30903 TRAN THUONG Tien, Thinh

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

Nous nous proposons d'étudier un réseau de neurones réalisant de la reconnaissance vocale, pour alléger le travail des opérateurs d'appel. Ce réseau de neurones devra être capable de classifier des fichiers audios selon des mots-clés relatifs aux appels d'urgence.
En 2018, le ministère de la Santé comptabilisa plus de 31 millions d'appels d'urgence répartis vers des centres toujours plus sollicités. Alors que les recommandations fixent un taux de 90% de réponses en moins de 60 secondes, seuls 69% des appels étaient décrochés dans la minute.

Positionnement thématique (ETAPE 1)

INFORMATIQUE (Informatique pratique).

Mots-clés (ETAPE 1)

Mots-Clés (en français)	**Mots-Clés** (en anglais)
Réseau de neurones	*Neural network*
Apprentissage profond	*Deep learning*
Algorithme du gradient	*Gradient descent*
Transformée de Fourier	*Fourier transform*
Reconnaissance vocale	*Voice recognition*

Bibliographie commentée

La première modélisation informatique du neurone, appelée perceptron, proposée par *McCulloch et Pitts*, date de 1943 [1]. Cette modélisation possède des poids d'apprentissage et une fonction d'activation. Lorsque les données sont entrées, elles sont pondérées par les poids du perceptron puis elles sont sommées et enfin transmises à la fonction d'activation. Le résultat obtenu devient alors la sortie renvoyée par le perceptron. Cela modélise bien le comportement d'un neurone qui récupère les différentes entrées par ses dendrites, les traite et transmet le résultat par son axone. Pour pouvoir traiter des données complexes, il est nécessaire de mettre plusieurs neurones
en réseau et d'effectuer un apprentissage profond.

L'apprentissage s'effectue sur des données dont la sortie attendue est connue. L'entraînement consiste alors à corriger les poids afin de faire converger le système vers un résultat "optimal". Cette rectification des poids du perceptron, se fait en calculant l'erreur en sortie, puis en effectuant la rétropropagation. On calcule la différence à appliquer au poids par rapport à l'erreur grâce à l'algorithme de descente de gradient. [2]

Un perceptron seul est capable de faire une séparation linéaire parmi les données d'entrée. Il peut reproduire les opérateurs logiques AND et OR qui sont linéaires, mais l'opérateur XOR nécessite un réseau de perceptrons car il effectue une séparation non linéaire. On associe ainsi les neurones en parallèle pour former une couche de perceptrons, puis on associe ces couches en série pour former

un réseau de perceptrons. [3]

Comme l'apprentissage d'un réseau de neurones se fait en utilisant des données pris en exemple, deux problèmes majeurs peuvent intervenir: un ensemble de données d'apprentissage biaisé, ou un temps d'apprentissage trop long.
Pour le premier, il est difficile d'y remédier. En effet, il est possible de faire une séparation des données pour l'apprentissage et le test, ou encore d'introduire des variables aléatoires comme la couche "Dropout", mais ces deux solutions ne permettent que d'éviter le sur-apprentissage ("overfitting") et en aucun cas de corriger les erreurs de données défaillantes.
Pour résoudre **le problème du temps d'apprentissage**, plusieurs études ont mené à des approches différentes. Il est possible d'adapter l'architecture du réseau de neurones en fonction du projet mis en œuvre [4], on peut également modifier les fonctions d'activation et d'erreur, ou encore faire varier le taux d'apprentissage des poids à chaque rétropropagation [5]. Dans notre étude, nous nous concentrerons principalement sur cette dernière optimisation qui est plus documentée car aussi plus généraliste. [6]

En 1976, face à la croissance en taille des réseaux de neurones entraînant un temps d'apprentissage plus long, le professeur *Stevo Bozinovski* présente l'idée du **transfert d'apprentissage**. [7] Le principe, au lieu de réentraîner tout un réseau de neurones, est d'en réutiliser un déjà fonctionnel pour un problème similaire et de l'adapter. Il faut pour cela initialiser les poids des perceptrons avec ceux de l'ancien réseau de neurones, en ne laissant modifiable par rétropropagation que les perceptrons de la dernière couche, celle de sortie. Le temps d'apprentissage est donc réduit par réduction du nombre de poids à entraîner. C'est ce que nous propose de réaliser la librairie *Tensorflow* avec les différents modèles qu'elle met à disposition. [8]

C'est en 1952 que les travaux de Bell Labs sur le système "Audrey" ouvrent la voie de la reconnaissance vocale, celui-ci permettant de dissocier la prononciation des 10 chiffres. Ainsi, le signal audio était converti en spectre de fréquence notamment grâce à la transformée de Fourier puis analysé par des règles prédéfinies en fonction de la fréquence d'enchaînement des sons, ces algorithmes étaient nommés "Hidden Markov Model" (HMM). [9] De nos jours, les HMM fonctionnent de pair avec une analyse faite par les réseaux de neurones, qui eux permettent d'identifier des sons plus subtils comme le bruit ambiant, le ton ou encore le rythme de la voix.

Problématique retenue

Concevoir un réseau de neurones capable de reconnaître des mots-clés prononcés dans un extrait audio. Les contraintes sont d'utiliser des méthodes rendant l'apprentissage peu gourmand en ressources et surtout exécutable en un temps raisonnable.

Objectifs du TIPE

1. Réaliser 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. Mettre en œuvre ce réseau sur la base de données du MNIST pour reconnaitre des chiffres.

4. Utiliser le **transfert d'apprentissage** pour reconnaître ce qui a été prononcé parmi une liste de mots-clés, dans un enregistrement audio.

Références bibliographiques (ETAPE 1)

[1] Warren McCulloch and Walter Pitts : A logical calculus of the ideas immanent in nervous activity : *volume 5. Bulletin of mathematical biophysics, 1943*

[2] 3Blue1Brown : Gradient descent, how neural networks learn : *youtube.com/watch?v=IHZwWFHWa-w, 2017*

[3] Kirill Goltsman : Introduction to artificial neural networks : *Data Science Foundation, 2017*

[4] Roberto Iriondo Pratik Shukla : Main types of neural networks and their applications : *pub.towardsai.net/main-types-of-neural-networks-and-its-applications-tutorial-734480d7ec8e*

[5] Sebastian Ruder : An overview of gradient descent optimization algorithms : *arxiv.org/abs/1609.04747, 2016*

[6] Christopher M. Bishop : Pattern Recognition and Machine Learning : *Springer, 2006*

[7] Stevo Bozinovski : Reminder of the first paper on transfer learning in neural networks, 1976 : *volume 5. Informatica, 2020*

[8] TensorFlow TutorialsSimple audio recognition: Recognizing keywords : Simple audio recognition: Recognizing keywords : *tensorflow.org/tutorials/audio/simple_ audio*

[9] Biing Hwang Juang and Laurence R Rabiner : Hidden markov models for speech recognition : *Technometrics, 1991*