SCIENCES SORBONNE UNIVERSITÉ

Projet en traitement avancé du son - M2 ISI 2020/2021

Groupe 9, sujet 4.2 : Reconnaissance de la Parole
Kadi Koussaila - Loto Mustpha - Rais Mohamed Ali - Xu Qi

Lien Github pour notre projet

1 Présentation et objectif du projet

Ce projet porte sur la transcription textuelle à partir d'un enregistrement de parole et vise à implémenter un réseau de neurones pour réaliser un système de reconnaissance de la parole à large vocabulaire à partir d'une architecture de réseau de neurones séquence-à-séquence avec une loss CTC (Connexionist Temporal Classification) [1]. D'abord, nous présenterons l'état de l'art de la reconnaissance automatique de la parole(voir la section 2). Puis nous détaillerons la méthode qu'on a utilisée(voir la section 3). Enfin, nous analyserons les résultats dans une dernière partie, avant une courte conclusion.

2 État de l'art

2.1 La reconnaissance de la parole

L'objectif d'un système d'extraction automatique de la parole est d'extraire les mots prononcés à partir du signal acoustique.

Il existe nombreuses applications qui utilisent la technologie de la reconnaissance vocale, comme Google Docs Voice Typing, Speechnotes, Dragon, etc. Ces applications sont très efficaces pour convertir des phrases dictées en texte. La technologie de la reconnaissance vocale est aussi utilisée dans la commande vocale, comme Google Assitant ou Siri d'Apple. La combinaison de la technologie de reconnaissance vocale avec d'autres technologies de traitement du langage naturel, telles que la traduction automatique et la synthèse vocale, permet de créer des applications plus complexes, telles que la traduction vocale, comme Google Traduction.

2.2 Historique

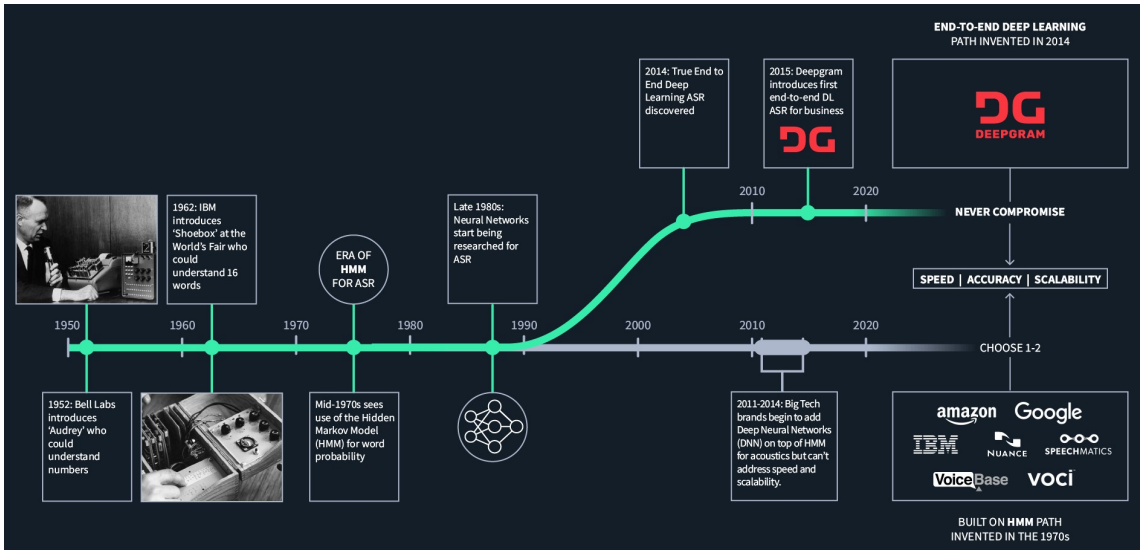


FIGURE 1 – L'historique de la reconnaissance de la parole [2]

Les travaux sur la reconnaissance de la parole datent du début du 20e siècle. Le premier système pouvant être considéré comme faisant de la reconnaissance de la parole date de 1952. Ce système électronique développé par Davis, Biddulph et Balashek aux laboratoires Bell Labs était essentiellement composé de relais et ses performances se limitaient à reconnaître des chiffres isolés.

Dix ans plus tard, le premier assistant vocal a été inventé en 1962. IBM a lancé "Shoebox" qui comprenait et répondait à 16 mots en anglais. Le système était nommé "Shoebox", car il ressemblait à une boîte à chaussures. La recherche s'est ensuite considérablement accrue durant les années 1970 avec les travaux de Jelinek chez IBM (1972-1993) et lorsque les chercheurs ont commencé à utiliser les modèles de Markov cachés (HMM). En fin de 1980, les recherches sur la reconnaissance de la parole commencent à baser sur les réseaux de neurones. [3]

Actuellement, la reconnaissance de la parole utilise deux grandes techniques : HMM et/ou les réseaux de neurones.

2.3 Les méthodes de la reconnaissance de la parole

2.3.1 Modèle de Markov caché

Avant le début du Deep Learning, les modèles de Markov caché (HMM) étaient le modèle plus utilisé pour la reconnaissance vocale. Par exemple,"*Maximum Mutual Information Estimation of Hidden Markov Model Parameters for Speech Recognition*" [4], "*Open vocabulary speech recognition with flat hybrid models*" [5], SGMM-HMM (Gaussian mixture models pour SGMM) [6], etc.

Un HMM est un modèle statistique à un processus stochastique qui peut être utilisé pour décrire l'évolution d'événements observables qui dépendent de facteurs internes et qui ne sont pas directement perceptibles, sachant que l'événement futur dépend de l'événement passé. [2]

Les HMM utilisent des fonctions de probabilité pour déterminer les mots corrects à transcrire. Ces modèles vocaux de la reconnaissance de la parole prennent des extraits d'audio pour déterminer la plus petite unité sonore d'un mot ou ce qu'on appelle un phonème.

Le phonème est inséré par la suite dans un autre programme qui utilise le HMM pour deviner le bon mot à l'aide d'une fonction de probabilité des mots les plus courants. Ces modèles de traitement en série sont affinés en ajoutant une réduction du bruit en amont et des modèles de langage pour créer des textes et des phrases compréhensibles.

Nous pouvons utiliser un spectrogramme pour représenter le signal audio. Étant donné les caractéristiques audio $X = (x_1, x_2, \ldots, x_T)$, déduire la séquence de texte $Y^* = (y_1, y_2, \ldots, y_L)$ la plus probable qui a causé par les caractéristiques audio.

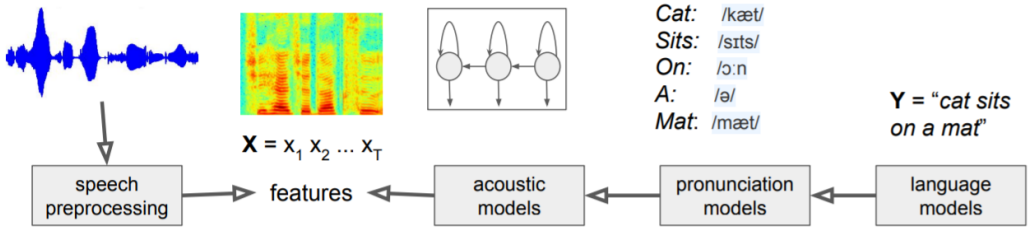


FIGURE 2 – HMM [7]

$$Y^* = \arg\max_{Y}(p(X|Y)p(Y)) \tag{1}$$

La recherche de faisceau est une fonction de probabilité dépendant du temps et examine les mots transcrits avant et après le mot cible pour trouver la meilleure correspondance pour le mot cible. L'ensemble de ce processus en série est appelé le modèle "trigramme", et 80% de la technologie de la reconnaissance vocale actuellement utilisée est une version raffinée de ce modèle des années 1970.

2.3.2 Les méthodes de réseaux de neurones

Il existe de nombreuses variantes de l'architecture d'apprentissage en profondeur pour la reconnaissance de la parole. L'approche le plus couramment utilisée est un réseau de séquence-à-séquence basé sur un RNN qui traite chaque trame du spectrogramme comme un élément d'une séquence. [8] [9] L'architecture que nous utilisons dans ce projet est aussi un RNN. [1]

Un réseau récurrent composé de quelques couches LSTM bidirectionnelles qui traitent les cartes de caractéristiques sous la forme d'une série de pas de temps distincts ou de trames qui correspondent à la

séquence de caractères de sortie souhaitée. Un LSTM est un type de couche récurrente très couramment utilisé, dont la forme complète est Mémoire à Long Court Terme(Long Short-Term Memory).

Il est aussi très courant d'ajouter un réseau convolutionnel avant les couches LSTM. [10] Un réseau convolutionnel est composé des couches CNN résiduelles qui traitent les images de spectrogrammes en entrée et produisent des cartes de caractéristiques de ces images.

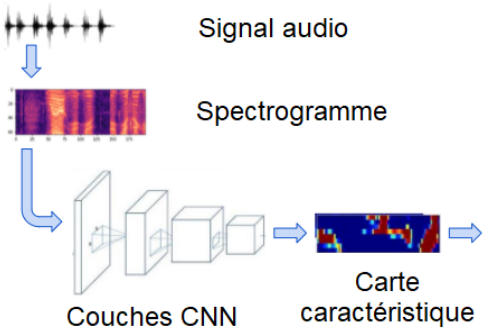


FIGURE 3 – Les spectrogrammes sont traités par un réseau convolutionnel pour produire des cartes de caractéristiques. [11]

2.3.3 HMM vs RNN (BLSTM)

Les deux modèles les plus utilisés pour la reconnaissance de la parole, sont les HMMs et RNNs qui ont des différences fondamentales. Le modèle de Markov caché s'appuie sur les statistiques et les distributions, et donc sur la maximisation des probabilités, alors qu'un LSTM qui est s'appuie sur les réseaux de neurones et donc la recherche des relations dans l'ensemble des données. [12]

Les modèles de Markov cachés (HMM) sont beaucoup plus simples que les réseaux de neurones récurrents (RNN) et reposent sur des hypothèses fortes qui ne sont pas toujours vraies. Si ces hypothèses sont vraies, un HMM peut offrir de meilleures performances, car il est moins difficile de le faire fonctionner.

Un RNN peut être plus performant si nous disposons d'un très grand ensemble de données, car la complexité supplémentaire permet de mieux exploiter les informations contenues dans vos données. Cela peut être vrai même si les hypothèses des HMM sont vraies.

Il existe aussi des méthodes hybrides en combinant HMM avec le réseau de neurones. Par exemple, "*Connectionist speech recognition : a hybrid approach*" [13], "*Application of pretrained deep neural networks to large vocabulary speech recognition*" [14],"*Hybrid automatic speech recognition model for speech-to-text application in smartphone*" [15], et etc.

3 Architecture proposée

L'architecture implémentée est proposée par l'article "*Towards End-to-End Speech Recognition with Recurrent Neural Networks*" [1], où le réseau est composé de 5 couches de Bidirectional Long Short-Term Memory **(BLSTM)** et une loss : Connectionist temporal classification **(CTC)**. Un exemple d'architecture proposée est ci-dessous :

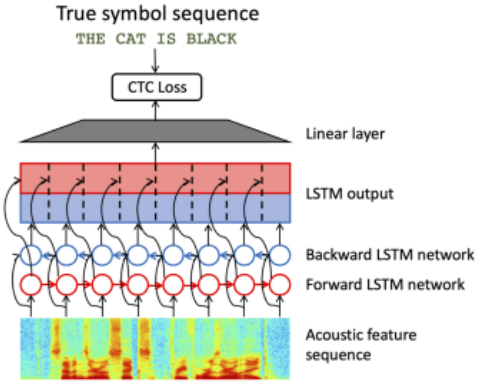


FIGURE 4 – Exemple d'architecture proposée [1]

3.1 Bidirectional Long Short-Term Memory

Le réseau **LSTM** est une architecture de réseau neuronal récurrent (RNN). Sa structure est similaire à un RNN avec un comportement interne différent où il se compose de plusieurs cellules telles que (Forget gate, output gate, input gate...etc). L'avantage de ce type d'architecture est la mémorisation des évènements du passé.

On utilise un type d'architecture du LSTM appelé **Bidirectional LSTM**, cette architecture permet d'exécuter les entrées de deux manières, l'une du passé vers le futur et l'autre du futur vers le passé. Ce qui différencie cette approche de l'approche unidirectionnelle, en utilisant les deux états cachés combinés, on est capable à tout moment de préserver les informations du passé et du futur [16].

Dans cette application on se base sur le modèle sequence-to-sequence **(S2S)** qui fait la correspondance entre deux séquences. Dans notre cas entre l'audio et sa transcription. Un exemple du modèle du LSTM bidirectionnelle est ci-dessous :

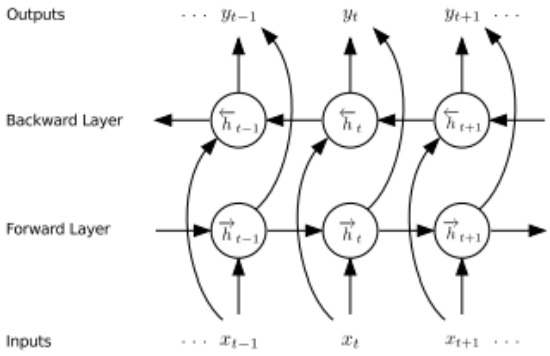


FIGURE 5 – Modèle du LSTM bidirectionnelle [1]

3.2 Connectionist Temporal Classification

La classification temporelle connexionniste **C**TC est une solution remarquable pour transformer les sorties d'un réseau neuronal récurrent en séquences de labels. La dernière couche du réseau contient une unité par étiquette qui produit la probabilité d'avoir l'étiquette correspondante à un pas de temps particulier. Lorsque l'on utilise l'approche CTC, une unité supplémentaire est définie pour modéliser la probabilité d'avoir une étiquette vierge.

Dans le processus de transcription de la parole en texte, on ne peut pas savoir comment les caractères de la transcription s'alignent sur l'audio. L'algorithme CTC permet de contourner ce problème puisqu'il ne nécessite pas d'alignement entre l'entrée et la sortie. Cependant, pour obtenir la probabilité d'une sortie en fonction d'une entrée, CTC additionne la probabilité de tous les alignements possibles entre les deux. Les alignements CTC nous donnent un moyen naturel de passer des probabilités à chaque étape temporelle à la probabilité d'une séquence de sortie. [17]

La CTC fonctionne selon deux modes :

— CTC loss (pendant l'entraînement) : Il dispose d'une transcription cible de vérité de base et tente d'entraîner le réseau pour maximiser la probabilité de produire cette transcription correcte.

— CTC décodage(pendant la prédiction) : ici, nous n'avons pas de transcription cible à laquelle nous référer, et nous devons prédire la séquence de caractères la plus probable.

3.2.1 CTC décodage

CTC utilise les probabilités de caractères pour choisir le caractère le plus probable pour chaque trame, y compris les blancs. Nous prenons l'exemple "-G-o-ood".

On fusionne tous les caractères qui sont répétés, et qui ne sont pas séparés par un blanc. Par exemple, nous pouvons fusionner le "oo" en un seul "o", mais nous ne pouvons pas fusionner le "o-oo". C'est ainsi que le CTC est capable de distinguer qu'il y a deux "o" distincts et de produire des mots épelés avec des caractères répétés, nous obtenons "-G-o-od".

Enfin, puisque les blancs ont rempli leur fonction, il supprime tous les caractères blancs, et le mot prédit est "Good".

3.2.2 CTC loss

La perte est calculée comme la probabilité que le réseau prédise la séquence correcte. Pour trouver la séquence de mots la plus probable, nous recherchons différentes combinaisons de chemins. Nous devons additionner tous les chemins qui génèrent la même séquence de mots. Par exemple, pour trouver la probabilité du mot "hello", nous additionnons tous les chemins correspondants comme "heellelloo", "hhelleeleo", "eeelleeloo", etc. [7]

Le CTC loss est calculé de manière suivante :

$$L_{ctc} = -log(P(y|x)) \tag{2}$$

où y la transcription cible, x la séquence d'entrée.

Nous devons sommer sur tous les chemins possible pour avoir $P(y|x)$:

$$P(y|x) = \sum_{a \in B^{-1}(y)} P(a|x) \tag{3}$$

La probabilité d'un chemin :

$$P(a|x) = \prod_{t=1}^{T} P(a_t, t|x) \tag{4}$$

Nous avons repris les codes de CTC sur Github pour construire notre architecture.

4 Données utilisées

Plusieurs bases de données sont proposées pour les applications de la transcription de la parole (TIMIT, VCTK ...). Dans ce projet, on a choisi d'utiliser la base de données TIMIT pour l'apprentissage et la validation de notre modèle.

4.1 Base de données TIMIT

Le corpus TIMIT de lecture vocale a été développé pour fournir des données vocales pour les études de recherche acoustiques et phonétiques et pour l'évaluation des systèmes de reconnaissance automatique de la parole.

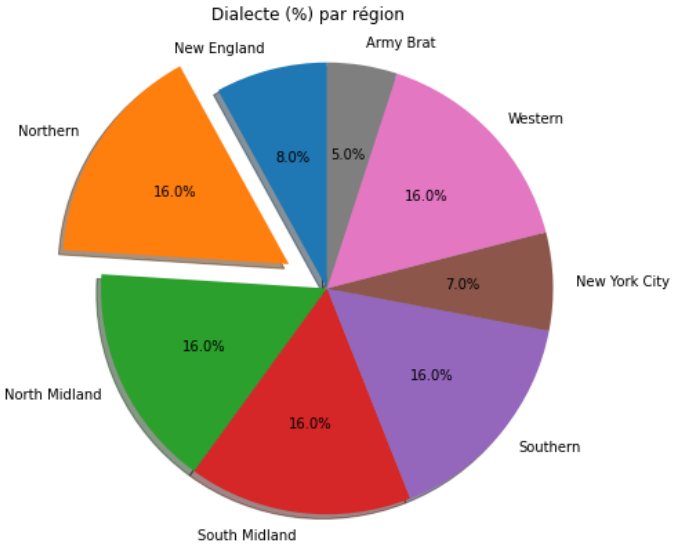


FIGURE 6 – Répartition des données par dialecte

TIMIT contient des enregistrements de haute qualité de 630 individus/parleurs avec 8 dialectes différents de l'anglais américain, chaque individu lisant jusqu'à 10 phrases phonétiquement riches.

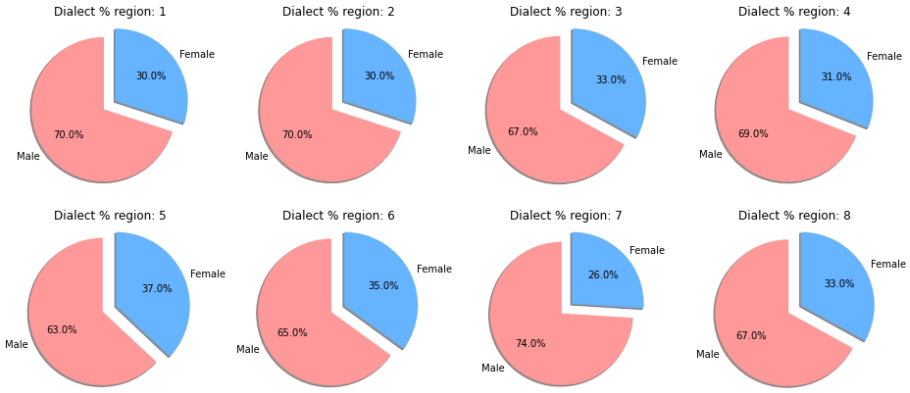


FIGURE 7 – Répartition des données par le gendre

4.2 Générateur des données

Un générateur de données est généralement utilisé dans le cas des grandes bases de données. Dans certains cas, même la configuration la plus moderne n'aura pas assez d'espaces mémoires pour traiter la grande quantité de données.

Pour cela, une solution est proposée par Keras "*D*ata Generator" [18] qui consiste a générer l'ensemble de données sur plusieurs batchs en temps réel et l'alimenter immédiatement au modèle d'apprentissage. Cette solution optimise la quantité de mémoire utilisée pour stocker les données et permet également d'effectuer le pré-traitement nécessaire sur les batchs (voir 4.3.1).

4.3 Pré-traitement des données

4.3.1 Pré-traitement des signaux audio

Afin de préparer les données d'entrée du réseau, on réalise dans un premier temps un pré-traitement des signaux audio en suivant les étapes suivantes :

— **La normalisation des signaux :** Soustraire la moyenne et diviser chaque son par l'écart-type.

— **Feature extraction :** On calcul la Transformée de Fourier à court terme (STFT) à l'aide de la librairie "Librosa" (utilisation STFT pour éviter le problème de filtres vides rencontré lors de l'utilisation du mel spectrogram) pour extraire les caractéristiques qui révèle la distribution de l'énergie de chaque signal (spectrogramme log-fréquence en convertissant l'axe linéaire des fréquences (mesuré en Hertz) en un axe logarithmique (mesuré en hauteurs)).
Les paramètres utilisés dans le calcul de la STFT sont proposés par l'article (largeur de la fenêtre de Fourier **N**fft = 256 , un overlap de 127 fenêtres, soit 128 entrées par fenêtre).

— **Padding :** La dernière étape est l'étape pour avoir des spectogrammes de la même taille (taille de spectrogramme le plus grand).

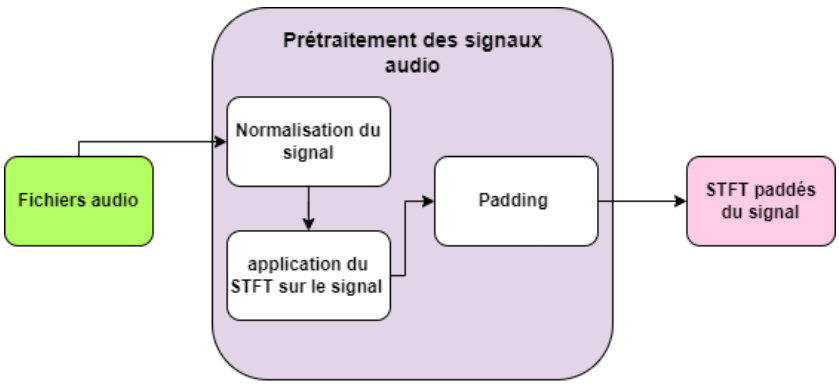


FIGURE 8 – Pré-traitement des signaux audio

4.3.2 Pré-traitement des labels

Les labels de notre base de données représentent les phrases associées aux signaux audio. Les pré-traitements réalisés sur ses données sont :

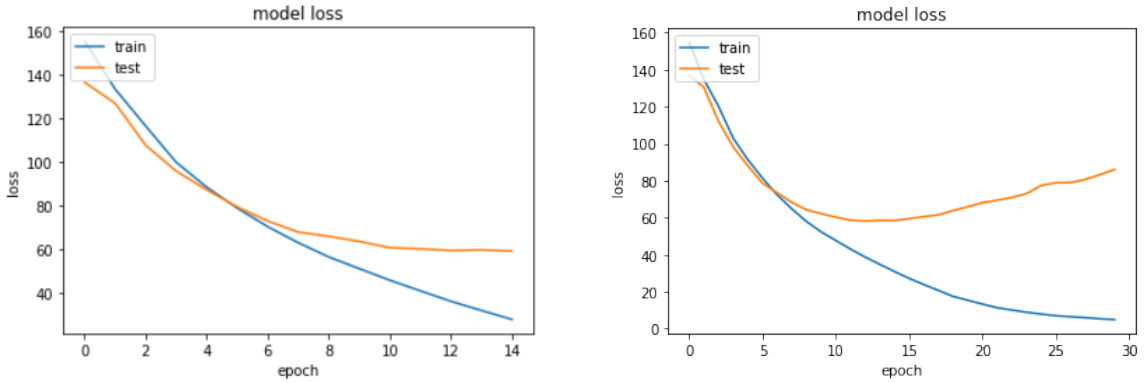
— Normalisation du texte (enlever la ponctuation et convertir toutes les lettres en minuscule).
— Encodage des phrases en séquences d'entiers en utilisant un dictionnaire qui mappe chaque lettre à un chiffre.
— Padding des séquences encodées selon la phrase la plus longue.

5 Apprentissage

Pour la phase d'apprentissage, on a tenté de jouer sur plusieurs paramètres afin de trouver la meilleure architecture qui minimise la loss.

L'apprentissage de nos modèles a été fait sur les serveurs de Google Colab Pro, cela nous a permit d'accélérer le temps d'apprentissage (environ 20 minutes par epoch), la base de données a été chargée sur Google drive puis récupérée par Colab à travers le Data Generator.

Les courbes d'apprentissage du meilleur modèle sont représentées ci-dessous :



(a) evolution du loss (b) Overfitting

FIGURE 9 – Courbes d'apprentissage

On constate à partir de la courbe d'apprentissage sur 30 epochs qu'à partir de 16 à 17 epochs on tombe dans le sur-apprentissage (overfitting), on a tenté de rajouter de dropout sur différentes couches ainsi que rajouter de la régularisation de norme L2 sauf qu'on n'a pas réussi à dépasser 19 epochs sans rentrer dans l'overfitting malgré ces changements.

6 Résultats et discussion

6.1 Métriques

6.1.1 Word Error Rate(WER)

Après avoir entraîné notre réseau, nous devons évaluer ses performances. Une métrique couramment utilisée pour les problèmes de conversion parole-texte est le taux d'erreurs de mots et de caractères : Word Error Rate (WER). Il compare la sortie prédite et la transcription label, mot par mot ou caractère par caractère pour déterminer le nombre de différences entre les deux. Une différence peut être un mot présent dans la transcription, mais absent dans la prédiction, il est considéré comme une suppression ; un mot absent dans la transcription mais ajouté à la prédiction considéré comme une insertion ; ou un mot modifié entre la prédiction et la transcription, c'est-à-dire une substitution.

La formule de la métrique est assez simple. Il s'agit du pourcentage de différences par rapport au nombre total de mots.

$$WER = \frac{\text{insertion} + \text{suppression} + \text{substitution}}{\text{nombre de mots total dans la transcription}} \tag{5}$$

6.1.2 Label Error Rate(LER)

Il existe une autre métrique : Label Error Rate(LER) qui a été décrit dans la section 2 d'un autre article de Graves. [19]

Soit S un ensemble d'exemples d'apprentissage. Étant donné un ensemble de test S'. Chaque exemple de S est constitué d'une paire de séquences $(x, z) \in X \times Z$, où x la séquence d'entrée et z la séquence de label. Ils sont reliés par la fonction $h : X \mapsto Z$. Le taux d'erreur LER de h est la distance d'édition normalisée entre ses classifications et les cibles S' :

$$LER(h, S') = \frac{1}{Z} \sum_{(x,z) \in S'} \frac{ED(h(x), z)}{|z|} \tag{6}$$

où Z est le nombre total de label dans S', et $ED(p, q)$ est la distance d'édition entre les deux séquences p et q, c'est-à-dire le nombre minimal d'insertions, de substitutions et de suppressions nécessaires pour transformer p en q.

6.2 Résultats obtenus

Les meilleurs résultats obtenus sont avec un apprentissage du modèle sur 19 epochs avec dropout de 40% et régularisation de norme L2 (kernel regularizer et recurrent regularizer avec $\lambda = 10^{-6}$) sur la dernière couche.

Les métriques d'évaluations ainsi que les résultats obtenus pour chaque métrique avec le meilleur modèle sont :

— Word Error Rate(WER) :73.3%
— Label Error Rate (LER) :35.4%
— Sequence Error Rate (SER) : 95.2%

Des exemples de résultats de prédictions sont présentés ci-dessous :

— **(1)**
predicted : dontt ask me to carry an oily rag like that
Ground truth : dont ask me to carry an oily rag like that
— **(2)**
predicted : ha twilyd of the tie t de wil hae shibly
Ground truth : at twilight on the twelfth day well have chablis
— **(3)**
predicted : abig go iblye amble throi the far near
Ground truth : a big goat idly ambled through the farmyard
— **(4)**
predicted : seas thanter then aon
Ground truth : she is thinner than i am
— **(5)**
predicted : she had your dark suit in greasy wash water all year
Ground truth : she had your dark suit in greasy wash water all year

6.3 Discussion

D'après les résultats, on remarque que dans certains cas le modèle peut prédire parfaitement les labels des sons, comme il peut rater l'épellation de quelques mots, mais en suivant toujours plus au moins la prononciation du mot, comme *throught* epelé *throi* dans l'exemple 3.

On observe que l'erreur LER est beaucoup plus petite que WER. Parce que LER s'agit d'une mesure plus naturelle dont l'objectif est de minimiser le taux d'erreurs de transcription.

Nous avons aussi tenté d'enregistrer nos voix et de les passer dans le modèle de prédiction, mais le modèle n'a pas réussi à prédire une phrase. En fait, il y a une grande variation de conditions d'enregistrement entre la base que nous avons utilisée pour les entraînements et les enregistrements faits maison.

Le modèle pourrait être amélioré davantage en l'entraînant sur d'autres bases de données plus riches, les contraintes de temps et de matériel nous ont empêchées d'aller plus loin avec ce projet.

7 Conclusion

Au cours de ce projet, nous avons cherché à améliorer la qualité de la reconnaissance. Les résultats obtenus ont montré que le modèle de prédiction arrive déjà à prédire des phrases même si les taux d'erreur sont encore élevés. Si nous avions eu plus de temps, nous aurions pu continuer à entraîner notre modèle avec une autre base de données VCTK pour améliorer la qualité de la prédiction, et à développer un système de correction avec un dictionnaire pour rectifier les mots qui n'ont pas de sens et de trouver le mot plus ressemblé dans le dictionnaire.

Références

[1] Alex Graves and Navdeep Jaitly. Towards end-to-end speech recognition with recurrent neural networks. In *International conference on machine learning*, pages 1764–1772. PMLR, 2014.

[2] The history of automatic speech recognition.

[3] Douglas O'Shaughnessy. Automatic speech recognition : History, methods and challenges. *Pattern Recognition*, 41(10) :2965–2979, 2008.

[4] Lalit Bahl, Peter Brown, Peter De Souza, and Robert Mercer. Maximum mutual information estimation of hidden markov model parameters for speech recognition. In *ICASSP'86. IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 11, pages 49–52. IEEE, 1986.

[5] Maximilian Bisani and Hermann Ney. Open vocabulary speech recognition with flat hybrid models. In *Interspeech*, pages 725–728, 2005.

[6] Daniel Povey, Arnab Ghoshal, Gilles Boulianne, Lukas Burget, Ondrej Glembek, Nagendra Goel, Mirko Hannemann, Petr Motlicek, Yanmin Qian, Petr Schwarz, et al. The kaldi speech recognition toolkit. In *IEEE 2011 workshop on automatic speech recognition and understanding*, number CONF. IEEE Signal Processing Society, 2011.

[7] Jonathan Hui. Speech recognition — deep speech, CTC, listen, attend, and spell.

[8] Alex Graves. Supervised sequence labelling. In *Supervised sequence labelling with recurrent neural networks*, pages 5–13. Springer, 2012.

[9] Florian Eyben, Martin Wöllmer, Björn Schuller, and Alex Graves. From speech to letters-using a novel neural network architecture for grapheme based asr. In *2009 IEEE Workshop on Automatic Speech Recognition & Understanding*, pages 376–380. IEEE, 2009.

[10] Takaaki Hori, Shinji Watanabe, Yu Zhang, and William Chan. Advances in joint ctc-attention based end-to-end speech recognition with a deep cnn encoder and rnn-lm. *arXiv preprint arXiv :1706.02737*, 2017.

[11] Ketan Doshi. Audio deep learning made simple (part 1) : State-of-the-art techniques.

[12] Bram de Wit. *Stock Prediction Using a Hidden Markov Model Versus a Long Short-Term Memory*. PhD thesis, 2019.

[13] Herve A Bourlard and Nelson Morgan. *Connectionist speech recognition : a hybrid approach*, volume 247. Springer Science & Business Media, 2012.

[14] Navdeep Jaitly, Patrick Nguyen, Andrew Senior, and Vincent Vanhoucke. Application of pretrained deep neural networks to large vocabulary speech recognition. 2012.

[15] Riky Kusumah, Rudy Hartanto, and Risanuri Hidayat. Hybrid automatic speech recognition model for speech-to-text application in smartphones. In *2019 International Conference on ICT for Smart Society (ICISS)*, volume 7, pages 1–5. IEEE, 2019.

[16] The-unfolded-architecture-of-bidirectional-lstm-bilstm.

[17] Thierry Paquet Yann Soullard, Cyprien Ruffino. Ctcmodel : a keras model for connectionist temporal classification. 2019.

[18] keras- how to generate data on the fly.

[19] Alex Graves, Santiago Fernández, Faustino Gomez, and Jürgen Schmidhuber. Connectionist temporal classification : labelling unsegmented sequence data with recurrent neural networks. In *Proceedings of the 23rd international conference on Machine learning*, pages 369–376, 2006.