Méthode par apprentissage de lecture sur les lèvres CNN+LSTM

Timothé MAZARD

Table des matières

Annexe

I. Introduction

Ce projet d'apprentissage profond a pour but de mettre en valeur le fait qu'une machine est capable par visualisation d'interpréter les mouvements des lèvres. Cette méthode ce base sur l'assemblage d'un réseau de neurones (CNN) et réseau récurrent un LSTM où on retrouve la connexion de retour d'information. Ce dernier peut traiter aussi bien les données isolées telles des images ou un enregistrement audio, qu'une séquence ou une vidéo au complète. Cette application sera de manière générale destinée à la reconnaissance vocale ou bien à la reconnaissance d'écriture manuscrite. Il y sera question dans ce projet de plusieurs grandes phases. Tout d'abord, le traitement de données est un point crucial car ici nous traitons des images. Par la suite une phase de test est mise en place en passant par le transfert d'apprentissage, nous reviendrons sur ce point ultérieurement.

II. Contexte

Pour ce qui est des données celle-ci ont été extraite sous format mp4 par un smartphone. Une meilleure qualité et une meilleure extraction des caractéristiques des lèvres ce sont fait ressentir. Les caractéristiques des lèvres sont au format jpg. Nous voulons être en mesure de pouvoir authentifier les chiffres allant de 0 à 9 avec la meilleure précision. Mais cela ne veut pas que cette méthode s'arrête ici. Ce domaine voudrait faire en sorte que des informations en temps réels soit analyser afin de faire de la retranscription. Selon, l'Organisation Mondiale de la Santé plus de 5% de la population souffre de mal audition. Les solutions auditives médicales ne sont pas en mesures de résoudre entièrement les problèmes de ces personnes. Dans de telles situations la reconnaissance vocale par l'articulation des lèvres et un moyen de contourner ce problème. Il sera peut-être plus tard de traduction vocale d'une langue à une autre en temps réelle.

D'après l'article de Garg et al [5], ils comparent différentes méthodes afin de voir les plus adéquates à ce genre de problème. Dans l'une des méthodes ils utilisent un LSTM pour extraire l'information de temporalité mais les résultats ne sont pas relevés pertinent. Ils pensent que c'est dû au manque d'entrainement en plus du fait que celui-ci soit entrainés après les cartes de caractéristiques. Cependant ici nous allons nous servir de MobilNet qui est un modèle beaucoup plus compact et donc demande moins de temps d'apprentissage. Un espoir de lueur, se fait cependant ressentir dans l'article de Pyateava et Dzyuba [4], le modèle comporte un CNN basé sur MobilNet et un LSTM. Ils obtiennent un score moyen de 68% de précision basé sur sept mots en russe. Nous pourrions voir si celui-ci performe mieux que celui avec VGG d'autant plus que nous ne connaissons pas le modèle VGG utilisé dans l'article précédent.

III. Objectifs

Les objectifs ont su évoluer depuis la proposition de projet. Ici Nous n'utiliserons plus Lipnet qui n'était plus vraiment au goût du jour. Une évolution de ce modèle est quand même à prévoir. De plus aucune version officielle n'est disponible sur les librairies python. Nous allons concevoir un modèle basé sur MobilNet disponible avec Keras.

Les objectifs sont toujours à l'ordre du jour, le traitement des vidéos afin d'isoler les lèvres se fait par la méthode de Viola Jonas qui utilise les caractéristiques de Haar, figure 1 dans l'annexe. Cette

méthode extrait des caractéristiques sans l'utilisation de ressources abondantes, algorithme facile à mettre en place. Grâce à ces filtres ils donc possible en passant par la librairie Open Cv d'extraire la région d'intérêt de la bouche. Une image représentative se trouve en annexe. Plusieurs images sont extraites et cela à pour but que le LSTM garde une trace temporelle des informations et puisse en juger sur le choix de la classe à laquelle cette séquence d'images appartient en se basant uniquement sur les informations visuelles et non plus sur les informations auditives comme il en était encore le cas dans la proposition de projet.

Le deuxième objectif porte sur l'entrainement d'un modèle ainsi que sur résultats. Pour ce qui est du modèle celui-ci va être pré entrainé sur Imagenet. Base de données regroupant des centaines de milliers d'images de toute sortes allant de l'humain aux roues de vélo. Nous allons concevoir un modèle basé sur MobilNet puis un LSTM prend place après les couches denses du CNN, annexe 3. Le transfert learning se fait sur les couches denses car ont juges les données similaires à celles aux données originales et nous avons ici peu de données. Cependant si des courbes anormales se font ressentir dans les données de pertes entre la validation et d'entrainement alors il sera à reconsidérer les données d'entrainement comme non similaire aux données originales c'est-à-dire celles présentent dans ImageNet. Cela engendrera un apprentissage non pas seulement des couches denses de MobileNet, évoquant la phase de classification mais aussi les quelques dernières couches du réseau à convolution. Pour affiner l'apprentissage au type de données traités

Afin de déterminer si le modèles a bien appris les classes. Il sera plus judicieux que d'utiliser une simple méthode de précision il est plus adéquate de partir sur des score WER et/ou BLEU qui sont des mesures de scores de prédiction de bon assemblage des mots afin de former une phrase. Pour ce projet il est question de mots, il nous vaudra vérifier s'il est possible d'ajuster ces méthodes pour notre problème. Mais ces métriques répondent à la façon dont un humain évaluerait le même texte, le constat sera beaucoup plus représentative de la vérité sur la prédiction de phrase.

IV. Méthodes

Dans cette partie il sera question de la collection des données car ces données ne proviennent pas de données trouvées dans des bases de données mais bien des données extraites à partir de mon smartphone. Je me suis rendu compte que la qualité de la caméra embarquée dans l'ordinateur ne suffisait pas à la collection des images des lèvres, collection de zone d'intérêt aléatoire, non représentatif des lèvres.

Pour la collection des données, nous utilisons des données vidéos pré-enregistrées mais il est tout à fait possible d'utiliser directement des séquences d'images provenant d'autres sources comme CNN ou base de données dans ce domaine. J'ai voulu choisir mon propre jeu de données car je voulais avoir des classes spécifiques, suite des chiffres de 0 à 9. D'ailleurs, il difficile de se procurer réellement des données comme des séquences d'images de lèvres. Néanmoins, il est tout à fait possible de créer son jeu de données à partir de ces vidéos. Environ 7 vidéos par classe sont créés pour chaque individu. Soit 140 vidéos au total, pour le moment seulement deux visages de personnes ont été filmés. Ces vidéos ont été enregistrés au format MP4, le format AVI avait été suggéré mais aucun codec vidéo installé de base sur windows 10. Par la suite la méthode de cascade de HAAR est utilisé afin d'extraire la région d'intérêt qui est la bouche. Cette méthode est incluse dans la librairie de OpenCV. Elle utilise le classifier de bouche, se composant des caractéristiques de HAAR nécessaire à la détection de cette dernière. Des méthodes de 'Backproject' et de 'Meanshift', ce sont des techniques qui nous ont permis d'éviter les erreurs de détection de zone d'intérêt. Sans rentrer dans

les détails la première méthode permet de faire ressortir les caractéristiques saillantes d'une image et l'autres de pouvoir corriger le pas de déplacement ce qui minimise le fait que la zone d'intérêt initialement sur la bouche passe sur l'œil. L'œil ayant des caractéristiques similaires que la bouche pour la méthode de HAAR.

Ces images sont enregistrées dans le format jpg, un format compressant les données mais conçu pour le traitement de données. Les images des lèvres sont enregistrées dans un dossier avec une taille de 64 x 64 afin de pouvoir passer en images d'entrées de MobilNet. Cependant, il a été relevé sur le site officiel de Keras[8] que mobilNet supporte toute taille d'image supérieure à 32 x 32, en revanche des tailles plus grandes offres de meilleures résultats. Des ajustements seront sûrement à considérer. En tout cela ne fait pas moins de 30 500 images enregistrées qui vont être passées en revue par le CNN+LSTM. Les données sont ensuite stocker dans cette hiérarchie.

- Lips
    - Labels
        - 1
        - 2
            - vidéo
                - Jpg images

Ensuite vient la tâche de création de jeu de données pour se faire nous utilisons ImageDataGenerator de Keras pour générer les données avec les images d'entrées en spécifiant la taille des images d'entrées mais aussi la taille du batch. Nous utilisons Image DataGenerator pour modifier les images d'entrées avec des rotation, des cisaillements et des flips haut/bas, gauche/droite. Cela permet de fortifier l'apprentissage des données d'entrainement. Le code est fourni en Annexe pour la création du Dataset. Les labels sont enregistré au format One Hot c'est-à-dire une représentation vectorielle binaires. Si nous avons 3 labels le label 2 est au format [0,1,0].

Le modèle a été créé en suivant la lecture [4] de Pyataeva et Dzyuba. C'est-à-dire qu'au modèle de base MobilNet nous rajoutions une couche Dense de 1024 suivit d'une autre couche Dense de 128 avec pour activation ReLu. Cette couche sera l'entrée du LSTM. Qui lui va faire un lien temporel entre chaque image de la vidéo et cela grâce aux connexions faites avec les données d'images précédentes. De plus il a été démontré dans [3] développé par Karan Shrestha que l'augmentation du Dropout procure de meilleur mais aussi la taille du kernel

Après cette phase de modélisation viendra la visualisation des résultats pour se faire les pertes ainsi que la précision du modèle pourront être affiché ainsi qu'une courbe ROC représentant le True Positive Rate/ False Positive Rate. C'est un moyen de mesure de performance, qui peut être amené à être vu lors de phase de présentation des résultats.

Mais aussi nous verrons d'autres outils de performance, plus à même d'établir une idée claire sur cette tâche de prédiction de mots, qui sont les score WER et BLEU. Relevant non pas des scores de TP, FP, TN et FN mais plutôt de comment le modèle a pu identifier le mot et ce avec quel degré d'exactitude.

Le WER [7] est la méthode la plus utilisée dans la reconnaissance vocale. Il mesure la différence entre deux mots. Il prend en considération aussi bien les mots insérés, supprimés que substitués. Si nous fonctionnons avec un seul mot il sera alors question du score CER score sur les lettres qui agit avec cette même méthodologie.

$$\text{Word Error Rate} = \frac{\text{Inserted + Deleted + Substituted}}{\text{Total words in transcript}}$$

V. Résultats

Pour le moment aucun résultat concret n'a réellement été démontré. Seulement les phases de traitement de données et de modélisation du modèle sont aboutis. Il reste une phase importante et pas des moindres, c'est la phase de validation qui permettra de valider ou non l'approche faite par [4] stipulant que l'association de MobilNet et d'un LSTM permette d'obtenir des scores favorables d'au moins 68% en générale sur la détection de mots. Or, il sera tout aussi intéressant de se pencher également sur le travail fait par Grag et al [5], mettant en évidence que RNN tel le LSTM obtient de meilleur résultat que la combinaison du CNN et du LSTM. Un de leur graphique de comparaison est en Annexe. Par la suite si le temps est en notre faveur, il sera envisageable de traité des données avec des phrases qui sont pour moi beaucoup plus pertinent que des chiffres. Quelques difficultés ont été rencontrés que la création du modèle sur Keras, utilisant une ancienne version de MobilNet. Il sera judicieux d'essayer les nouvelles versions qui peuvent apporter notamment des améliorations de compatibilités avec Python mais aussi des gains en performance et en temps.

VI. Conclusion

En ce jour, il est difficilement concevable d'avoir une idée claire quant aux résultats. Il est vrai que nous avons une idée de ce que peuvent être les résultats. Mais le fait d'avoir préparer soi-même la collection de données, peut apporter une autre source d'erreur. Les données n'ont peut-être pas été prises dans les meilleures conditions extérieures (source de lumière, contre-jour, saturation de l'images, ROI) pouvant avoir des coïncidences sur les résultats. Mais restons optimiste sur la suite du déroulement. Le forage des données a été produite par la méthode Haar Cascade qui offre des résultats recevables avec quelques ajustements faits par les méthodes de Backproject et MeanShift. Il aurait été tout à fait possible de les extraire grâce aux landmarks points d'intérets pour le tracking de visage. Toutefois Haar Cascade était beaucoup plus maitrisé pour ma part.

Grâce à l'api de Keras il a été assez simple de modéliser un modèle reprenant les caractéristiques de celui voulant être traité au cours de ce projet. Tout est à disposition pour assembler différentes couches les unes avec les autres de manières élémentaires.

Nous remercions K. Ditsch d'avoir sacrifié de son temps pour la collection de données.

Références

[1] G. T. Themos Stafylakis, «Combining Residual Networks with LSTMs for Lipreading,» 2017.

[2] C. K. H. J. Y. WoMa, «Read My Lips,» 2019.

[3] K. Shrestha, «Lip Reading using Neural Network and Deep learning».

[4] A. D. Anna Pyataeva, «Artificial neural network technology for lips reading,» 2021.

[5] J. N. S. B. Amit Garg, «Lip reading using CNN and LSTM,» 2015.

[6] M. Tyagi, «Towardsdatascience - Viola Jones Algorithm and Haar Cascade Classifier,» 13 Juillet 2021. [En ligne]. Available: https://towardsdatascience.com/viola-jones-algorithm-and-haar-cascade-classifier-ee3bfb19f7d8. [Accès le 10 juillet 2022].

[7] K. Doshi, «Towardsdatascience,» 9 Mai 2021. [En ligne]. Available: https://towardsdatascience.com/foundations-of-nlp-explained-bleu-score-and-wer-metrics-1a5ba06d812b. [Accès le 10 Juillet 2022].

[8] «Keras API,» [En ligne]. Available: https://keras.io/api/applications/mobilenet/.

ANNEXES

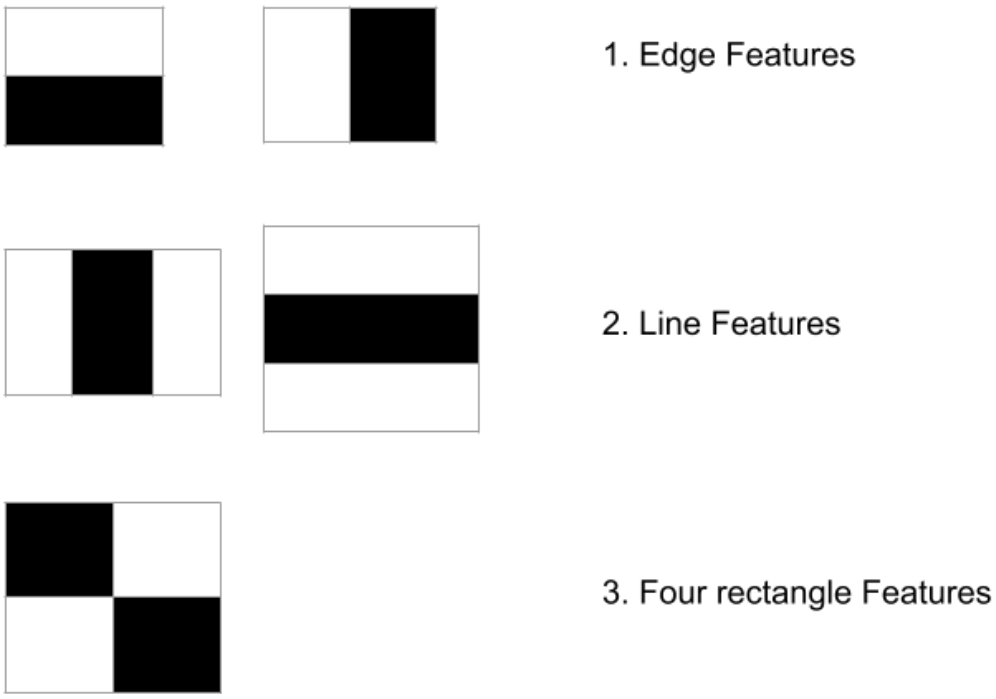

Figure 1 Filtres de Haar

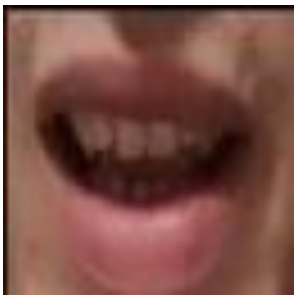

Figure 2 Exemple ROI lèvres

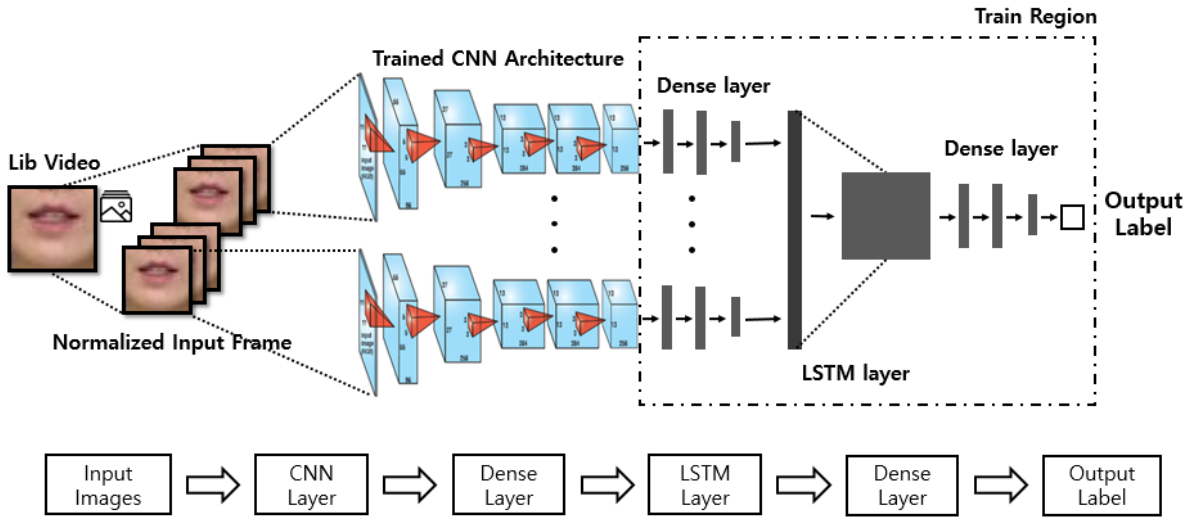

Figure 3 Modèle MobilNet LSTM

```
# Load Dataset
for i in range(n_labels):
    nb = 0
    # Counting datasets in each labels
    for root, dirs, files in os.walk('normalized cascade/dataset/' + str(i+1)): # set directory
        for name in dirs:
            nb = nb + 1
    print(i,"Label number of Dataset is:",nb)
    TotalNb = TotalNb + nb
    # by Counting size, cross subfolder and read image data, resize image, and append list
    for j in range(nb):
        temp = []
        for k in range(timesteps):
            name = 'normalized cascade/dataset/' + str(i+1) + '/' + str(j+1) + '/' + str(k+1) + '.jpg'
            img = cv2.imread(name)
            res = cv2.resize(img, dsize=(img_col, img_row), interpolation=cv2.INTER_CUBIC)
            temp.append(res)
        label.append(i)
        data.append(temp)
print("Total Number of Data is",TotalNb)

# Convert List to numpy array, for Keras use
Train_label = np.eye(n_labels)[label] # One-hot encoding by np array function
Train_data = np.array(data)
print("Dataset shape is",Train_data.shape, "(size, timestep, column, row, channel)")
print("Label shape is",Train_label.shape,"(size, label onehot vector)")
```

Figure 4 Création du Dataset

```
# declare data for training and validation, if you want, you can seperate testset from this
X_train=Train_data[0:TotalNb,:]
Y_train=Train_label[0:TotalNb]
# declare input layer for CNN+LSTM architecture
video = Input(shape=(timesteps,img_col,img_row,img_channel))
# Load transfer learning model that you want
model = applications.MobileNet(input_shape=(img_col,img_row,img_channel), weights="imagenet", include_top=False)
model.trainable = False
# FC Dense Layer
x = model.output
x = Flatten()(x)
x = Dense(1024, activation="relu")(x)
x = Dropout(0.3)(x)
cnn_out = Dense(128, activation="relu")(x)
# Construct CNN model
Lstm_inp = Model(input=model.input, output=cnn_out)
# Distribute CNN output by timesteps
encoded_frames = TimeDistributed(Lstm_inp)(video)
# Contruct LSTM model
encoded_sequence = LSTM(256)(encoded_frames)
hidden_Drop = Dropout(0.3)(encoded_sequence)
hidden_layer = Dense(128, activation="relu")(encoded_sequence)
outputs = Dense(n_labels, activation="softmax")(hidden_layer)
# Contruct CNN+LSTM model
model = Model([video], outputs)
```

Figure 5 Génération du modèle fondé sur Keras