DataScientest • com

Rapport Technique d'évaluation

PenPyText

Promotion : DS avril 2021

Participants :

Samy AIT-AMEUR
Sophie AMEDRO
Stéphane TCHATAT

Contexte

Nous étions trois coéquipiers de profils différents et complémentaires à la réalisation de ce projet. Tous trois, nous sommes en reconversion et étions novices en Deep Learning au commencement de PenPyText. Ce projet n'a donc pas pu s'inscrire dans le cadre d'une insertion métier à l'heure actuelle. Nous l'avons choisi pour les raisons suivantes:
Samy " Faire apprendre à un ordinateur à lire, est un grand pas dans le domaine de l'intelligence artificielle"
Sophie " Ancienne enseignante en GS-CP, j'apprenais à lire à des enfants. Ce projet était un nouveau challenge car ici il fallait apprendre à lire à un ordinateur. "
Stéphane " La reconnaissance des textes manuscrits est une problématique majeure dans les domaines bancaires "

La technique d'OCR (*optical character recognition*) permet de **situer** et de **reconnaître les chaînes de caractères** dans une image et donc de faire la conversion des mots qui peuvent ensuite être utilisés pour faire une recherche plein texte.

Ce sujet est aussi l'opportunité pour nous d'avoir un aperçu sur les points de vue suivants :

- ☐ *Du point de vue technique :*

Il existe plusieurs solutions d'océrisation sur le marché. Elles sont soit open-source, soit payantes. En réalisant à partir de rien une solution d' OCR, ce projet nous a permis de comprendre l'architecture des modèles de deep learning utilisés pour répondre à cette problématique, cela a nécessité l'adaptation et la modification des outils existants afin de permettre l'entraînement de nos différents modèles.

- ☐ *Du point de vue économique :*

Avec l'OCR, un grand nombre de documents papier peuvent être numérisés en texte lisible par la machine, peu importe la langue et le format dans lesquels ils sont rédigés. Cette technique facilite non seulement le stockage, mais rend disponibles des données qui auparavant étaient difficilement accessibles. Si elle est correctement implémentée, elle ouvre la possibilité à l'automatisation d'un certain nombre de processus au sein de l'entreprise, ce qui permettrait de faire des économies. Par exemple, automatiser à 100% le traitement des chèques est un avantage financier pour tout le monde : le débiteur, la banque et le créditeur.

- ☐ *Du point de vue scientifique :*

Comprendre les effets des différentes couches d'un réseau de neurones (Conv2D, MaxPooling, Dropout), ce qu'est la zone de réception d'un neurone. Comment un réseau de neurones extrait les informations d'une image.

Objectifs

Le projet PenPyText pose la problématique complète et ambitieuse de la reconnaissance de textes manuscrits. Au cœur des objectifs de digitalisation des entreprises, la capacité à convertir une image numérisée d'un texte manuscrit en un document textuel exploitable par une application informatique est une brique essentielle. Il devient alors possible de digitaliser un plus grand nombre de documents, et d'automatiser de nouveaux processus.

Les enjeux sont multiples tels que la protection des données d'entreprise, en permettant des détecter de potentielles fuites de données sensibles via des notes manuscrites, ou encore l'automatisation de la saisie des formulaires papiers que nous sommes encore régulièrement amenés à remplir.

En plus de son usage applicable à de nombreux domaines, sa mise en œuvre nécessite de maîtriser plusieurs savoir-faire essentiels de la Data Science, que nous avons vu tout au long de notre formation.

Ainsi, les principaux objectifs posés pour ce projet sont les suivants :

- L'exploitation d'un jeu de donnée important, comportant comme pour tout jeu de données, des erreurs ou malformations,
- La mise en oeuvre de différentes techniques de pré-traitement des images,
- La construction d'un réseau de neurones avec les contraintes techniques du jeu de données, mais aussi de la problématique abordée (la reconnaissance de textes manuscrits),
- L'interprétation et l'analyse des résultats intermédiaires et finaux
- La restitution d'un modèle et la présentation des résultats.

Les différents profils du groupe ont trouvé dans ce projet l'opportunité d'une belle complémentarité. En effet, chacun a pu apporter une contribution significative en fonction de son expérience et de ses objectifs, mais aussi de développer les sujets connexes à chaque contribution :

- Samy, grâce à son expertise en mathématiques, a notamment étudié et implémenté certains filtres appliqués aux images initiales et étapes intermédiaires du modèle de deep learning construit,

- Sophie, avec son expérience en développement et en informatique en général a pu travailler sur l'implémentation en mettant en oeuvre des algorithmes efficaces pour préparer le jeu de données et générer le modèle rapidement,

- Stéphane, avec son expérience en gestion de projet, a également travaillé sur l'implémentation du modèle et sur l'analyse des nombreux documents techniques liés au projet et à sa problématique.

Au-delà des membres du groupe, nous avons sollicité notre mentor pour qu'il nous apporte son expertise sur les modèles les plus pertinents pour répondre à la problématique, mais aussi pour nous aider à apprécier la qualité des résultats que nous avons obtenus tout au long de notre progression. Ce sujet ne portant pas sur un cas d'usage spécifique, et pour éviter de restreindre sa portée, nous n'avons pas jugé nécessaire d'entrer en contact avec des experts métiers. Cela pourrait au contraire faire l'objet d'une étude sur l'exploitation des résultats de ce projet sur un domaine spécifique car les entreprises dans lesquelles nous travaillons ou avons travaillé ne traitent pas ces problématiques.

Data

Cadre

Pour réaliser ce projet, nous avons utilisé le jeu de données **IAM Handwriting Database 3.0** disponible à l'adresse suivante: https://fki.tic.heia-fr.ch/databases/iam-handwriting-database

La base de données contient des formes de **texte anglais** de natures variées, écrites à la main sans contrainte, qui ont été scannées à une résolution de 300 dpi et enregistrées sous forme d'images PNG avec 256 niveaux de gris.

Les données sont disponibles gratuitement. Une inscription sur le site suffit pour y avoir accès. Cette base de données ne peut être utilisée qu'à des fins de recherches non commerciales.

Caractéristiques de la IAM Handwriting Database 3.0:

- 657 personnes ont participé à la rédaction des fichiers manuscrits
- 1 539 pages de texte scannées
- 5 685 phrases isolées et labellisées
- 13 353 lignes de texte isolées et labellisées
- 115 320 mots isolés et labellisés

Nous avons restreint notre modèle à la reconnaissance de mots. Les volumétries pour ce périmètre sont les suivantes:

- Dataset : 115 320 mots
  - Données d'entraînement initial (trainset) : 53 839 mots
  - Données de validation 1 (validationset 1): 7 899 mots
  - Données de validation 2 (validationset 2): 8 566 mots
  - Données de test (testset) : 17 616 mots

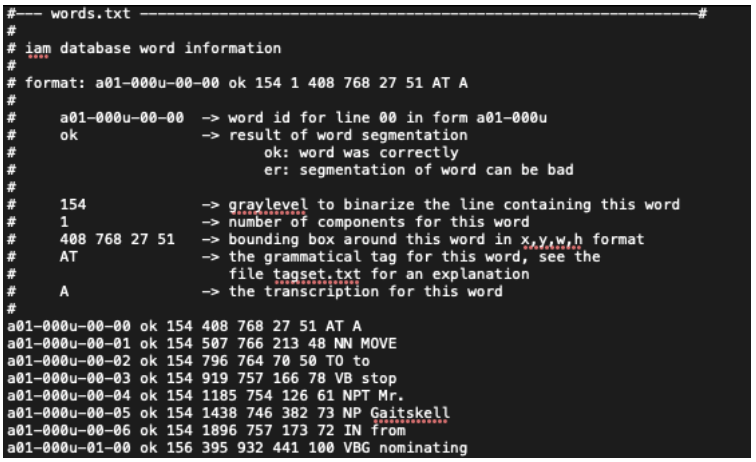
Pertinence

Le jeu de données IAM Handwriting est composé d'images et de documents .txt contenant divers informations sur les images ainsi que nos labels.
Le processus de préparation des données s'est déroulé en trois étapes principales:

1. Traitement des informations dans les documents textes,
2. Préparation des images,
3. Analyse et transformation des labels

1. ***Traitement des documents textes***

   Cette base de données étant destinée à la recherche, certaines informations contenues dans les documents étaient non pertinentes pour notre problématique.
   Nous avons donc recréé une base de données adaptée à notre situation, sous forme de dataframe. Nous avons ainsi supprimé les informations telles que l'étiquette grammaticale, la boîte de délimitation,...



```
#--- words.txt ------------------------------------------------------------#
#
# iam database word information
#
# format: a01-000u-00-00 ok 154 1 408 768 27 51 AT A
#
#     a01-000u-00-00  -> word id for line 00 in form a01-000u
#     ok              -> result of word segmentation
#                            ok: word was correctly
#                            er: segmentation of word can be bad
#
#     154             -> graylevel to binarize the line containing this word
#     1               -> number of components for this word
#     408 768 27 51   -> bounding box around this word in x,y,w,h format
#     AT              -> the grammatical tag for this word, see the
#                        file tagset.txt for an explanation
#     A               -> the transcription for this word
#
a01-000u-00-00 ok 154 408 768 27 51 AT A
a01-000u-00-01 ok 154 507 766 213 48 NN MOVE
a01-000u-00-02 ok 154 796 764 70 50 TO to
a01-000u-00-03 ok 154 919 757 166 78 VB stop
a01-000u-00-04 ok 154 1185 754 126 61 NPT Mr.
a01-000u-00-05 ok 154 1438 746 382 73 NP Gaitskell
a01-000u-00-06 ok 154 1896 757 173 72 IN from
a01-000u-01-00 ok 156 395 932 441 100 VBG nominating
```

Figure 1: Fichier words.txt

	word_id	ok_err	gray_level	transcript	data_path	line_id
0	a01-000u-00-00	ok	154	A	data/words/a01/a01-000u/a01-000u-00-00.png	a01-000u-00
1	a01-000u-00-01	ok	154	MOVE	data/words/a01/a01-000u/a01-000u-00-01.png	a01-000u-00
2	a01-000u-00-02	ok	154	to	data/words/a01/a01-000u/a01-000u-00-02.png	a01-000u-00
3	a01-000u-00-03	ok	154	stop	data/words/a01/a01-000u/a01-000u-00-03.png	a01-000u-00
4	a01-000u-00-04	ok	154	Mr.	data/words/a01/a01-000u/a01-000u-00-04.png	a01-000u-00
5	a01-000u-00-05	ok	154	Gaitskell	data/words/a01/a01-000u/a01-000u-00-05.png	a01-000u-00
6	a01-000u-00-06	ok	154	from	data/words/a01/a01-000u/a01-000u-00-06.png	a01-000u-00
7	a01-000u-01-00	ok	156	nominating	data/words/a01/a01-000u/a01-000u-01-00.png	a01-000u-01
8	a01-000u-01-01	ok	156	any	data/words/a01/a01-000u/a01-000u-01-01.png	a01-000u-01

Figure 2: dataframe

2. ***Traitement des images***

Une première analyse des images nous a permis de constater que la qualité et la taille de celles-ci n'étaient pas égales.

Certaines images sont une composition de fragments de mots, les outils scripteurs utilisés sont variés. Afin de pouvoir identifier les caractères et les mots, il est nécessaire d'effectuer un pré-traitement des images.

Pour cette étape, nous avons réalisé les opérations suivantes:

- ***Retirer du jeu de données les images endommagées:*** deux images du dataset sont endommagées et retirées (a01-117-05-02.png et r06-022-03-05.png)
- ***Nettoyage du bruit*** : pour éliminer le bruit nous utilisons un filtre Gaussien qui permet une meilleure préservation des bords.
- ***Harmoniser les tailles des images:*** l'image de plus grande surface est de 1156×238 pixels. Nous avions choisi, dans un premier temps, d'harmoniser toutes les images sur ces dimensions. Le nombre de pixel étant très important, l'entraînement d'un modèle est alors très gourmand en temps et en mémoire. Nous avons alors choisi de redimensionner les images en 128x32 pixels. Ces dimensions correspondent à environ 10% de la taille initiale, ce qui permet de limiter la déformation des écritures et d'avoir des dimensions en puissances de 2 qui permettent aux librairies utilisées d'exploiter pleinement la mémoire des cartes GPU disponibles.

- ***Érosion :*** dans le cadre d'un système d'océrisation de texte manuscrit c'est une étape nécessaire. Nous sommes dans une situation où les textes ont été écrits par différents auteurs avec des outils scripteurs variés. Ils ont donc des écritures différentes, et donc une largeur de trait différente. Cette étape d'érosion permet d'uniformiser cette largeur de trait (plus mince) et de conserver uniquement le squelette des lettres manuscrites.
- ***Images en noir et blanc*** : la binairisation permet de faire la différence entre le fond et les écritures. On utilise le niveau de gris seuil de nos données, qui diffère selon les images, pour effectuer cette opération.

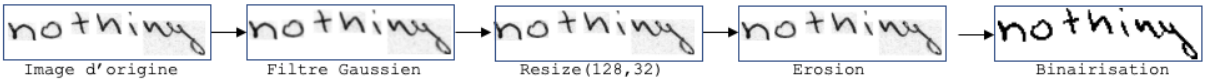


Figure 3: Étapes de pré-processing des images

- ***Retirer du trainset les images avec erreur de segmentation*** : certaines images de notre jeu de données sont mal segmentées par rapport à leur labels. En d'autres termes, l'image n'est pas cadrée avec le mot transcrit. Nous avons fait le choix de les retirer du jeu de données d'entraînement.(Exemple Image comportant une erreur de segmentation)

transcript : will

Figure 4: Image a01-011x-03-04.png comportant une erreur de segmentation

3. ***Analyse et Transformation des labels***

Notre objectif est de retranscrire des écritures manuscrites.
Les variables explicatives sont le chemin d'accès vers une image qui contient un mot manuscrit, ainsi que le niveau de gris de l'écriture.
La variable cible de notre jeu de données est donc la variable transcript que nous avons extraite du document words.txt.

Le graphe suivant nous montre la distribution du nombre de caractères pour les transcriptions.

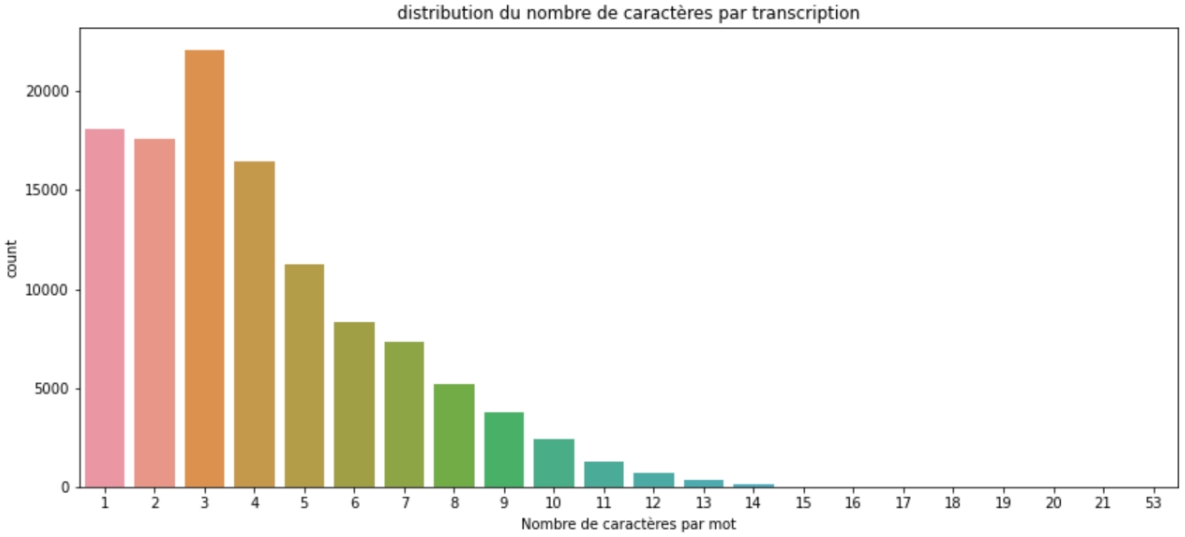


Figure 5 : Distribution de nombre de caractère par transcription

À la lecture de ce graphe, nous remarquons que notre jeu de données est composé d'un grand nombre d'images contenant jusqu'à 4 caractères. Nous pouvons imaginer que les modèles entraînés sur ce jeu de données reconnaîtront plus facilement des transcriptions jusqu'à 4 caractères.

Toujours à la lecture de ce graphe, nous pouvons constater quelques particularités:

- Il y a un outlier de 53 caractères '----------' (image p02-109-01-00.png)
- Le nombre de mots disponibles décroît fortement au fur et à mesure que la taille du mot augmente (97,65 % des mots ont moins de 10 caractères) . Ce qui rend plus difficile d'entraîner le modèle à la reconnaissance de mots longs. Pour optimiser les performances d'un modèle d'apprentissage, il serait judicieux de réaliser une augmentation des données sur les mots de plus de 10 caractères.

Regardons à présent une autre particularité de notre jeu de données, les erreurs de segmentation. Comme vu précédemment (paragraphe Traitement des images, page précédente), certaines images ont été mal segmentées par rapport à leur transcription.
Regardons la répartition de ces erreurs sur le graphe suivant

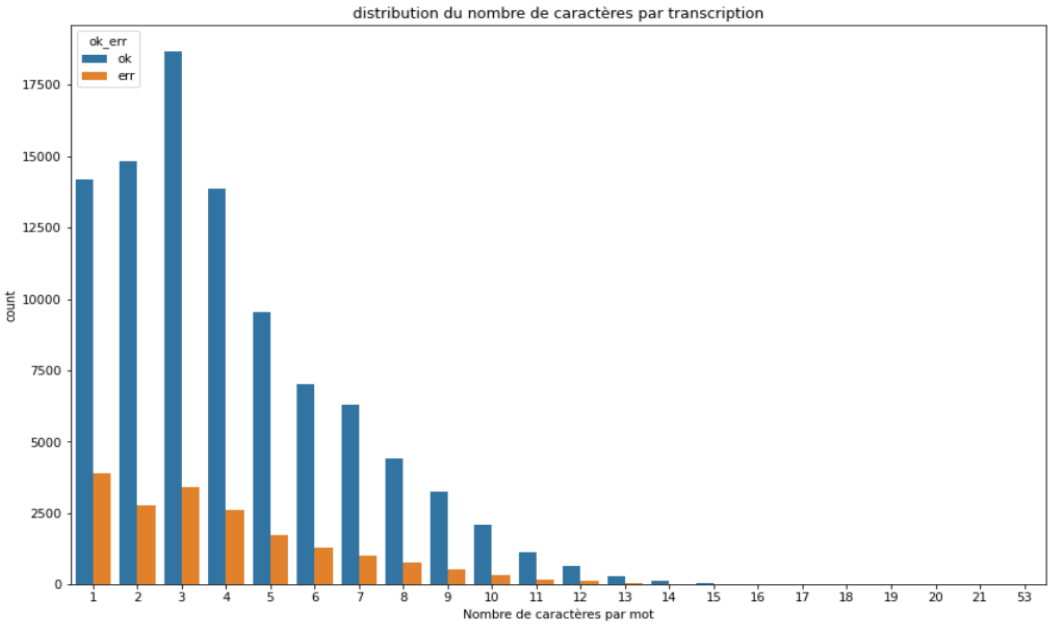


Figure 6: Répartition des images avec erreur de segmentation

Notons que les erreurs de segmentation sont présentes principalement sur les mots de moins de 4 caractères (voir ci-dessus). Comme il s'agit des données les plus représentées de notre dataset, nous pouvons retirer de notre jeu d'entraînement les images avec erreurs de segmentation sans créer de biais supplémentaires. Ce qui permettra ainsi à notre modèle de s'entraîner sur des images sans erreurs.

Nous avons analysé notre variable cible, mais il reste un problème: un modèle ne pourra pas utiliser nos labels sous la forme de chaîne de caractères, il va falloir transformer ces labels.

Tokenization et vectorisation des labels

Plusieurs solutions s'offraient à nous (CountVectorizer, Tfidfvectorizer,...), celles qui ont retenu notre attention furent Tokenizer et LabelEncoder.

- Dans le cadre de notre premier modèle de type CNN, nous avions choisi d'encoder les transcriptions entièrement. Nous avons utilisé la fonction **Tokenizer** de keras qui permet de vectoriser un corpus de mots en nombres entiers. Les avantages de cette fonction sont de pouvoir récupérer les transcriptions grâce à un dictionnaire créé par la fonction, et le coefficient pour chaque jeton est basé sur le nombre de mots.
- Dans le cadre de notre second modèle de type CRNN, nous devions encoder les transcriptions par caractères, c'est-à-dire que chaque transcription doit être un vecteur et tous les vecteurs de même taille. Nous avons donc utilisé la liste des caractères imprimables, que nous avons réduite en retirant des caractères qui n'apparaissent pas dans notre jeu de données (comme des accents qui n'existent pas dans la langue anglaise, @, |, …). Le **labelEncoder** (sklearn.preprocess) a été ensuite entraîné sur cette liste de caractères, pour transformer ensuite chaque caractère de nos labels.

  Cependant, notre modèle a besoin de vecteurs de dimensions égales et ces derniers ont des tailles variées, nous avons alors complété arbitrairement ces vecteurs par un nombre entier très grand et non utilisé par labelEncoder.

  À noter, que nos modèles CRNN sont entraînés sur des transcriptions de 10 caractères maximum, comme remarqué précédemment nous avions trop peu de données sur des transcriptions plus grandes.

Projet

Classification du problème

La reconnaissance de mots, hors contexte, comme il est question ici s'apparente à un problème de classification sur un grand nombre de classes.

Le principe général de la reconnaissance de caractères passe par l'extraction de plusieurs caractéristiques uniques. Ainsi la comparaison des caractéristiques du texte manuscrit avec celles des caractères appris permet la reconnaissance.
Pour extraire les caractéristiques des images nous utilisons un réseau de neurones. Nous avons utilisé deux types de modèles: un modèle de type CNN puis un modèle de type CRNN.

La performance d'un modèle de classification dépend directement de la métrique utilisée pour l'évaluer. Nous avons traité notre problème de classification de deux manières.

Dans un premier temps, nous avons tenté de reconnaître les mots dans leur globalité à l'aide d'un dictionnaire extrait de notre jeu de données d'entraînement. Ainsi, soit le mot était était correctement retranscrit, soit il ne l'était pas. Nous avons donc utilisé l'accuracy comme métrique.

Dans un second temps, comme nous étions limités par le nombre de mots de notre dictionnaire, nous avons vu les mots comme un ensemble de caractères. Aussi, notre OCR devait reconnaître chaque caractère avant de les assembler. Nous nous sommes alors retrouvés avec des transcriptions qui n'avaient parfois aucun sens. On a alors utilisé une métrique complémentaire à l'accuracy, la CER (Character Error Rate).

Nous avons décidé d'appliquer la CER au top 1 des prédictions de notre modèle pour évaluer notre output d'OCR. On définit 3 types d'erreurs:

- S : Substitution error : un caractère mal identifié
- D : Deletion error : un caractère manquant
- I : Insertion error : inclusion d'un caractère

La CER calcul le nombre minimum d'opérations sur les caractères (S, D, I) qui permet de passer du label de notre mot à notre output d'OCR.
La formule se présente ainsi:
CER = (S+D+I)/N où N est le nombre de caractères dans le label de notre mot.
On obtient la CER en faisant une moyenne sur l'ensemble du dataset.

Notons que la CER est une métrique qui ne peut être utilisée que pour comparer les modèles CRNN dans notre cas, puisque le CNN reconnaît les mots entiers.

Au cours de nos entraînements, nous nous sommes aperçus que la prédiction la plus probable ne correspondait parfois pas à notre label, alors que les prédictions suivantes, si. C'est pourquoi, nous avons choisi de regarder deux types d'accuracy : celle de la première prédiction proposée par le modèle et celle sur le top 5 des prédictions.

Choix du modèle & Optimisation

Les réseaux de neurones sont à ce jour les meilleurs outils pour traiter ce genre de problème, les modèles de machine learning sont trop coûteux en temps, mémoire et sont limités à des dictionnaires de mots.

Nous avons traité deux approches des réseaux de neurones autour de la classification:

- ➔ Dans un premier temps, nous avons choisi des mots entiers comme classes, où chaque mot distinct de notre jeu d'entraînement correspondait à une classe. Nous avions ainsi grâce à notre jeu d'entraînement un dictionnaire de 8004 mots. Un modèle de **réseau de neurones convolutifs** suffisait à extraire les caractéristiques des images et à en obtenir une classification.

  Ce type de modèle était suffisant dans un premier temps pour comprendre l'extraction de caractéristiques d'images. Mais pour une reconnaissance efficace d'un mot dans sa globalité, il nous aurait fallu un vocabulaire beaucoup plus grand que celui dont nous disposions. En effet, la langue anglaise est composée de plus de 200000 mots sans compter la ponctuation et les chiffres, notre vocabulaire en comptait moins de 5%.

- ➔ Nous avons donc poursuivi notre démarche par la reconnaissance de caractères grâce à un modèle de **réseau de neurones récurrents** associé à un algorithme de rétro-propagation.

Présentation des modèles retenus

Les figures ci-dessous montrent une description des modèles retenus par l'équipe.

Les résultats de ces modèles sont visibles dans les notebook Evaluation CNN.ipynb et Evaluation RNN.ipynb, disponibles sur le dépôt Github avec l'ensemble des codes.

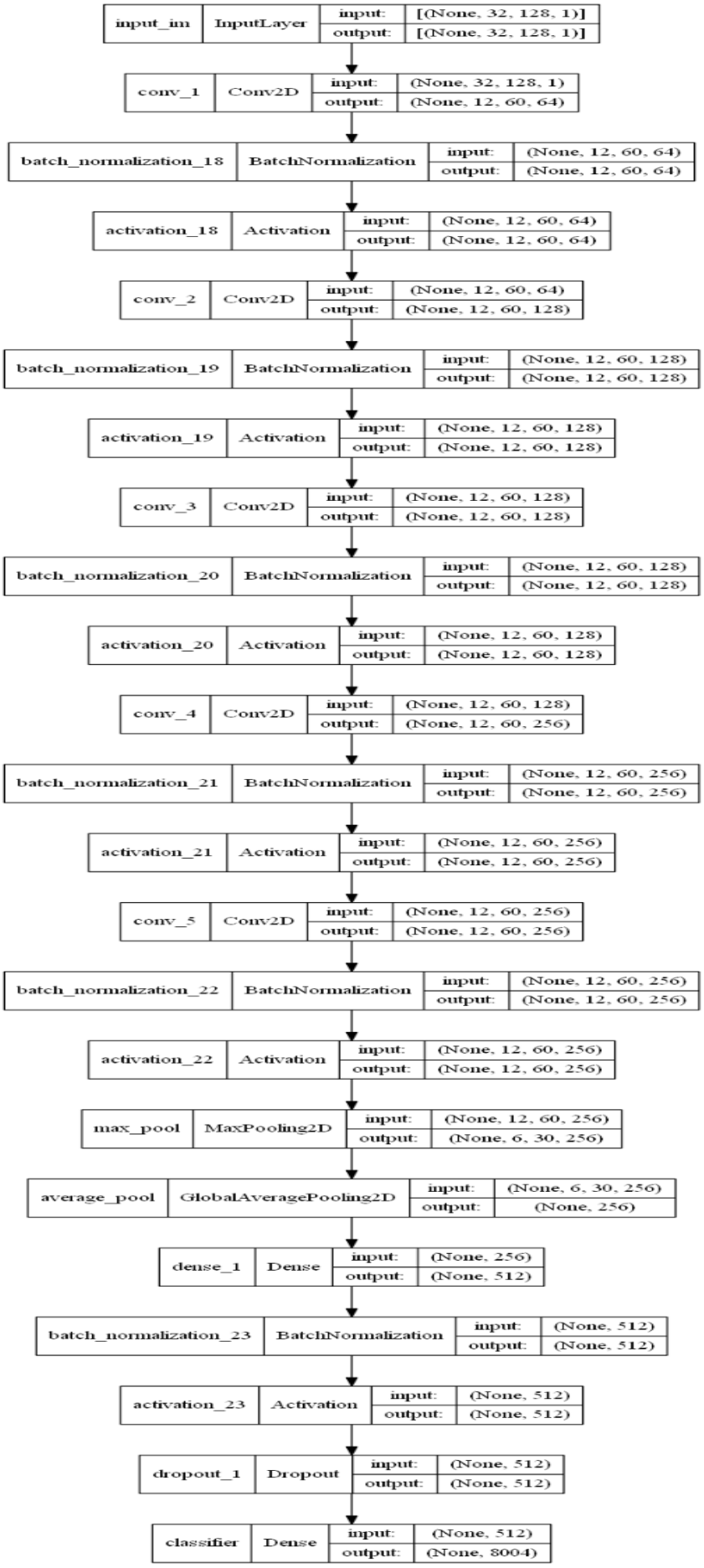


Figure 7: description du modèle CNN retenu

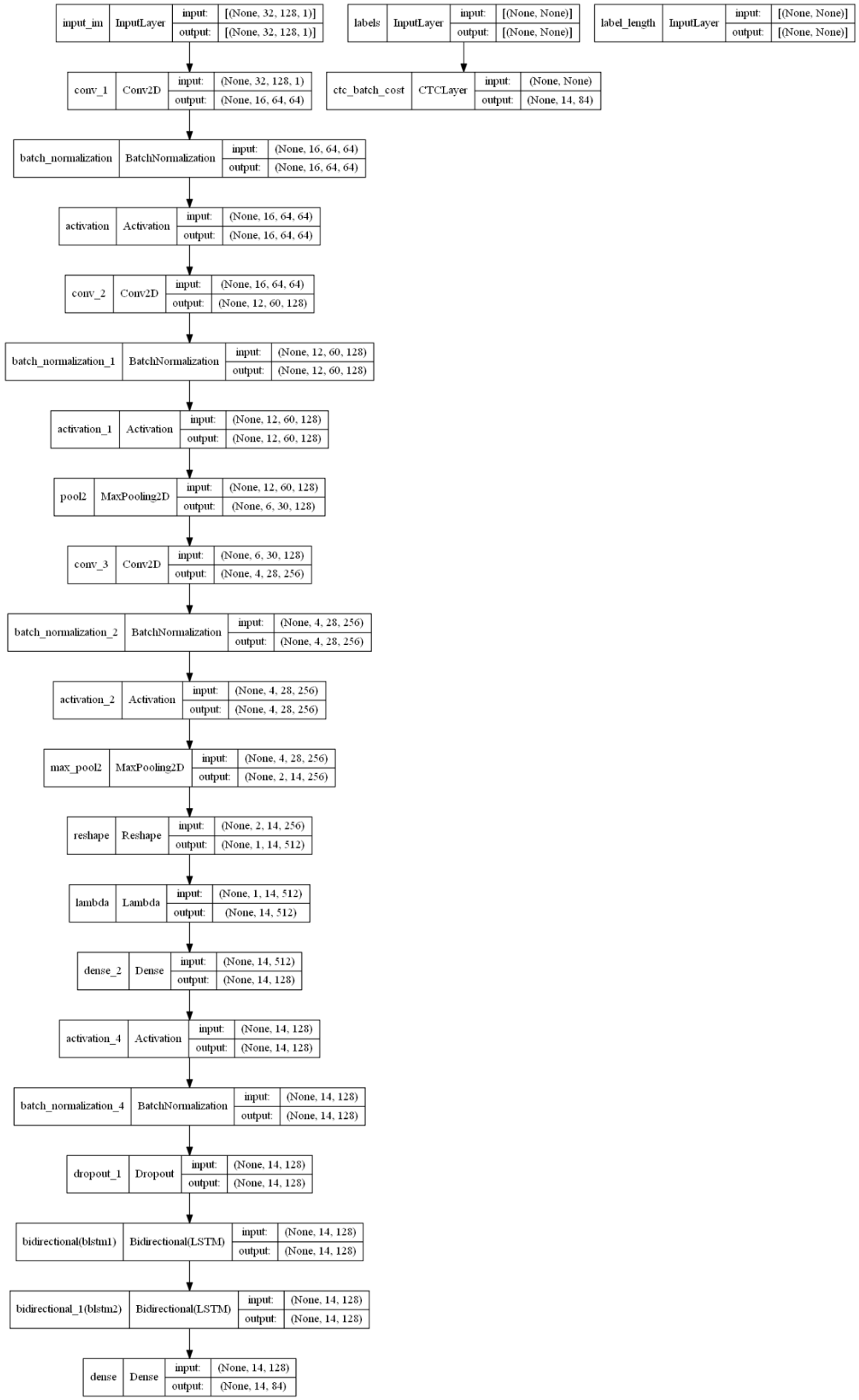


Figure 8: description du modèle CRNN retenu

Comparaison de différents modèles testés

Les modèles entraînés ci-dessous dans les conditions suivantes:

- Équipement: Apple Mac mini M1 2020, équipé de 16 Go de mémoire, 8 coeurs CPU et 8 coeurs GPU
- Nombre d'epochs = 25
- Taille des batchs = 100
- trainset composé des images répertoriées trainset et validationset1
  validationset est composé des images répertoriées validationset2
  testset sont les images répertoriées testset dans le dossier 'largeWriterIndependentTextLineRecognitionTask'

Tableau de comparaison de quelques modèles CNN (nombre de classes 8004)

Layers	temps par epochs	accuracy
Conv(k=(9,9,s=(2,2),64) **Conv(k=(5,5),128)** **Conv(k=(3,3),128)** **Conv(k=(3,3),256)** **Conv(k=(3,3),256)** **Dense(512)**	**215 s**	**pred 1: 0.224852** **top_5: 0.596446**
Conv(k=(3,3),64) Conv(k=(3,3),128) Conv(k=(3,3),128) Conv(k=(3,3),256) Conv(k=(3,3),256) Dense(512)	1100 s	pred 1: 0.114612 top_5: 0.385786
type vgg16 Conv(k=(3,3),64) MaxPool((2,2), s=(2,2)) Conv(k=(3,3),128) *2 MaxPool((2,2), s=(2,2)) Conv(k=(3,3),256) *3 MaxPool((2,2), s=(2,2)) Conv(k=(3,3),512) *3 MaxPool((2,2), s=(2,2)) Conv(k=(3,3),512) *3 Dense(4096) Dense(4096)	300 s	pred 1:0.21787 top_5: 0.599569
type lenet Conv(k=(5,5),32) MaxPool((2,2)) Conv(k=(3,3),64) MaxPool((2,2)) GlobalAveragePooling Dense(512) Dense(1024)	35 s	pred 1:0.086796 top_5: 0.315622
Conv(k=(9,9),s=(3,3,64) Conv(k=(5,5),128) Conv(k=(3,3),256) Conv(k=(3,3),256) Dense(1024) Dense(2048)	726 s	pred 1: 0.135502 top_5: 0.40696

Tableau de comparaison de quelques modèles CRNN

Layers	max_len	temps par epochs	accuracy	CER moyenne sur pred 1
Conv(k=(3,3),64) MaxPool((2,2)) Conv(k=(3,3)128) MaxPool((2,2)) Conv(k=(3,3),256) MaxPool((2,2)) Dense(256) BLSTM(128) BLSTM(64)	10	760 s	top: 0.395833 top_5: 0.495453	0.4001
Conv(k=(3,3),64) MaxPool((2,2)) Conv(k=(3,3)128) MaxPool((2,2)) Conv(k=(3,3),256) MaxPool((2,2)) Dense(256) **GRU**(128) **GRU**(64)	10	780 s	top: 0.414192 top_5: 0.51715	0.3780
Conv(k=(9,9),s=(2,2),64) Conv(k=(5,5)128) MaxPool((2,2)) Conv(k=(3,3),256) MaxPool((2,2)) Dense(256) Dense(128) BLSTM(64) BLSTM(64)	10	760 s	top: 0.432666 top_5: 0.52866	0.3653

Certains modèles pouvaient nous faire gagner 1 ou 2 % de réussite mais le temps d'entraînement était doublé, celui-ci étant déjà conséquent. Nous avons choisi des modèles quasi-équivalents en performances mais plus intéressant au niveau temps d'exécution.

La décision de retirer les images ayant des erreurs de segmentation du jeu de données d'entraînement fût bénéfique aussi à une échelle moins importante.

Nous pouvons remarquer que les accuracy sont proches entre notre modèle CNN et les modèles CRNN. Le modèle CRNN retient toute notre attention car il n'est pas limité par un lexique mais fait des prédictions caractère par caractère.

Analyse des erreurs et améliorations

- Sur le modèle CNN:
  - La principale erreur que l'on peut identifier concerne les mots qui ne sont pas dans notre dictionnaire d'origine. Avec ce type de modèle, une écriture manuscrite dont les caractéristiques n'ont pas été répertoriées, ne peuvent pas être transcrites correctement.
    Pour améliorer notre modèle, il nous faudrait un jeu de données beaucoup plus important avec une augmentation de la taille de notre vocabulaire. Ce type de modèle est intéressant lorsque l'on cherche à retranscrire un lexique spécifique avec un nombre de mots de ce fait limité, par exemple dans le cadre d'une analyse de sentiments. Notre problématique n'imposant pas de

limite lexicale, cela impliquait pour nous les limites de ce type de modèle. C'est pourquoi nous nous sommes orientés vers un modèle avec réseaux de neurones récurrents.

La première amélioration notable entre les deux types de modèle est sur la première prédiction. Alors que sur le modèle CNN on obtient une accuracy qui peine à atteindre 20%, avec un modèle CRNN elle est de 40%.

La seconde amélioration, c'est qu'il n'y a plus de limite de vocabulaire. Tout mot peut être transcrit.

- Sur le modèle CRNN:
  - Au cours des premiers entraînements, nous avons vu une progression des résultats en affinant la partie CNN du modèle. Nous avons alors étudié les erreurs et nous avons remarqué que les mots de plus de 4 lettres n'étaient pas prédits.

    Cela avait probablement pour cause une plus faible représentation des mots plus longs dans le dataset. Nous avons alors choisi de nous concentrer sur des mots de maximum 10 caractères. Cela a amélioré d'environ 10% notre accuracy sur les cinq premières prédictions.
  - Nous nous sommes ensuite concentrés sur les erreurs qui concernaient les mots de moins de cinq caractères. Ces mots étaient très bien représentés dans notre jeu de données et pourtant, nous avions encore beaucoup d'erreurs sur ces mots.

    Il s'est avéré qu'une grande partie des erreurs observées avaient lieu sur des images mal segmentées. Nous les avons alors retirées du jeu de données d'entraînement, car cela provoque un biais pour notre modèle. L'accuracy de notre modèle CRNN a alors subi une amélioration de l'ordre de 5%.
  - Réduire la taille du lexique de caractère en supprimant les caractères non présents dans notre jeu de données, nous a permis de gagner en accuracy.

Nous avons identifié une dernière erreur au niveau de la préparation de nos données images. Nos machines n'étant pas très puissantes, nous avions fait une première étape de pré-processing pour laquelle nous avions harmonisé la taille des images selon la plus grande surface. Nous avions alors construit un nouveau jeu d'images sans bruit, toutes de dimensions égales 1156x238. L'harmonisation de la taille, s'opérait par une fonction qui calcule le ratio pour ne pas déformer les images et comblait le fond par des blancs. Au cours de nos entraînements suivants, pour des problèmes de mémoire nous avons redimensionné ces images mais cela a provoqué des déformations. En reprenant le processus de préparation des images pour le rendu du projet, nous nous sommes rendu compte de notre erreur. Nous avons alors redimensionné directement nos images en 128x32, ce qui a permis d'améliorer notre modèle CNN de façon considérable (de 4% d'accuracy nous sommes passé à plus de 40%) et notre CRNN a obtenu une performance légèrement améliorée.

D'une façon générale, nous avons utilisé les ressources de Stackoverflow, ainsi que des exemples de code disponibles sur Github pour résoudre des problèmes d'implémentation.

Description des travaux réalisés

Répartition de l'effort sur la durée et dans l'équipe

Initialement nous étions quatre personnes pour ce projet. Après le rendu du rapport initial, nous nous sommes retrouvés à trois. Dans l'optique d'apprendre le plus de techniques possibles, nous avons fait le choix que chacun participe à toutes les tâches du projet.

Les étapes du projet PenPyText peuvent être schématisées ainsi:

- Etude de la problématique
- Preprocessing des images
- Création des dataframes train, test et validation
- Implementation de plusieurs modèles (CNN, Random Forest, CRNN Gru,CRNN LSTM,..)
- Implémentation de la fonction de coût (loss)
- Evaluation
- Livrables du projet

Le diagramme de Gantt en annexe, permet de visualiser le temps passé sur les différentes tâches et la participation de chaque membre de l'équipe.

Difficultés rencontrées lors du projet

Au commencement de ce projet, nous étions tous les trois novices dans le domaine et n'avions pas d'application métier directe. Nous avons rencontré des difficultés de différents types:

- ❖ Connaissances scientifiques:
  - ➢ Compréhension du fonctionnement des réseaux de neurones
  - ➢ Fonctionnement d'un réseau **Long short-term memory (LSTM) et d'un Gated Recurrent Unit (GRU)**
- ❖ Gestion du temps:
  - ➢ Nous avons passé beaucoup de temps à améliorer le modèle CNN, alors que nos modèles étaient entraînés sur des images déformées.
  - ➢ Trouver une fonction de coût qui optimise l'entraînement de notre model
- ❖ Étapes de pré processing:
  - ➢ Écrire une fonction qui génère les données pour le CRNN (image, label et longueur des labels)
- ❖ Compétences techniques et théoriques:
  - ➢ Acquérir les techniques de deep learning qui étaient proposées en fin de formation.
  - ➢ Obtenir un CRNN entraînable
  - ➢ Utiliser une fonction de coût CTC destinée à identifier des labels dans des séquences de données temporelles.
- ❖ Capacité et performance des équipements informatiques
  - ➢ La puissance de nos machines ne permettait pas à tous les membres de l'équipe d'entraîner des modèles.

Bilan

Ce projet a été un véritable challenge pour tous les membres de l'équipe. Il a été très instructif, aussi bien en termes de programmation, qu'en termes de préparation des données d'imagerie. Nous avons acquis des bases solides sur le fonctionnement des réseaux de neurones et les différentes couches qui le composent, ainsi que la fonction de coût et les données temporelles.

Le projet PenPyText portait sur la reconnaissance de caractères manuscrits. Nous sommes plutôt satisfaits de nos résultats, car nous avons réussi à implémenter un CRNN et à l'entraîner. Nous avons programmé notre propre générateur de données pour le CRNN. Enfin, notre accuracy est correcte pour un premier projet d'OCR car proche des 50% dans les 5 premières prédictions, sur le jeu de données test.

Aujourd'hui la reconnaissance optique de caractère facilite la recherche et la manipulation de données et peut apporter une aide confortable pour les personnes malvoyantes.
En permettant la reconnaissance de caractères, nous pourrons alors numériser des documents manuscrits, afin de réaliser des recherches plein texte, de classer les documents selon leur type, date, auteur afin de rendre plus accessible de vieux documents écrits à la main. Ce qui pourrait être utile dans le cadre de recherche en histoire, en architecture, en art…

Les techniques utilisées pour extraire les propriétés des images peuvent s'appliquer dans de nombreux domaines comme l'imagerie médicale, la numérisation de documents juridiques, bancaires ou de tout type afin de rendre les données plus facilement accessibles.

Suite du projet

Nous avons encore plusieurs pistes pour améliorer les performances de modèle CRNN comme:

- Entraîner notre modèle sur des mots de longueur maximum plus grande (jusqu'à 21 pour notre dataset) afin d'améliorer les capacités de notre modèle.
- L'augmentation de données sur les mots plus longs
- L'utilisation de la CER dans notre entraînement pour surveiller les distances
- Reconnaître les caractères individuellement puis validation par reconnaissance de lexicale.
- Utiliser le transfer learning

Bibliographie

[1] U. Marti and H. Bunke. The IAM-database: An English Sentence Database for Off-line Handwriting Recognition. Int. Journal on Document Analysis and Recognition, Volume 5, pages 39 - 46, 2002

[2] Build a Handwritten Text Recognition System using TensorFlow, juin 2018
https://towardsdatascience.com/build-a-handwritten-text-recognition-system-using-tensorflow-2326a3487cd5

[3] Get started with deep learning OCR, avril 2020
https://towardsdatascience.com/get-started-with-deep-learning-ocr-136ac645db1d

[4] Deep Learning avec Keras et TensorFlow, Aurélien Géron, 2020
Build a Handwritten Text Recognition System using TensorFlow | by Harald Scheidl | Towards Data Science

[5] Understanding Stateful LSTM Recurrent Neural Networks in Python with Keras, by Jason Brownlee on July 28, 2016 in Deep Learning
Understanding Stateful LSTM Recurrent Neural Networks in Python with Keras

[6]Understanding LSTM Networks, colah's blog, posted on August 27, 2015
Understanding LSTM Networks -- colah's blog

[7] IAM-OnDB - an On-Line English Sentence Database Acquired from Handwritten Text on a Whiteboard, Marcus Liwicki and Horst Bunke,
https://www.researchgate.net/publication/224626482_IAM-OnDB_-_An_on-line_English_sentence_database_acquired_from_handwritten_text_on_a_whiteboard

[8] CTCModel: a Keras Model for Connectionist Temporal Classification by Yann Soullard, Cyprien Ruffino, Thierry Paquet
CTCModel: a Keras Model for Connectionist Temporal Classification

[9] Keras.Conv2D Class - Understand params
Keras.Conv2D Class - Understand params

[10] An Intuitive Explanation of Connectionist Temporal Classification, 2018
https://towardsdatascience.com/intuitively-understanding-connectionist-temporal-classification-3797e43a86c

[11] Creating a CRNN model to recognize text in an image (Part-1), (part-2)
Creating a CRNN model to recognize text in an image (Part-1) | TheAILearner
Creating a CRNN model to recognize text in an image (Part-2) | TheAILearner

[12] Practical Machine Learning and Image Processing: For Facial Recognition, Object Detection, and Pattern Recognition Using Python, Apress, 2019

[13] [TensorFlow 2 / Keras] Comment exécuter l'apprentissage avec CTC Loss dans Keras
[TensorFlow 2 / Keras] Comment exécuter l'apprentissage avec CTC Loss dans Keras

[14] Reconnaissance de l'écriture manuscrite avec des réseaux récurrents, Luc Mioulet, 2016
https://hal.archives-ouvertes.fr/tel-01301728

[15] Réglage des hyper-paramètres du réseau neuronal avec Keras Tuner et Hiplot,
https://ichi.pro/fr/reglage-des-hyper-parametres-du-reseau-neuronal-avec-keras-tuner-et-hiplot-129980331205114

[16] Réaliser son propre générateur, cours pratique, Bastien Maurice, décembre 2020
https://deeplylearning.fr/cours-pratiques-deep-learning/realiser-son-propre-generateur-de-donnees/

[17] Image Augmentation for Deep Learning using Keras and Histogram Equalization
https://towardsdatascience.com/image-augmentation-for-deep-learning-using-keras-and-histogram-equalization-9329f6ae5085

[18] TextVectorization layer with keras
https://keras.io/api/layers/preprocessing_layers/text/text_vectorization/#textvectorization-class89b

[19] CRNN (CNN+RNN) for OCR using Keras / License Plate Recognition
https://github.com/qjadud1994/CRNN-Keras/tree/58f57507fd0f05d12a6c2123dba7049c7210289b

[20] ImageDataGenerator – flow method
https://theailearner.com/2019/07/06/imagedatagenerator-flow-method/

Annexes

Diagramme de gantt

Description des fichiers de code

DIAGRAMME DE GANTT

TITRE DU PROJET	PenPyText		Datascientest
EQUIPE DE PROJET	1 - Samy AIT-AMEUR 2 - Sophie AMEDRO 3 - Stéphane TCHATAT		DS avril 2021

Description des fichiers de code

objectif code	Fichier de code	définition
PRE PROCESSING	words_txt_to_df.py	Lecture du fichier words.txt et transcription des données utiles dans un dataframe.
	nettoyage_fichiers.py	Retire du jeu de données les images qui ont des erreurs de lecture.
	harmonisation.py	Redimensionne les images, applique une érosion et nettoie le bruit.
	traintestsplit.py	Répertorie les ID images en trois groupes (entraînement, validation et test) selon les données du fichier LargeWriterIndependentTextLineRecognitionTask.txt
	preprocess_cnn.py	Applique le preprocess des images (features), encode les labels(target) et les met sous la forme d'un dataset tensorflow utilisable pour un modèle CNN.
	keep_n_chars.py	Supprime du dataset les données dont la transcription est composée de plus de n caractères
	generator_rnn.py	Générateur de données pour modèle RNN
MODÉLISA-TION	cnn_model.py	Construit un modèle CNN de classification d'images pour obtenir leur transcription
	cnn_pred.py	Prédictions du CNN (5 meilleures retenues) et décodage des prédictions. Construction d'un dataframe pour retour des prédictions avec ID de l'image associée
	main_cnn.py	Méthode d'exécution du modèle CNN qui comprend l'ensemble des étapes du preprocess aux prédictions
	rnn_model.py	Construit la fonction de perte CTC et le modèle CRNN pour obtenir la transcription des écritures manuscrites
	rnn_pred.py	Prédictions du CRNN (5 meilleures retenues) et décodage des prédictions. Construction d'un dataframe pour retour des prédictions avec ID de l'image associée
	main_rnn.py	Méthode d'exécution du modèle RNN qui comprend l'ensemble des étapes du preprocess aux prédictions
	evaluation.py	Fonctions de vérification si prédictions sont dans le top 5, visualisation images/prédictions et calcul de la CER
ETUDES	Visualisation des données.ipynb	Notebook d'étude du jeu de données initial
	Evaluation CNN.ipynb	Notebook d'évaluation du modèle CNN retenu
	Evaluation RNN.ipynb	Notebook d'évaluation du modèle CRNN retenu
Non résolus	generator_rnn_tf.py	Générateur de données pour modèle RNN avec outils tensorflow. Non utilisé car erreur non résolue (voir erreur en fin de code)
	cer_callback.py	Créer un rappel pour surveiller les distances d'édition (en cours)