Danxin CUI Xinyi SHEN Jianying LIU

Compte-rendu du projet final

-- Identification des Langues

Introduction

L'initiative «une ceinture et une route» proposée par le gouvernement chinois en 2014 concerne géographiquement plus de 60 pays avec une vocation politico-économique. Dès lors, la coopération et les échanges internationaux transfrontaliers entre ces pays sont de plus en plus fréquents. Les communications multilingues augmentent également au cours de l'essor de mondialisation des entreprises chinoises. Dans ce contexte, notre groupe a choisi de classifier 4 langues des pays sur la «ceinture» et la «route» avec le modèle de CNN appris en cours.

Présentation du corpus

Notre corpus vient d'un concours de la reconnaissance automatique de la parole, organisée par une entreprise chinoise en 2020 (http://challenge.xfyun.cn/topic/info?type=multilingual). Les données sont stockées au format wav avec une fréquence d'échantillonnage de 16000 Hz et une quantification de 16 bits. La taille de chaque langue est de 329M à 721M, la taille totale du corpus est 2.15G. La durée moyenne de chaque langue est autour d'une heure, qui contient environ 250 phrases (environ 10s par enregistrement) pour la partie d'entraînement et 400 pour la partie de développement (environ 2s par enregistrement).

Le jeu de données se compose de 17 langues (hongrois, grec, serbe, slovaque, géorgien, croate, swahili, amharique, malais, philippin, khmer, bengali, turc, cinghalais, azerbaïdjanais, hébreu, et afrikaans zoulou). Considérant sa quantité et la limite de nos outils, nous avons décidé de choisir 4 langues parmi eux, soit le hébreu, le philippin, le khmer et l'azéri. Pour chaque langue, nous utilisons la partie d'entraînement et la partie de développement en dispersant la moitié du développement pour l'évaluation finale.

Méthodologie

I.traitement sur l'audio

Avant de convertir tous les audios en spectrogrammes, nous avons constaté trois problèmes importants: 1) une silence longue apparaît dans beaucoup d'enregistrement, qui pourrait perturber les traitements suivants; 2) chaque langue n'a qu'environ 450 enregistrements pour l'entraînement, qui a l'air insuffisant pour entraîner un réseau de neurones; 3) la durée de chaque enregistrement est longue (surtout ceux dans l'entraînement), par conséquent, les images générées directement de ces enregistrements pourraient être trop larges pour l'expérience du modèle par la suite.

Nous utilisons 2 stratégies pour résoudre ces problèmes:

Notre prétraitement commence par l'élimination des longues silences (VAD). Ensuite, nous avons généré des enregistrements supplémentaires pour l'entraînement en modifiant la vitesse de parole (0.9, 0.95, 1.05 et 1.1 par rapport à la vitesse originale). De cette manière, nous avons quintuplé nos données. À noter que la hauteur a aussi changé après cette manipulation.

Deuxièmement, à l'égard du découpage des enregistrements, nous utilisons la librairie **pydub**. Nous essayons d'abord de couper le son en fonction du silence. Mais cette méthode ne garantit pas une durée de moins de 2 secondes pour chaque morceau, en plus, il est difficile de définir la longueur et l'intensité sonore d'un silence à l'intérieur d'un discours. Par conséquent, nous décidons finalement de couper les enregistrements à une durée fixe de 2 secondes. Nous avons aussi ajouté un fade-in et un fade-out, ainsi qu'un silence de 100 ms avant et après chaque morceau, pour qu'il soit plus naturel.

Néanmoins, puisque le jeu de données final était trop large pour manipuler dans Google colab, nous avons dû supprimer une partie de données pour diminuer le temps de chargement dans l'expérience.

II.Conversion en spectrogrammes

Finalement, afin de générer les spectrogrammes, nous utilisons le logiciel **Praat** et le script fourni par M. Audibert. Le contenu et les paramètres de ce script ne sont pas touchés : le maximum fréquence de spectrogramme est fixé à 8000 Hz puisque les sons de parole sont généralement limité à ce seuil; la longueur du fenêtre est gardé à son standard 5 ms; et on utilise le gaussian qui évide les lobes secondaires comme notre *Window Shape*. Avec ces paramètres, nous obtenons des spectrogrammes se composant de 250 bandes de fréquence, avec une bande passante de 260 Hz, et distinguant proprement les valeurs entre -20 dB/Hz et 30 dB/Hz en niveaux de gris. Finalement, nous

écrivons un script python qui parcourt tout le répertoire et appelle l'exécution du script Praat dedans, permettant d'automatiser le travail.

Voir l'annexe **Tableau 1** pour la synthèse du jeu de données final de spectrogrammes.

III.Préparation de données pour le réseaux neurone

Premièrement, nous devons réduire les images de spectrogramme en array, puisque la taille initiale de ces images en array (1500*900) est coûteuse pour les démarches suivantes. Après des essais, la performance d'une taille de 50*30 n'était pas idéale, vu qu'elle perd beaucoup de détails, celle de 120*72 est similaire à celle de 100*60 et le dernier est moins coûteux que l'autre. Ainsi, on l'a choisi.

À noter qu'on a également désordonné les données pour éviter le sur-apprentissage.

IV.modèle cnn

Quant au modèle du réseau neurone, nous construisons un modèle CNN avec la libraire Keras selon l'exemple présenté en cours.

Nous avons mis 64 pour la taille de batch, cette petite taille nous permet d'entraîner le modèle plus rapidement avec un grand corpus. Pour le nombre de l'époque, nous en avons choisi 100 au début, et qu'il en résulte le sur-apprentissage. En constatant les courbes de perte et d'exactitude (voir le **figure 1** dans l'annexe) et essayant plusieurs fois, nous fixons sur un époque de 40.

Le modèle de l'entraînement est le modèle Séquentiel. Dans un premier temps, nous avons ajouté une couche de convolution, qui nous a permis d'avoir plusieurs features différentes. Le deuxième genre couche est la couche de changement non linéaire. Ici, nous avons choisi la fonction d'activation relu, celle la plus populaire. Après nous avons fait une couche de pooling qui nous a permis de garder les caractéristiques de l'image (le son) et de diminuer la taille de l'image. Pour avoir une bonne performance, nous avons répété plusieurs couches de pooling et couches de convolution. En plus, nous avons ajouté aussi quelques couches de dropout afin d'éviter la sur-apprentissage. Après tout ça, nous avons ajouté une couche de flatten pour faire les données bidimensionnelles aux données unidimensionnelles. À la fin, c'est la couche du dense.

Ensuite, nous avons compilé notre modèle. La compilation du modèle prend trois paramètres: l'optimiseur, la perte et les métriques. Afin d'améliorer le résultat, nous avons changé à l'optimiseur **Nadam**, qui introduit Nesterov dans Adam. Nesterov permet de prévoir le futur et fait un ajustement en avance.

À la fin, pendant l'entraînement de modèle, on introduit le **ReduceLROnPlateau** qui évolue le taux d'apprentissage quand les indices d'apprentissage changent. Il aide aussi à réduire le sur-apprentissage.

Résultat

Le résultat de notre réseau de neurones est idéal par rapport aux 4 langues que nous avons choisies, notamment pour reconnaître le hébreu et le philippin. Mais il est faible pour reconnaître le khmer et l'azéri (Voir la **figure 2** dans l'annexe dans Colab notebook). En plus, les courbes de perte et d'exactitude nous assurent que ce modèle est fitting (voir la **figure 3** dans l'annexe ou dans Colab notebook). Résultats historiques des expériences est montrés dans la dernière partie de l'annexe (Note des résultats).

Conclusion et difficultés

En conclusion, nous avons beaucoup appris dans ce cours le semestre dernier et à travers ce projet (du prétraitement de fichier audio à construire un modèle de réseau neuronal pour classifier les spectrogrammes).

Il y a encore une marge d'amélioration pour le résultat de notre travail. Si nous connaissons ces quatre langues, nous serons capables de faire les prétraitements plus fine sur les fichiers audio, par exemple, extraire et amplifier les traits (prononciation unique de certaine langue, par exemple) spécifiques de chaque langue et aider le modèle à les reconnaître.

La difficulté la plus grande que nous avons rencontrée est que la taille de notre corpus est trop grande pour l'ordinateur portable et il est extrêmement coûteux de télécharger ces données sur Google Colab.

Un autre problème qui nous prend beaucoup de temps à résoudre est le problème de sur-apprentissage (voir la **figure 1**). Au début de l'entraînement, il y avait un sur-apprentissage grave. Le taux d'exactitude était plus de 95%, mais celui de prédiction n'était que d'environ 30%.

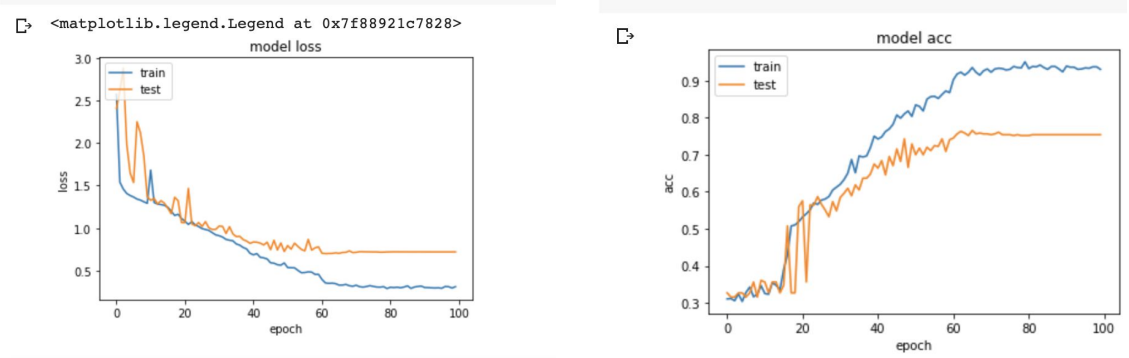
Afin de résoudre ce problème, nous avons mélangé l'ordre de données d'entraînement et d'évaluation, introduit le ReduceLROnPlateau qui permet la décroissance du taux d'apprentissage et changé le nombre d'époques. Nous avons aussi essayé d'ajouter le regularizer dans le modèle. Il a pourtant conduit au sous-apprentissage.

Annexe:

Tableau 1: Taille de spectrogramme du corpus

Langue	Taille d'entraînement	Taille d'évaluation
Hébreu	3978 items/2,48 G	398 items/200 MB
Philippin	4106 items/2,49 GB	395 items/204 MB
Khmer	4151 items/2,65 GB	397 items/203 MB
Azéri	5658 items/3,43 GB	395 items/180 MB

Figure 1 Courbe de perte et courbe d'exactitude

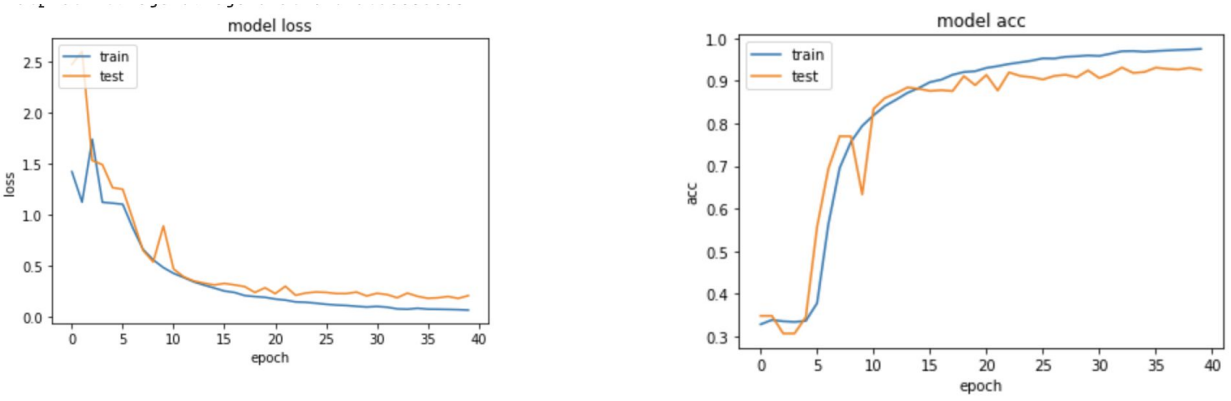


Ces deux courbes montrent que le sur-apprentissage commence au 20eme époque.

Figure 2 Précision, Rappel et F1-score de quatre langues

	precision	recall	f1-score	support
0	0.6056	0.4332	0.5051	397
1	0.6009	0.6650	0.6313	394
2	0.7027	0.3939	0.5049	396
3	0.4886	0.6532	0.5590	395

Figure 3 Courbe de perte et courbe d'exactitude du modèle final



Notes des résultats:

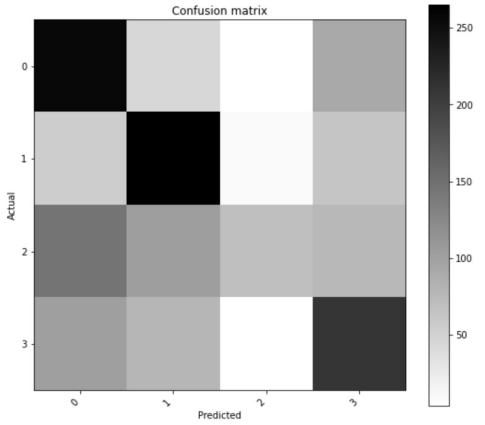
Après plusieurs lancements, les trois suivants sont les meilleurs résultats（cependant les 2 premiers ne sont pas dans le notebook final):

Résultat 1:

Tableau
Précision, Rappel et F1-score de trois langues

Langues	Précision	Rappel	F-mesure
Hébreu	0,5755	0,5088	0,5401
Philippin	0,5341	0,6751	0,5964
Khmer	0,6151	0,3914	0,4784
Azéri	0,4641	0,4582	0,4611

Figure Matrice de confusion

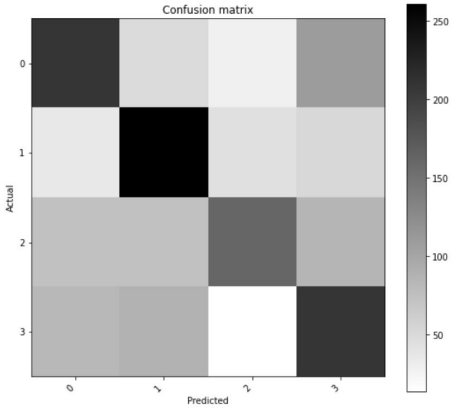


Résultat 2:

Tableau
Précision, Rappel et F1-score de trois langues

Langues	Précision	Rappel	F-mesure
Hébreu	0,5399	0,5113	0,5252
Philippin	0,5682	0,6345	0,5995
Khmer	0,6652	0,3712	0,4765
Azéri	0,4749	0,5266	0,4994

Figure Matrice de confusion



Résultat 3:

Tableau
Précision, Rappel et F1-score de trois langues

Langues	Précision	Rappel	F-mesure
Hébreu	0,6056	0,4332	0,5051
Philippin	0,6009	0,6650	0,6313
Khmer	0,7027	0,3939	0,5049
Azéri	0,4886	0,6532	0,5590

Figure Matrice de confusion