", "Châteaux", "Gayout", "Le Grandic", "Mistler", "Schuschu"}. Pour cela j'ai décomposé l'audio avec la transformée de Fourier .



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'entraînement des réseaux de neurones.

J'ai entraîné un modèle sur une base de données de Google avec les mots {"down", "go", "left", "no", "right", "stop", "up", "yes"} afin de l'entraîner à reconnaître des sons caractéristiques avec un grand nombre de données. Le modèle entraîné, j'en ai extrait les premières couches de neurones et dont j'ai fixé les poids. J'ai ensuite rajouté des neurones permettant de reconnaître les noms qui sont prononcés à partir des sons caractéristiques.

Model: "Simple_audio"			Model: "Mon_Model"		
Layer (type)	Output Shape	Param #	Layer (type)	Output Shape	Param #
resizing (Resizing)	(None, 32, 32, 1)	0	input_1 (InputLayer)	[(None, 124, 129, 1)]	0
normalization (Normalization)	(None, 32, 32, 1)	3	resizing (Resizing)	(None, 32, 32, 1)	0
conv2d (Conv2D)	(None, 30, 30, 32)	320	normalization (Normalization)	(None, 32, 32, 1)	3
conv2d_1 (Conv2D)	(None, 28, 28, 64)	18496	conv2d (Conv2D)	(None, 30, 30, 32)	320
max_pooling2d (MaxPooling2D)	(None, 14, 14, 64)	0	conv2d_1 (Conv2D)	(None, 28, 28, 64)	18496
dropout (Dropout)	(None, 14, 14, 64)	0	max_pooling2d (MaxPooling2D)	(None, 14, 14, 64)	0
flatten (Flatten)	(None, 12544)	0	dropout (Dropout)	(None, 14, 14, 64)	0
dense (Dense)	(None, 128)	1605760	flatten (Flatten)	(None, 12544)	0
dropout_1 (Dropout)	(None, 128)	0	dense (Dense)	(None, 128)	1605760
dense_1 (Dense)	(None, 8)	1032	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: 130,246		
			Non-trainable params: 1,624,579		

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

4.2 Augmentation de données

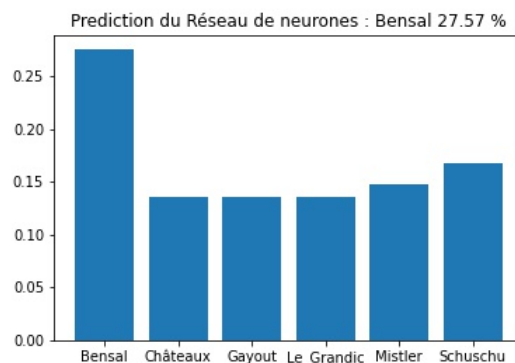
J'ai enregistré Agathe et moi en train de dire les noms 8 fois. Mais il fallait beaucoup plus de données, j'ai donc fait de l'augmentation de données (petites modifications des données permettant d'avoir des données différentes) :

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 résultats sont concluants, il y a plus de 80% de bonnes réponses lorsque c'est Agathe ou moi qui testons. Ce taux d'exactitude est réduit en fonction de la voix et l'intonation de la personne qui teste le modèle.



Prediction pour un audio où "Bensal" a été prononcé

Annexes

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