الجمهورية الجزائرية الديمقراطية الشعبية
République Algérienne démocratique et populaire

وزارة التعليم العالي والبحث العلمي
Ministère de l'enseignement supérieur et de la recherche scientifique

جامعة سعد دحلب البليدة
Université SAAD DAHLAB de BLIDA

كلية التكنولوجيا
Faculté de Technologie

قسم الإلكترونيك
Département d'Électronique

Mémoire de Master

Filière Électronique
Spécialité Électronique des Systèmes embarqués

présenté par

BENNACER Aimen Souheib

&

RABAHI Irfane-Abderrachid

Détection et Reconnaissance de caractères par l'algorithme de deep learning EAST sous l'OCR Tesseract

Proposé par : Naceur Djamila

Année Universitaire 2019-2020

Remerciements

"Nous remercions Allah, le tout puissant de nous avoir aidé à accomplir ce travail, et qui a été avec nous à tous les moments de notre chemin d'étude. A nos chers parents pour leur Patience, leur Amour, leur Soutien et leur Encouragement. Nos sincères remerciements à tous les membres du jury qui nous ont fait l'honneur de réviser ce travail. Un remerciement spécial à Madame Djamila NACEUR pour sa patience, son aide et ses conseils pour élaborer ce travail. "

Résumé

La détection et la reconnaissance de texte est un champ d'étude qui a longtemps intéressé la communauté des chercheurs. Le problème de la détection et de la reconnaissance d'un texte a de nombreuses solutions en utilisant différentes techniques. L'une des solutions utilisées et développées récemment est l'utilisation de l'apprentissage profond (deep learning) qui a permis d'obtenir d'excellents résultats et une grande précision. Dans ce travail, nous allons effectuer à la fois la détection de texte et la reconnaissance de texte en utilisant des méthodes de traitement d'images traditionnelles avec open CV sous python, des nouveaux algorithmes d'apprentissage profond (deep learning) et des moteurs d'OCR, l'ensemble appliqué à des images de scènes naturelles.

Mots clés : *Détection de texte ; Reconnaissance de texte ; Apprentissage profond ; openCV ; moteurs OCR.*

ملخص

تحديد النص في الصورة والتعرف عليه هو موضوع اهتم به الباحثين طويلا ، ومشاكل تحديد النص في الصورة والتعرف عليه لها العديد من الحلول باستخدام العديد من تقنيات واحدة من التقنيات التي يتم استخدامها وتطويرها مؤخرًا هي الذكاء الاصطناعي (التعلم العميق) و التي بدورها ساعدة في الوصول إلى نتائج مرضية ودقة كبيرة في تحديد و معرفة النص ، سنقوم في هذا العمل بإجراء عملية التحديد و التعرف على النص باستخدام الطرق التقليدية (معالجة الصور) باستخدام open CV في python و خوارزميات التعلم العميق و محركات التحديد و الكشف عن النص OCR

كلمات مفتاحية: تحديد النص ; التعرف على النص ; التعلم العميق ; open CV ;محركات التحديد والتعرف عن النص OCR

Abstract

The text detection and recognition is a subject that interested the community of researchers for a long time, the problem of detecting and recognizing a text have many solutions using different techniques one of the solutions that are being used and developed recently is the use of deep learning which helped reach great results and accuracy, in this work we will we will perform both text detection and text recognition using traditional image processing methods with open CV under python and new deep learning algorithms and OCR engins.

Optical Character Recognition (OCR) is the process of extracting text from an image. The main purpose of an OCR is to make editable documents from existing paper documents or image files. Significant number of algorithms is required to develop an OCR and basically it works in two phases such as character and word detection. In case of a more sophisticated approach, an OCR also works on sentence detection to preserve a document's structure.

Character recognition has been an active area of research. Over time, this field has grown and with the advent of new advances, it has appeared much more interesting. Computers will still take many decades, if they ever succeed, to read all documents with the same precision as humans. This domain mainly exploits deep learning and recurring and memory networks. This field of artificial intelligence is however a little less mature than that of identifying people and objects.

Early versions were to be trained with images of each letter and used on one font at a time. Current systems are capable of producing a high degree of recognition accuracy for most fonts and support a variety of inputs for image or document file formats.

Research continues to advance in this niche and the growing need for systems automation has affected the development of text detection and recognition from images to a large extent. Text recognition has a wide range of applications, each with challenges and complications. Localization and separation of text, non-textual information is the biggest problem in the operation procedure given the variety of entries we can have, ranging from a simple picture of a book in perfect lighting and circumstances to an image of a natural scene with so many imperfections.

OCR tools have been developed in a range of field-specific applications, including recognition of receipts, invoices, checks, legal documents, etc.

However, we are a long way from fully automating the validation of handwritten characters, as each person's writing style is almost unique.

On the other hand, the application of deep learning methods applied to OCR writing allows us to be optimistic about hoping to automate check processing 100%.

The Applications of this domain give us a several advantages:

A reduction in check processing time is a financial advantage for everyone: the debtor, the bank and the creditor.

We can save our scanned file as .doc, .rtf, .txt, pdf, etc. after converting our scanned file to readable text. We can easily make these files available by including them in an appropriate database.

We can easily make changes to an old contract we wrote a few years ago or revise an old will without spending hours typing in manually. After scanning our document using OCR, we can easily edit it with any word processing tool.

Once a scanned document is made available on a common database, it becomes instantly accessible to multiple people. This is particularly useful for banks that can check a customer's credit history at any time.

This technique makes government records available so that the records of business owners or the attorney's grandfather's birth certificate can be found by simple text search.

Digitization obviously reduces the space required for the storage of paper archives, as computers take up very little physical space. In addition, these, once digitized, become useless and can be recycled.

Instead of keeping duplicate or triplicate documents, scanning can be done inexpensively and without limits. We can also keep a version of all changes. In addition, scanned documents are not likely to be damaged over time. It greatly simplifies document management.

Modern OCR is capable of handling a large number of languages, from Arabic to Indian to Chinese. This implies that a document, in one language, can be searched, scanned and automatically translated into any other language.

This work is greatly simplified with the Unicode standard and machine-based translation programs (for example, Google Translate).

Therefore, we can almost eliminate the need for professional translators.

One of the most important advantages of OCR is it contribution in the organizations of companies documents. Nowadays, companies often generate a very high volume of data and documents: legal contracts, packing list, government forms, user licenses, certificates, catalogs, etc.

Thanks to OCR and scanning, in addition to digital archiving, comparison between documents is possible and much simpler.

We can check for discrepancies and conflicting information in our systems. For example, checks can be verified to validate the correct amount and invoices can be compared to accounts receivable and payments received, etc.

Finally, by scanning your documents, we make them available for statistical analysis. This activity can quickly give us some improvement points for the processes for your business.

OCR is the first critical phase in transforming analog recordings into digital documents.

In this work, our objective has been to explore the field of text detection and recognition, to give an overview of what this problem is and to represent the different techniques and approaches proposed and their results.

In the first chapter we exposed the field of text detection and recognition and we talked about the functioning of these systems while showing the important techniques used in this field.

In the second chapter, the concept of deep learning, the most important architectures and algorithms as well as their usefulness and their importance in our work, were all reviewed.

In the last chapter, we explained the stages of our implementation work, the necessary tools and libraries as well as the results obtained.

Key words: *Text detection ; Text recognition ; Deep learning ; openCV ; OCR engins.*

Acronymes

OCR: Optical Character Recognition (Reconnaissance optique des caractères)

ML: Machine Learning (Apprentissage automatique)

DL: Deep Learning (Apprentissage profond)

AI: Artificial Intelligence (Intelligence artificielle)

CNN: Convolutional Neural Network (Réseau neuronal convolutif)

RNN: Recurrent Neural Network

LSTM: Long Short Term Memory

RLSA: Run Length Smoothing Algorithm

RLSO: Run Length Smoothing with Or

SIFT: Scale Invariant Feature Transform

SURF: Speeded Up Robust Features

K-NN: K Nearest Neighbors (k plus proches voisins)

SVM : support vector machine

ReLU : (Unité de Rectification Linéaire)

RCNN: Regions with CNN

R2CNN: Rotational CNN

SSD: Single Shot multibox Detector.

YOLO: You Only Look Once

CRNN: Convolutional Recurrent Neural Network

STN: Spatial Transformer Network

EAST: An Efficient and Accurate Scene Text Detector

Table des figures

Liste des tableaux

Table des matières

Chapitre 2 L'apprentissage profond

Introduction générale

La reconnaissance optique de caractères, en anglais *optical character recognition* (OCR), désigne les procédés d'extraction de texte à partir d'images de textes.

Un ordinateur réclame pour l'exécution de cette tâche un logiciel d'OCR. Celui-ci permet de récupérer le texte dans l'image et de le sauvegarder dans un fichier pouvant être exploité dans plusieurs applications comme le tri automatique des documents, l'aide aux non-voyants, l'automatisation des services publics, l'archivage numérique des documents physiques, l'amélioration des interfaces homme-machine.

La première machine d'OCR fut créée en 1929 et depuis cette date, la reconnaissance de caractères est un domaine actif de recherche. Au fil du temps, ce domaine a pris de l'ampleur et avec l'avènement de nouvelles avancées, il est apparu beaucoup plus intéressant. Il faudra encore de nombreuses décennies aux ordinateurs, s'ils y parviennent un jour, pour lire tous les documents avec la même précision que les êtres humains. Ce domaine exploite surtout le deep learning et les réseaux récurrents et à mémoire. Ce champ de l'intelligence artificielle est cependant un peu moins mature que celui de l'identifier des personnes, et objets.
La recherche continue d'avancer dans ce créneau et le besoin croissant d'automatisation des systèmes a affecté le développement de la détection et de la reconnaissance de texte à partir d'images dans une large mesure. La reconnaissance de texte a un large éventail d'applications, chacune avec des défis et des complications. La localisation et la séparation du texte, des informations non textuelles constituent le plus gros problème dans la procédure d'océration compte tenu de variétés des entrées que nous pouvons avoir, allant d'une simple image d'un livre dans un éclairage et des circonstances parfaites à une image d'une scène naturelle avec tant d'imperfections.

Dans ce travail, notre objectif a été celui d'explorer le domaine de la détection et de la reconnaissance de texte, de donner un aperçu de ce qu'est ce problème et de représenter les différentes techniques et approches proposées et leurs résultats.

Dans le premier chapitre nous avons exposé le domaine de la détection et de la reconnaissance de texte et nous avons parlé du fonctionnement de ces systèmes tout en montrant les techniques importantes utilisées dans ce domaine.

Dans le second chapitre, le concept d'apprentissage profond (deep learning), les architectures et algorithmes les plus importants ainsi que leur utilité et leur importance dans notre travail, ont tous été passés en revue.

Dans le dernier chapitre, nous avons expliqué les étapes de notre travail d'implémentation, les outils et les bibliothèques nécessaires ainsi que les résultats obtenus.

Chapitre 1: Détection et Reconnaissance de texte

1 Introduction

La reconnaissance optique de caractères (OCR) est la conversion d'images de texte dactylographié, manuscrit ou imprimé en texte codé par machine, que ce soit à partir d'un document numérisé, d'une photo d'un document, d'une photo de scène (par exemple du texte sur des panneaux d'affichage sur la route) ou à partir d'un texte de sous-titre superposé à une image (par exemple d'une émission de télévision).

La reconnaissance optique de caractères est largement utilisée comme forme de numérisation de texte à partir des documents papiers imprimées.

Qu'il s'agisse de passeports, de livres, de relevés bancaires ou de toute autre documentation appropriée, c'est une méthode courante de récupération de texte pour qu'il puisse être exploité, modifié, stocké électroniquement, ou affiché en ligne.
L'OCR est un domaine de recherche en reconnaissance de formes, en intelligence artificielle et en vision par ordinateur.

Les premières versions devaient être entraînées avec des images de chaque lettre et utilisées sur une police à la fois. Les systèmes actuels sont capables de produire un haut degré de précision de reconnaissance pour la plupart des polices et prennent en charge une variété d'entrées de formats de fichiers d'images ou de documents (PDF, JPG jpeg, bmp …. Etc.) et peuvent même fonctionner en temps réel [1].

1.1 Histoire

Chronologie	Récapitulation
1870–1931	Les premières idées de reconnaissance optique de caractères (OCR) sont conçues. L'optophone de Fournier d'Albe et la machine de lecture de Tauschek sont développés comme des dispositifs pour aider les aveugles à lire.
1931–1954	Les premiers outils OCR sont inventés et appliqués dans l'industrie, capables d'interpréter le code Morse et de lire le texte à haute voix. L'Intelligent Machines Research Corporation est la première entreprise créée à vendre de tels outils.
1954–1974	L'Optacon, le premier dispositif OCR portable, est développé. Des appareils similaires sont utilisés pour numériser les coupons et les adresses postales. Les polices de caractères spéciales sont conçues pour faciliter la numérisation.
1974–2000	Les scanners sont utilisés massivement pour lire les codes à barre et les passeports. Des sociétés telles que Caere Corporation, ABBYY et Kurzweil Computer ProductsInc. sont créées. Ce dernier développe le premier logiciel OCR omni-font, capable de lire n'importe quel document texte.
2000–2016	Le logiciel OCR est disponible gratuitement en ligne, via des produits comme Adobe Acrobat, WebOCR et Google Drive.

Tableau 1.1 : Chronologie de développement de technologie d'OCR [Web1].

1.2 Applications

Les outils d'OCR ont été développés en une gamme d'applications spécifiques au domaine, notamment la reconnaissance de reçu, de facture, de chèques, de documents légaux, etc.

D'autres cas d'utilisation peuvent être cités:

- **Saisie automatique de données** pour des documents d'entreprise, par exemple: formulaires papier, chèques, factures, relevés bancaires, reçus, etc.
- **Reconnaissance automatique des plaques d'immatriculation**.
- **Reconnaissance des passeports** de voyageurs dans un aéroport et l'extraction des informations importantes.
- **Extraction automatique d'informations clés** dans des documents.
- **Numérisation de gros documents imprimés**, par exemple des livres, d'archives.
- **Rendre disponible à la recherche** le texte des documents imprimés (partage en ligne).
- **Conversion de l'écriture manuscrite en temps réel** pour contrôler un ordinateur (pencomputing – via une tablette graphique ou un écran tactile par exemple).
- **Améliorer les systèmes anti-bots CAPTCHA,** bien que ceux-ci soient spécifiquement conçus pour empêcher l'OCR. Le but peut également être de tester la robustesse des systèmes anti-bots CAPTCHA.
- **Technologie d'assistance** pour les utilisateurs aveugles et malvoyants.

2 La détection et la reconnaissance de texte

Le challenge de l'OCR repose principalement sur la difficulté de localiser les régions de texte (ROI : regions of interest) dans une image et de reconnaître les caractères (feature extraction) qui sont écrits avec des différentes polices qui démultiplient les façons d'écrire chaque symbole. Ceci fait en sorte qu'avant même d'essayer d'identifier un caractère l'image en elle-même doit être prétraitée pour en assurer la facilité d'exécuter une telle tâche.

Les étapes de la détection et la reconnaissance de texte sont :

2.1 Acquisition d'images

Cela implique de numériser un document et de le stocker en tant qu'image sur laquelle les techniques d'OCR doivent être effectuées (scanner, capturer en photo …).

Pour un système d'OCR l'image est un tableau multidimensionnel [2]. (Tableau 2D si l'image est en niveaux de gris (ou) binaire, tableau 3D si l'image est colorée). Chaque cellule de la matrice est appelée pixel et peut stocker un entier de 8 bits, ce qui signifie que la plage de pixels est comprise entre 0 et 255.

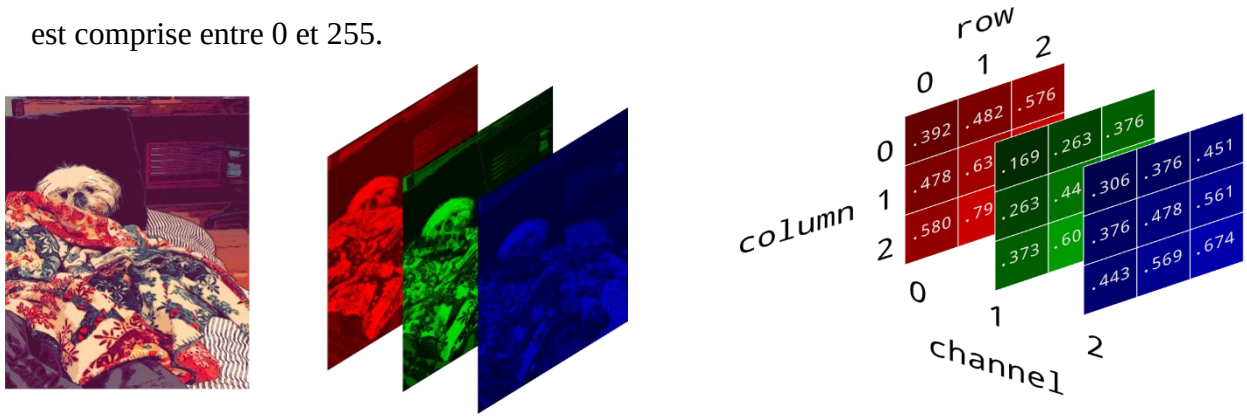


Figure 1.1 : Représentation d'une image rgb avec les chaines rouge, vert, Blue avec l'intensité de chaque pixel (de 0 à 1 équivalent de 0 à 255) dans chaque chaine [web2].

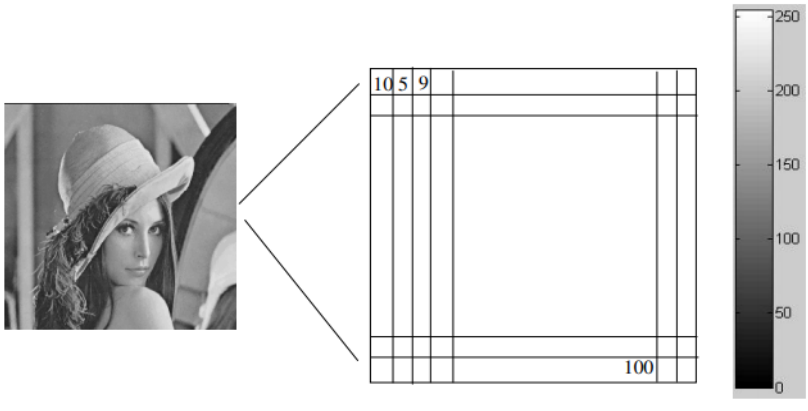


Figure 1.2: Représentation d'une image en niveau de gris (grayscale) avec une seule chaine [2].

2.2 Prétraitement

Nous ne pouvons pas donner une image en entrée, directement pour le système d'OCR. Un certain prétraitement de l'image doit être effectué afin qu'il soit assez facile pour notre système de reconnaître les informations dans l'image et de détecter le texte, les techniques de prétraitement comprennent :

-suppression du bruit.
-binarisation.
-réalignement.
-segmentation.
-normalisation.

2.2.1 La suppression du bruit (*noise removal*) :

Consiste à enlever les taches sur le document ou lisser ses bords [3], le bruit (petits points ou composants de premier plan) peut être facilement introduit dans une image lors de la numérisation pendant l'acquisition d'image en raison de la faible clarté de la caméra, de l'ombre sur l'image, etc.
On peut citer quelques méthodes de suppression de bruit :

- L'utilisation des filtres linéaires.
- Markov random field based methods.
- Anisotropic non linear diffusion.
- Wavelet based methods.

Figure 1.3 : Image avant (gauche) et après (droite) la suppression du bruit avec le filtre médian [4].

2.2.2 La binarisation :

Consiste à convertir une image en noir et blanc (appelée une « image binaire »). La tâche de binarisation est effectuée comme un moyen facile et précis de distinguer le texte de l'arrière-plan.

En termes simples, la binarisation signifie la conversion d'une image colorée en une image composée uniquement de pixels noirs et blancs (valeur de pixel noir = 0 et valeur de pixel blanc = 255). En tant que méthode de base, cela peut être fait en fixant un seuil et le pixel est considéré comme un pixel blanc si la valeur du pixel est supérieure au seuil, sinon comme un pixel noir.

```
if ( currentpixelvalue > threshold )
        currentpixelvalue=255  #Setting it as white pixel
else
        currentpixelvalue=0    #Setting it as black pixel
```

Mais cette approche (general thresholding) [Web3] ne nous donne pas toujours les résultats escomptés. Dans les cas où les conditions d'éclairage n'étaient pas uniformes dans l'image, cette méthode échoue terriblement à binariser correctement l'image.

En pratique, la conversion d'image colorée en image binaire se fait en créant une image intermédiaire en niveaux de gris.

Image en couleur -> Image en niveaux de gris -> Image binaire.

Ainsi, la partie cruciale de la binarisation consiste à déterminer le seuil. Cela peut être fait en utilisant diverses techniques comme.

- Adaptive Thresholding [Web4]

Cette méthode nous donne un seuil pour une petite partie de l'image en fonction des caractéristiques de sa localité et de ses voisins, c'est-à-dire qu'il n'y a pas de seuil fixe unique pour l'image entière mais chaque petite partie de l'image à un seuil.

- Local Maxima and Minima Method.

$$c(i,j) = \frac{I_{max} - I_{min}}{I_{max} - I_{min} + \varepsilon}$$ **Equation 1.1**

I_{max} Est la plus grande valeur de pixel dans l'image.

I_{min} Est la plus petite valeur de pixel dans l'image.

ε Valeur constante.

cEst le seuil.

- Méthode d'Otsu

la méthode d'Otsu est utilisée pour effectuer un seuillage automatique à partir de la forme de l'histogramme de l'image, L'algorithme suppose alors que l'image à binariser ne contient que deux classes de pixels, (c'est-à-dire le premier plan et l'arrière-plan) puis calcule le seuil optimal qui sépare ces deux classes afin que leur variance intra-classe soit minimale.

- Méthode de Niblack

L'idée de la méthode est de varier le seuil dans l'image en fonction des valeurs de la moyenne locale et de l'écart type local, Le seuil calculé pour le pixel (x, y) est :

$$T(x,y) = m(x,y) + k * \delta(x,y)$$ **Equation 1.2**

où $m(x,y)$ et $\delta(x,y)$ sont respectivement la moyenne et l'écart type calculés dans un voisinage local de (x,y).

Figure 1.4 : Résultat de binarisation (la moitié droite) avec la méthode d'otsu [web5].

2.2.3 **Réalignement (*de-skew*) :** Rendre le texte parfaitement horizontal.

L'image obtenue à l'étape précédente peut ne pas être correctement orientée, elle peut être alignée à n'importe quel angle. il peut avoir besoin d'une rotation de quelques degrés dans le sens horaire ou antihoraire pour s'assurer que les lignes de texte soient parfaitement horizontales ou verticales pour nous assurer que l'image transmise aux étapes suivantes est correctement orientée.

Il existe plusieurs techniques utilisées pour la correction de l'inclinaison [5].

- Projection profile method.
- Hough transformation method.
- Topline method.
- Scanline method.

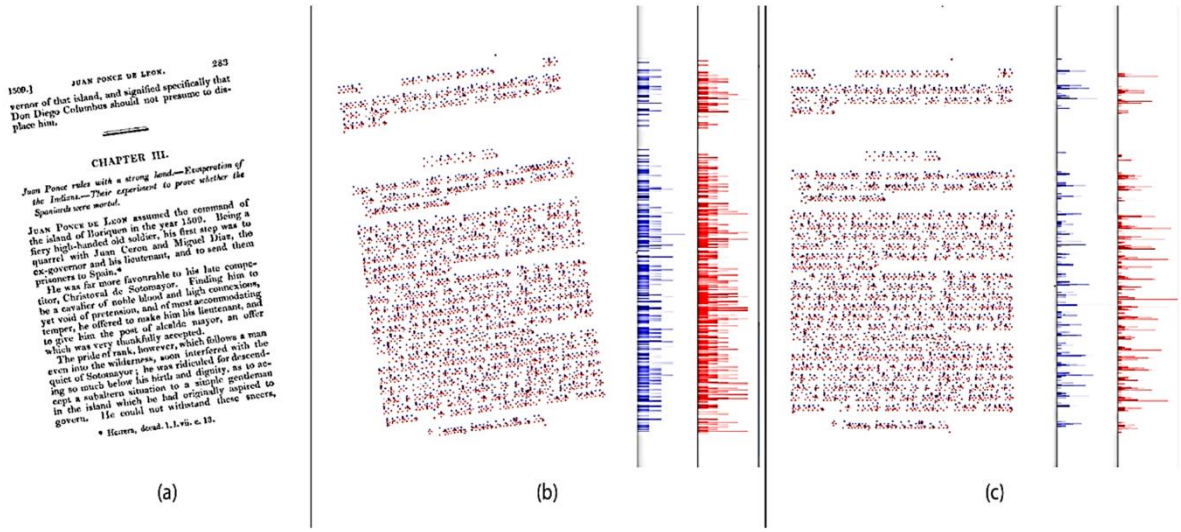


Figure 1.5: Réalignement de texte avec scanline method, (a) image initial mal orienté, (b) image mal orienté avec l'histogramme projeté horizontalement, (c) image avec orientation correct et son histogramme, on remarque que les pics sont devenu minces parce que l'espace entre les lignes est horizontal [web6].

2.2.4 Normalisation

Normaliser le ratio des dimensions de l'image (aspect ratio) et l'échelle (scale ratio).

Il s'agit d'une tâche de prétraitement facultative. L'exécution de cette tâche dépend du contexte dans lequel l'OCR est utilisé.

- Si nous utilisons le système OCR pour le texte imprimé, pas besoin d'effectuer cette tâche car le texte imprimé a toujours une largeur de trait uniforme.
- Si nous utilisons le système OCR pour le texte manuscrit, cette tâche doit être effectuée car les différents écrivains ont un style d'écriture différent et donc une largeur de trait différente. Donc, pour uniformiser la largeur des traits, nous devons effectuer l'amincissement et la squelettisation.

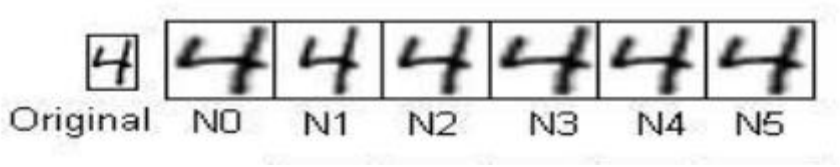


Figure 1.6: Normalisation d'image originale [6].

Avec :

symbole	Description	Aspect ratio
N0	normalisation linéaire avec rapport d'aspect fixe	$R2 = 1$
N1	normalisation linéaire avec rapport d'aspect préservé	$R2 = R1$
N2	normalisation linéaire avec rapport racine carrée	$R2 = \sqrt{R1}$
N3	normalisation linéaire avec rapport racine cubique	$R2 = \sqrt[3]{R1}$
N4	normalisation linéaire avec rapport d'aspect fixe*	$R2 = 0.9$
N5	normalisation linéaire avec racine carrée du sinus du l'aspect ratio	$R2 = sin(\frac{\pi}{2R1})$

Tableau 1.2 : les relations de quelques techniques de normalisation.

Ou aspect ratio = rapport d'aspect = le rapport hauteur / largeur.

Avec le rapport d'aspect de l'image originale est :

$$R1 = \frac{min\,(W1,H1)}{max\,(W1\,,H1)}$$ **Equation 1.3**

et le rapport d'aspect de l'image normalisée est :

$$R2 = \frac{min\,(W2,H2)}{max\,(W2\,,H2)}$$ **Equation 1.4**

Avec W1 la largeur et H1 la hauteur de l'image originale.

Et W2 la largeur et H2 la hauteur de l'image normalisée.

2.3 Détection de texte

La détection de texte consiste à localiser les régions de texte dans une image donnée appelées régions d'intérêt.

2.3.1 Segmentation (Isolation des caractères ou lignes):

Après la phase de prétraitement, une image «propre» est obtenue. L'étape suivante est la segmentation. La segmentation n'est rien d'autre que de diviser l'image entière en sous-parties pour les traiter davantage.

Pour l'OCR, trois types de segmentation peuvent être effectués:

- Segmentation au niveau ligne.
- Segmentation au niveau des mots.
- Segmentation au niveau du caractère.

Pour tous ces types de segmentation, nous allons utiliser la technique de projection d'histogramme, pour déterminer où nous devons segmenter.

Après la binarisation d'une image nous n'avons que des pixels en noir et blanc.

Projection de l'histogramme horizontal:

Dans cette méthode, nous comptons le nombre de pixels blancs (pixels de premier plan) le long de toutes les lignes de la matrice de l'image.

La projection horizontale du nombre de pixels blancs dans chaque ligne de la matrice de l'image résultant sous forme d'histogramme (tracé sur le graphique) ressemble à ceci.

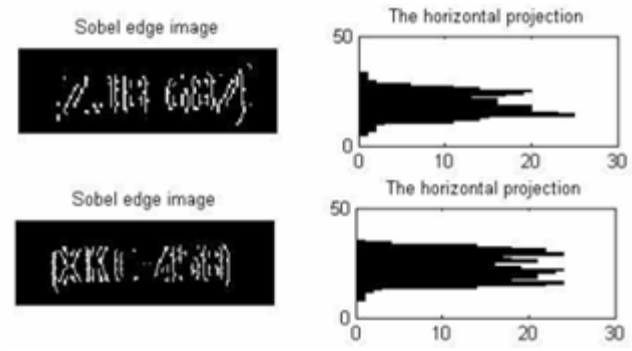


Figure 1.7: Projection horizontale d'histogramme sur une image binaire [7].

Dans cette figure :

Des pics plus élevés impliquent que le nombre de pixels blancs (pixels de premier plan) le long de cette ligne est élevé et que le nombre de pixels noirs (pixels d'arrière-plan) le long de cette ligne est faible.

Des pics inférieurs impliquent que le nombre de pixels blancs (pixels de premier plan) le long de cette ligne est faible et que le nombre de pixels noirs (pixels d'arrière-plan) le long de cette ligne est élevé.

Projection de l'histogramme vertical:

Dans cette méthode, nous comptons le nombre de pixels noirs (pixels de premier plan) le long de toutes les colonnes de l'image.

La projection verticale du tableau du nombre de pixels résultants sous forme d'histogramme (tracé sur le graphique) ressemble à ceci :

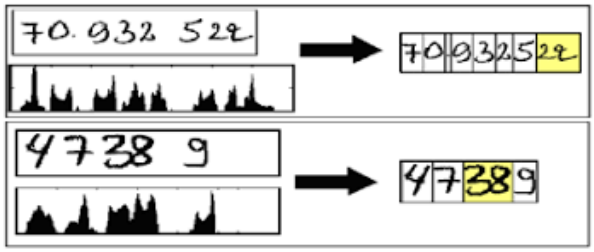


Figure 1.8: Projection verticale d'histogramme sure une image binaire

Dans cette figure :

Des pics plus élevés impliquent que le nombre de pixels noirs (pixels de premier plan) le long de cette colonne est élevé et le nombre de pixels blancs (pixels d'arrière-plan) le long de cette colonne sont faibles dans cette image.

Les pics inférieurs impliquent que le nombre de pixels noirs (pixels de premier plan) le long de cette colonne est faible et que le nombre de pixels blancs (pixels d'arrière-plan) le long de cette colonne est élevé dans cette image.

- Segmentation au niveau lignes

Dans ce niveau de segmentation, nous aurons une image contenant du texte écrit en lignes en entrée. L'objectif principal de la segmentation de niveau ligne est de déterminer les coordonnées des lignes dans l'image, ce qui peut diviser l'image en lignes.

Généralement, lorsque nous écrivons quelque chose sous forme de lignes, nous laissons suffisamment d'espace entre les lignes. Nous prenons cet espace entre les lignes comme avantage et procédons à la segmentation.

L'idée est que, si nous utilisons la projection de l'histogramme horizontal, les lignes qui contiennent le texte seront affichées comme un pic dans l'histogramme, nous sélectionnons toutes les lignes qui ont des pics inférieurs dans l'histogramme, comme les lignes de segmentation pour séparer les lignes.

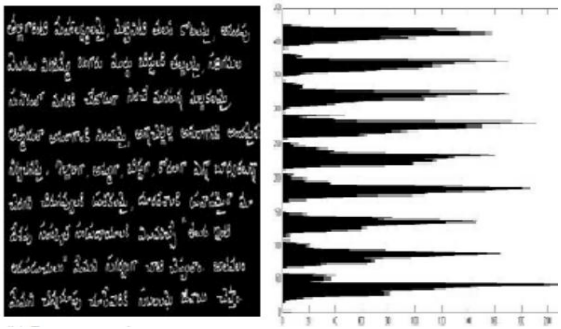


Figure 1.9 : la segmentation au niveau des lignes à l'aide de la projection horizontale d'histogramme [8].

- Segmentation au niveau des mots :

Une fois que nous avons segmenté l'image entière en lignes à l'étape précédente, nous nous retrouvons avec des lignes contenant plusieurs mots. Dans cette étape, l'objectif principal est de déterminer où nous devons segmenter les lignes pour séparer les mots.

Encore une fois, si nous nous souvenons des règles générales que nous suivons lors de l'écriture, nous laissons suffisamment d'espace entre les mots. Semblable à l'étape précédente, nous profitons de cet espace pour séparer les mots.

L'idée est la même que l'étape précédente, mais cette fois on utilise la projection de l'histogramme vertical les colonnes de l'image qui contiennent du texte affiche des plus grands pic dans l'histogramme toutes les colonnes qui ont des petits pics sont utilisées comme lignes de séparation des mots.

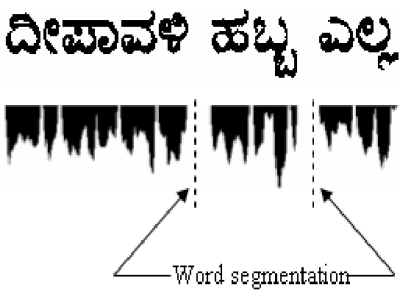


Figure 1.10: Segmentation au niveau des mots [Web7].

- Segmentation au niveau des caractères :

Cette segmentation au niveau du caractère est facultative, ce qui dépend du contexte dans lequel l'OCR est utilisé (OCR d'un texte imprimé ou manuscrit)

Si le système OCR est appliqué sur du texte imprimé, dans lequel les caractères dans le mot ne sont pas joints les uns aux autres, les caractères sont également segmentés à l'étape précédente elle-même (avec les mots) parce que le texte imprimé maintient un espacement uniforme considérable (mais petit) entre les caractères. Il n'est donc pas nécessaire d'effectuer une segmentation au niveau de caractères si non dans le cas du texte manuscrit une autre segmentation pour les caractères est nécessaire.

2.3.2 Algorithmes de segmentation

On cite quelques algorithmes :

a- RLSA (Run Length Smoothing Algorithm)

Le RLSA est un algorithme de segmentation qui a pour but de créer des zones permettant d'identifier les différentes composantes d'un document. Basé sur l'explication d'Olivier Augereau dans [9], le RLSA crée ces zones en reliant les pixels noirs (foreground) qui sont séparés par des pixels blancs (background). La quantité de pixels blancs minimale (n) pour créer la séparation est définie par rapport au détail de segmentation : plus n est petit, plus il y aura de zones segmentées (séparation lettre à lettre) et plus n est grand moins il y aura de zones segmentées (séparation paragraphe à paragraphe). L'algorithme s'applique de manière horizontale puis verticale (les espacements horizontaux et verticaux n'ayant, d'habitude, pas la même taille) puis les résultats sont combinés par ET logique [Web8]. Cependant, cet algorithme n'est applicable que sur des documents ayant une structure de type Manhattan.

b- RLSO (Run Length Smoothing with OR)

L'algorithme RLSO est un algorithme basé sur RLSA mais qui est destiné aux documents ayant une structure de type Non-Manhattan. Dans [10], les auteurs expliquent le fonctionnement de l'algorithme. RLSO reste assez similaire à RLSA mais utilise le OU logique (plutôt que le ET logique) ce qui permet de préserver les pixels noirs de l'image originelle et ne requiert donc pas une deuxième passe horizontale pour rétablir la structure d'origine. Une autre amélioration apportée sur cet algorithme est la définition automatique du seuillage, basée sur la distribution des espacements dans le document.

Figure 1.11 : Segmentation avec RLSA.

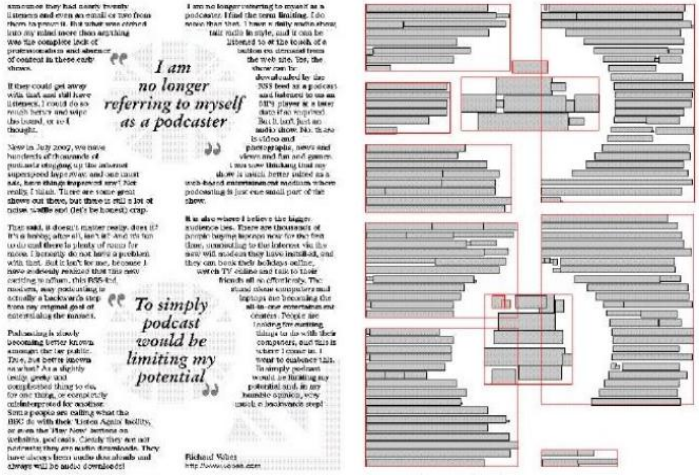


Figure 1.12 : Segmentation avec RLSO.

2.4 Reconnaissance de texte

Apres la détection des régions de texte dans une image on passe à la phase de reconnaissance de texte.

La phase de reconnaissance de texte se compose de deux grandes étapes :

L'extraction des caractéristiques de l'image (feature extraction) et la classification

2.4.1 Extraction des caractéristiques de l'image (Feature extraction)

L'extraction des caractéristiques est une étape fondamentale du processus de reconnaissance de texte qui commence à partir d'un ensemble initial de données mesurées (caractéristiques) pour faciliter les étapes ultérieures d'apprentissage et de généralisation.

Lorsque les données d'entrée sont trop volumineuses pour être traitées manuellement et sont susceptibles d'être redondantes, les données peuvent être transformées en un ensemble réduit de caractéristiques (appelé vecteur de caractéristiques), puis toutes les caractéristiques sont comparées au vecteur de caractéristique pour identifier le caractère.

Les caractéristiques d'un caractère peuvent être (dépend du police)

Suivant l'anatomie de typographie on peut citer quelque caractéristique :

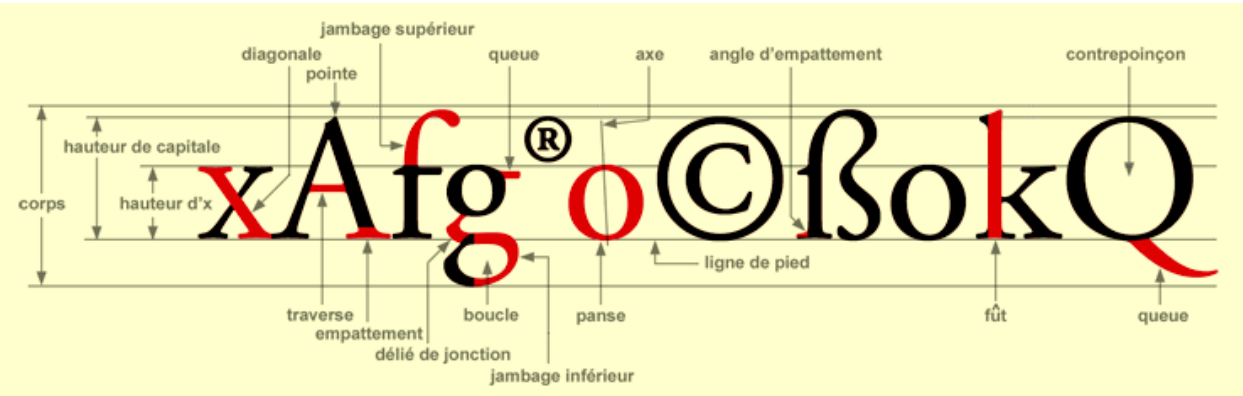


Figure 1.13 : Démonstration de quelques caractéristiques essentielles dans l'identification du caractère [Web9].

Quelque Algorithmes d'extraction des propriétés statistiques de l'image :

a- SIFT

SIFT ou Scale Invariant FeatureTransform est un algorithme utilisé pour détecter et identifier des fragments identiques entre deux images. Comme expliqué dans [11], l'algorithme est divisé en deux parties : détection et calcul des descripteurs puis mise en correspondance. SIFT commence ainsi par calculer des descripteurs (les descripteurs SIFT) qui vont permettre de caractériser le contenu de l'image de manière indépendante de l'échelle, de la résolution, de la rotation ainsi que de la netteté. La détection s'effectue sur des zones circulaires de rayon "facteur d'échelle" (x, y, σ), centrées autour d'un point d'intérêt de l'image. Cela mène à une construction d'un histogramme des orientations locales des contours puis l'histogramme est mis sous forme d'un vecteur à 128 dimensions. Ce descripteur est connu pour être robuste et est utilisé dans plusieurs domaines comme la détection de points d'intérêt, la mise en correspondance d'images, le suivi d'objets ou l'assemblage d'images.

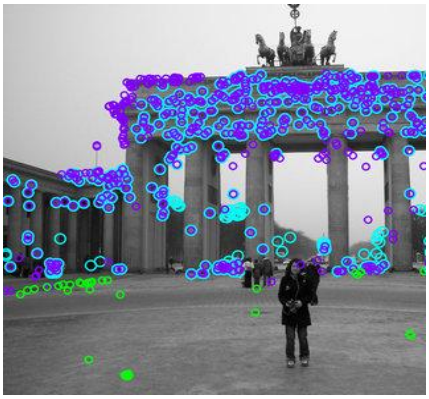


Figure 1.14: Points d'intérêt détectés par SIFT [12].

Suite au calcul des descripteurs, SIFT effectue une mise en correspondance avec une base d'images de référence utilisant la distance euclidienne. Parmi toutes les correspondances effectuées par l'algorithme, des sous-ensembles sont définis basés sur la qualité de la correspondance puis les meilleurs sont conservés. Un modèle probabiliste est enfin appliqué afin de confirmer cette correspondance.

Figure 1.15 : Mise en correspondance de deux images par SIFT [web10].

b- SURF

SURF ou Speeded Up Robust Features est un algorithme, en partie basé sur SIFT, mais plus rapide et robuste que ce dernier. Sa rapidité est due à l'utilisation d'images intégrales et le calcul des points d'intérêt effectué par approximation de matrice Hessienne. L'algorithme est expliqué en détail dans [13].

MÉTHODE	TEMPS	ÉCHELLE	ROTATION	FLOU	ILLUMINATION	AFFINE
SIFT	normal	meilleur	meilleur	meilleur	normal	bon
PCA-SIFT	bon	normal	bon	normal	bon	bon
SURF	meilleur	bon	normal	bon	meilleur	bon

Tableau 1.3 : comparaison entre la méthode SIFT et SURF [web11].

2.4.2 La classification :

La classification est le processus de prédiction de la classe des données détectées (les caractères dans notre cas) .

Quelque Algorithmes de classification :

a- K-PPV (K-NN)

k-NN (k-nearest neighbor) ou k-PPV (k plus proches voisins) est une méthode d'apprentissage supervisée. Cette méthode consiste à calculer la distance entre un point et les k points qui lui sont les plus proches. Pour calculer cette distance, on utilise la distance euclidienne :

$d(x_i, x_j) = \sqrt{\sum_{k=1}^{d}(x_{ik} - x_{jk})^2}$ **Equation 1.5**

Appliqué à la reconnaissance des documents, cette méthode va permettre de comparer les vecteurs de caractéristiques d'une image à classer, avec les vecteurs de caractéristiques des images de la base d'apprentissage. On choisit ensuite la classe majoritaire parmi ses k plus proches voisins. Pour déterminer k on peut également utiliser la technique de validation croisée.

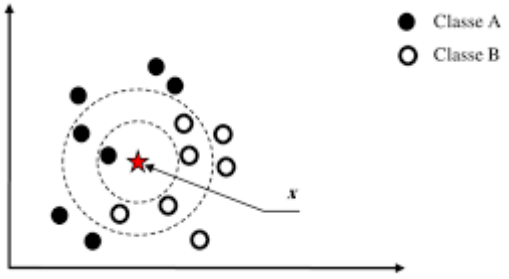


Figure 1.16: Exemple de classification par k-PPV [14].

b- SVM

SVM (séparateur à vaste marge ou support vector machine est, tout comme k-PPV, une méthode d'apprentissage supervisée. SVM est cependant plus complexe que cette dernière. SVM se base sur les notions d'hyperplan, marges et vecteurs support. Pour un ensemble de données, SVM va trouver un classificateur linéaire qui permet de séparer les différentes classes, l'hyperplan. On détermine ensuite les points les plus proches de cet hyperplan, qui constitueront les vecteurs de support. Ces vecteurs de support permettront de calculer la marge maximale de l'hyperplan. La **Figure 1.17**a été reprise de [15] et démontre l'application de SVM sur un ensemble de données.

Il peut exister une multitude de plans séparateurs (hyperplans) possibles pour un jeu de données. Pour résoudre ce problème, SVM cherche l'hyperplan optimal : celui qui maximise la marge définie par les vecteurs de support. Lors que les données ne sont pas linéairement séparables, SVM permet de changer leur espace de représentation (espace de ré-description). Plus cet espace est grand, plus il est probable de trouver un hyperplan séparateur convenable.

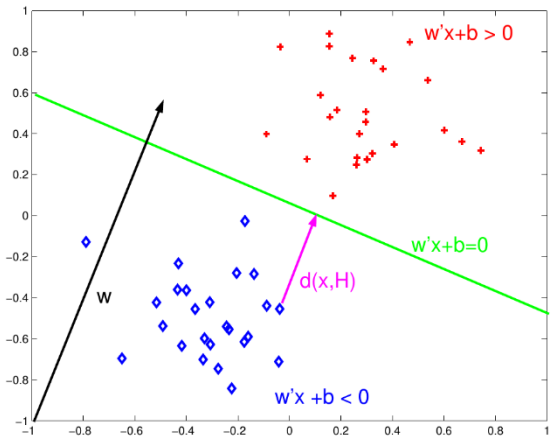


Figure 1.17 : Exemple de classification par SVM [web12].

2.5 Post traitement :

La précision de l'OCR peut être améliorée si son extrant est limité par un lexique (une liste de mots autorisés dans un document). Par exemple, un lexique pourrait comprendre tous les mots en anglais ou une liste de mots plus techniques spécifiques à un domaine en particulier.

Évidemment, cette méthode sera moins efficace si le document contient des mots qui ne sont pas dans le lexique. C'est souvent le cas avec les noms propres par exemple.

Heureusement, pour améliorer la précision il existe plusieurs librairies d'OCR gratuites sur Internet. La librairie Tesseract par exemple utilise son dictionnaire pour contrôler la segmentation des caractères et double passe (après la première passe de reconnaissance en exécute une deuxième itération de reconnaissance pour rectifier les faute de première itération) de classification pour améliorer les résultats.

L’extrant de l’algorithme peut être une seule chaîne ou un fichier de caractères. Les systèmes OCR plus avancés peuvent conserver la structure de page d’origine et créer un PDF contenant à la fois les pages d’image d’origine et le rendu textuel décodé.

- Les erreurs

Afin de corriger certaines erreurs, l’analyse des proches voisins peut utiliser des fréquences d’occurrence en notant que certains mots ont été vus ensemble. Par exemple de ces deux expressions similaires, « Washington, D.C. » est plus répandu dans la langue anglaise que « Washington DOC ». .Donc, ce type d'erreurs d'orthographe peut être corrigé en utilisant des modèles de langage, des modèles Word2Vec (comme CBOW et skip gram), etc.

- La grammaire

La grammaire peut également aider à déterminer la langue numérisée, par exemple, un mot est susceptible d’être un verbe ou un nom, fournit une plus grande précision.

Dans le post-traitement OCR, l’algorithme de distance de Levenshtein est souvent utilisé pour maximiser davantage les résultats de l’OCR.

3 Limitations d’un système OCR

L’identification et la reconnaissance du texte reste un domaine où l'humain est toujours supérieur à l'ordinateur: un système OCR a besoin d'une forme reconnaissable pour fonctionner **[Web13]**, il ne peut pas remplir les blancs et faire des suppositions comme l'humain le fait sans effort.

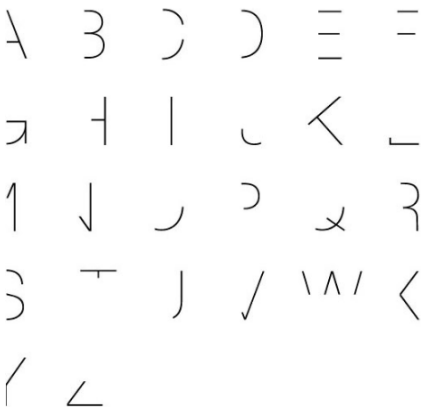


Figure 1.18 : L’alphabet - des formes que l'on devine facilement contrairement à l’ordinateur.

Il y a des polices avec lesquelles il est difficile de segmenter.

Figure 1.19 : Police swash difficile à segmenter.

4 Conclusion :

Dans ce chapitre, nous avons expliqué le fonctionnement de l'OCR, ses domaines d'applications et chaque étape importante de la détection et de la reconnaissance avec les techniques et algorithmes les plus utilisés ainsi que la limitation de ces systèmes.

Chapitre 2: l'apprentissage profond.

1 Introduction

L'apprentissage profond est une branche du Machine Learning, introduite pour amener l'intelligence artificielle à son objectif principal celui d'imiter l'intelligence humaine. Les briques élémentaires d'apprentissage profond sont les réseaux de neurones, qui sont combinés pour former les réseaux de neurones profonds.

Ces techniques ont permis des progrès significatifs dans les domaines du traitement du son et de l'image, notamment la reconnaissance faciale, la reconnaissance vocale, la vision par ordinateur, le traitement automatique du langage, la classification des textes, les applications potentielles sont très nombreuses.

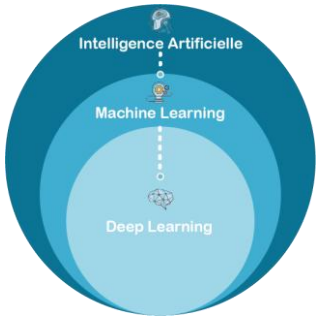


Figure 2.1: La relation entre l'intelligence artificielle, le ML et le deep learning (apprentissage profond) [Web14].

L'apprentissage profond est basé sur l'idée des réseaux de neurones artificielles Il existe plusieurs types d'architectures qui sont basés sur une cascade profonde de couches. Ils ont besoin d'algorithmes d'optimisation stochastique intelligents et d'initialisation, ainsi que d'un choix intelligent de la structure. Ils conduisent à des résultats très impressionnants, bien que très peu de fondements théoriques soient disponibles jusqu'à présent.

Essentiellement, un neurone est un nœud avec plusieurs entrées et une sortie, et de nombreux neurones interconnectés formant un réseau neuronal. Pour que les réseaux de neurones accomplissent leurs tâches, ils doivent passer par une «phase d'apprentissage», ce qui signifie qu'ils doivent apprendre à corréler les signaux entrants et sortants. Une fois cela fait, ils commencent à fonctionner, c'est-à-dire à recevoir des données d'entrée et à générer des signaux de sortie sur la base des informations accumulées.

1.1 Neurone biologique

Les neurones ont un corps cellulaire, un axone, des dendrites et un terminal axonal. Le corps cellulaire contient le noyau. L'axone s'étend du corps cellulaire et donne souvent naissance à de nombreuses petites branches avant de se terminer aux terminaisons nerveuses.

Les dendrites s'étendent du corps des cellules neuronales et reçoivent des messages d'autres neurones. Les synapses sont les points de contact où un neurone communique avec un autre. Les dendrites sont couvertes de synapses formées par les extrémités des axones d'autres neurones.

Les neurones sont connectés les uns aux autres. Ils ne se touchent pas et forment à la place de minuscules lacunes appelées synapses. Ces lacunes peuvent être des synapses chimiques ou des synapses électriques et elles aident à faire passer le signal d'un neurone au suivant.

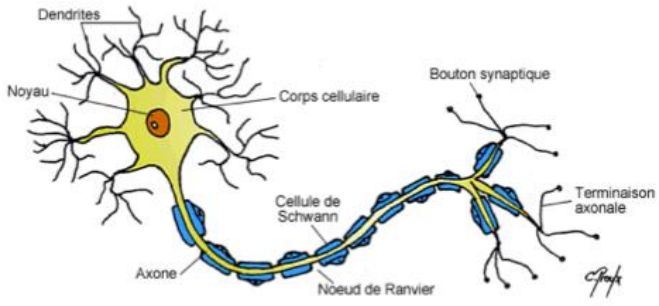


Figure 2.2: Représentation d'un neurone biologique [Web15].

1.2 L’implémentation du neurone biologique (perceptron)

Un perceptron est un neurone formel [16]. Un classifieur linéaire. Le perceptron ne trouve son intérêt que lorsqu’on veut effectuer une classification binaire.

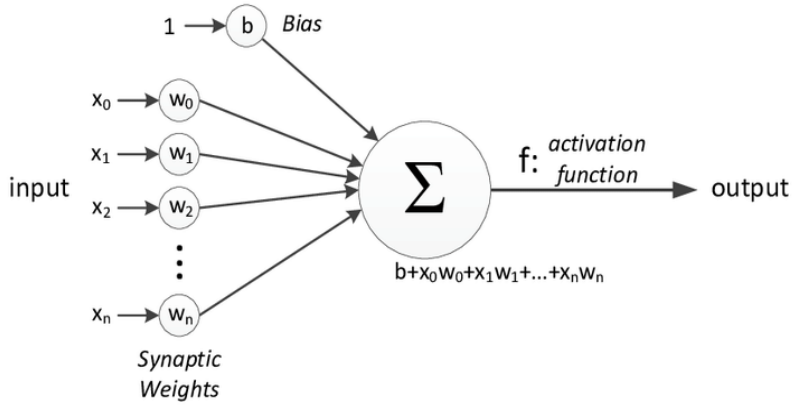


Figure 2.3: Représentation d’un perceptron.

Les entrées (x0, x1, x2, x3…xn) sont des variables indépendantes tandis que la sortie est une variable dépendante.

Une fois les entrées choisies, un poids est attribué à chaque entrée ; en changeant les poids, le perceptron apprend quel signal d’entrée est important.

La fonction d'activation est appliquée à la somme des entrées et des poids. En fonction de la fonction d'activation, le perceptron apprend quelle sortie il doit générer.

Puis la sortie est générée et la fonction de coût est calculée pour la sortie réelle correcte et la sortie calculée (sortie trouvé après prédictions). Et sur la base des corrections de cette fonction de coût, tous les poids sont réaffectés et le processus entier se répète jusqu'à ce que nous obtenions de meilleurs prédictions et tout ce processus est connu sous le nom « phase d’apprentissage d’un modèle ».

1.3 L’implémentation dans un réseau de neurone

Alors que la plupart des situations nécessitent plus qu'une classification binaire ou plus de traitement de données pour produire une sortie significative, c'est là que nous devons introduire des réseaux de neurones qui contiennent plusieurs couches en fonction de l'application.

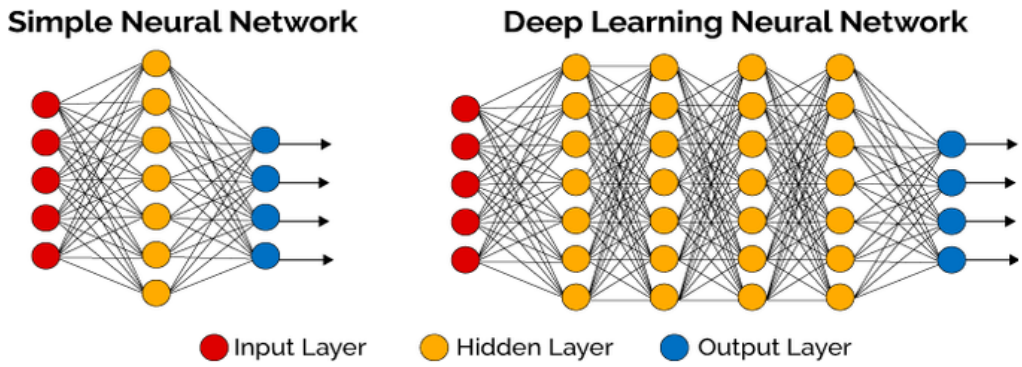


Figure 2.4: Réseaux de neurones simples avec plusieurs couches, à gauche réseaux de neurones avec une seule couche cachée, à droite un réseau avec 4 couches cachées [Web16].

- Les neurones d'un réseau neuronal sont disposés en couches où la première et la dernière couche sont appelées couches d'entrée et de sortie.
- Les couches d'entrée ont autant de neurones que le nombre d'attributs dans l'ensemble de données.
- La couche de sortie a autant de neurones que le nombre de classes de la variable cible en cas de problème de classification.
- La couche de sortie possède un neurone en cas de problème de régression.
- Les neurones d'une même couche n'interagissent pas entre eux.
- Toutes les entrées entrent dans le réseau par la couche d'entrée et toutes les sorties quittent le réseau par la couche de sortie.
- Les neurones des couches consécutives sont densément connectés, c'est-à-dire que tous les neurones de la couche n sont connectés à tous les neurones de la couche$n + 1$.
- Chaque interconnexion dans un réseau neuronal a un poids qui lui est associé, et chaque neurone à un biais qui lui est associé.
- Tous les neurones d'une couche particulière utilisent la même fonction d'activation.

Les différents types de réseaux de neurones sont schématisés et donnés en Annexe A.

1.4 Pourquoi le deep learning ?

Les algorithmes d'apprentissage profond sont utilisés pour les problèmes d'IA tels que la reconnaissance vocale et la reconnaissance d'objets que le Machine Learning n'a pas réussi à résoudre.

Mais ce n'est que lorsque de plus grandes quantités de données devenues disponibles et la montée en puissance de l'IoT et d'objets connectés ainsi que le grand avancement dans le hardware (cartes graphiques nvidia récentes), que le vrai potentiel de l'apprentissage profond a été révélé.

Les algorithmes d'apprentissage profond s'adaptent mieux que les algorithmes de ML, plus la quantité de données est grande, meilleures sont les performances des algorithmes d'apprentissage profond, contrairement aux algorithmes ML qui ont une limite sur la quantité de données qu'ils peuvent recevoir, les algorithmes d'apprentissage profond n'ont théoriquement pas une telle limite et ont même dépassé les performances humaines dans certains domaines.

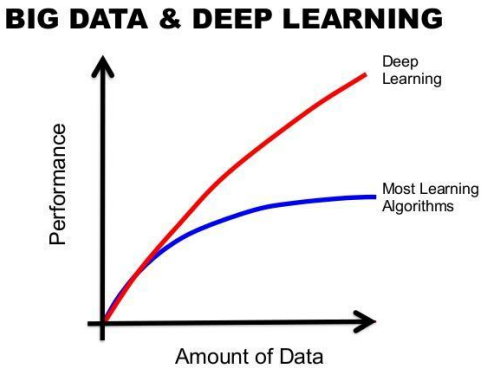


Figure 2.5: La différence de performance entre l'apprentissage profond et la plupart des algorithmes de ML en fonction de la quantité de données [Web17].

Une autre différence entre ML et l'apprentissage profond est qu'en ML, la procédure d'extraction des caractéristiques se fait manuellement, cela prend non seulement beaucoup de temps mais nécessite également un spécialiste dans ce domaine.

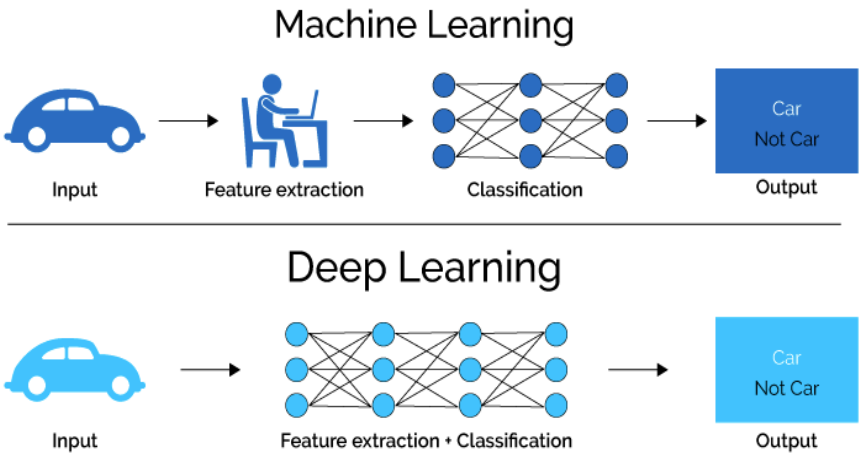


Figure 2.6: ProcedureML vs deeplearning [Web18].

2 Les différents types de modèles

Il existe un grand nombre d'architectures d'apprentissage profond, la plupart d'entre eux sont dérivés ou une variation des architectures de base, il est impossible de comparer deux architectures différentes à moins qu'elles ne soient destinées à la même application et même dans ce cas-là, les seuls critères de comparaison seront les performances, l'apprentissage profond est un domaine en pleine croissance et de nouvelles architectures, variantes ou algorithmes apparaissent chaque jour.

2.1 Les réseaux de neurones convolutifs CNN

Un réseau neuronal convolutionnel se compose d'une couche d'entrée et d'une sortie, ainsi que de plusieurs couches cachées. Les couches cachées d'un CNN consistent généralement en une série de couches convolutives qui se convoluent avec une multiplication ou un autre produit scalaire. La fonction d'activation est généralement une couche RELU et est ensuite suivie de convolutions supplémentaires telles que des couches de mise en commun, des couches entièrement connectées et des couches de normalisation, appelées couches cachées car leurs entrées et sorties sont masquées par la fonction d'activation et la convolution finale.

Les réseaux de neurones convolutifs sont une catégorie de réseaux de neurones qui se sont révélés très efficaces dans le domaine de classification d'images.

2.1.1 Inspiration

Dans l'apprentissage profond, un réseau convolutif est un type de réseau de neurones feed-forward, Le modèle de connectivité entre les neurones d'un CNN s'inspire de l'organisation du cortex visuel animal [17].

Les neurones corticaux individuels répondent aux stimuli dans une région restreinte de l'espace connu sous le nom de champ réceptif. Les champs réceptifs des différents neurones se chevauchent partiellement de sorte qu'ils couvrent le champ visuel.

La réponse d'un neurone individuel aux stimuli dans son champ réceptif peut être approchée mathématiquement par une opération de convolution.

En mathématiques, la convolution est une opération mathématique sur deux fonctions (f et g) qui produit une troisième fonction exprimant comment la forme de l'une est modifiée par l'autre.

Les réseaux de neurones convolutifs CNN se démarquent des réseaux de neurones traditionnels en s'appuyant sur trois idées importantes qui peuvent aider à améliorer la performance :

sparse interactions, parameter sharing et equivariant representations.

- **Sparse interactions** : contrairement au réseau neuronal entièrement connecté, pour la couche Convolution, chaque sortie n'est connectée qu'à des entrées limitées comme ci-dessus. Pour une couche cachée qui prend les neurones en entrée et les neurones en sortie, une couche cachée entièrement connectée a une matrice de poids de taille pour calculer chaque sortie. Lorsque m est très grand, le poids peut être une énorme matrice. Avec **sparse interactions**, seules k entrées sont connectées à chaque sortie, ce qui entraîne une diminution de l'échelle de calcul de O (m * n) à O (k * n), et une diminution de l'utilisation de la mémoire.

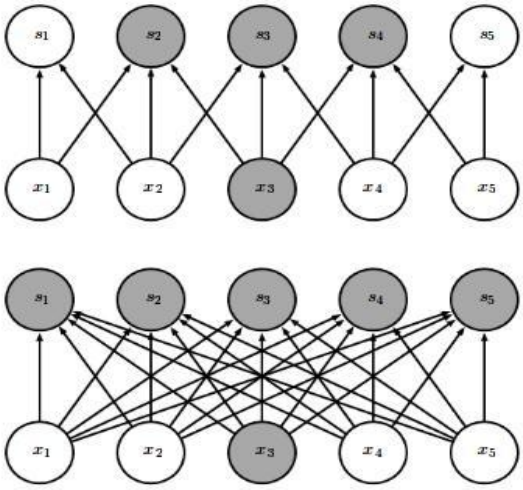


Figure 2.7:Sparse interactions : on met en évidence l'unité d'entrée, x_3, et les unités de sortie qui sont affectées par cette unité. (En haut) Lorsque s'est formé par convolution avec un noyau de largeur 3, seules trois sorties sont affectées par x. (en bas) Lorsque s'est formé par une multiplication matricielle, toutes les sorties sont atteintes par x_3, [Web19].

- **Parameter sharing** : (Le partage des paramètres) a plus de perspicacité lorsqu'il est considéré avec **sparse interactions**. Parce que **sparse interactions** crée une segmentation entre les données. Par exemple, x1, x5 est indépendant dans la figure ci-dessus à cause de **sparse interactions**. Cependant, avec le **parameter sharing**, la même matrice de poids est utilisée dans toutes les positions, ce qui conduit à une connectivité cachée. De plus, il peut réduire davantage le stockage en mémoire de la matrice de poids de $k \times n$à k. Surtout lorsqu'il s'agit d'une image, de $m \times n$ à k peut être une énorme amélioration de l'utilisation de la mémoire.

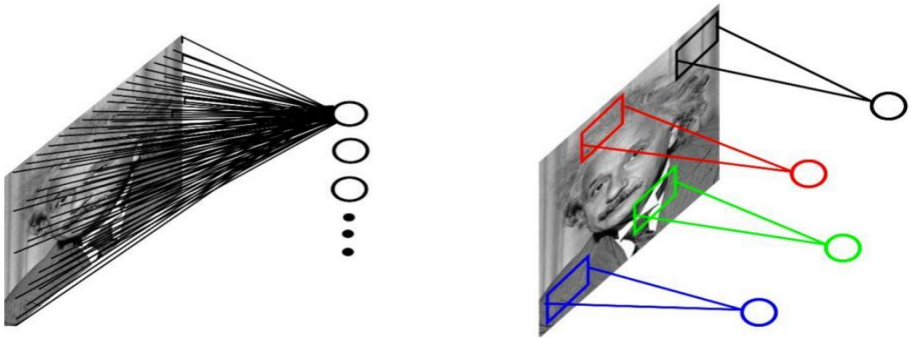


Figure 2.8: Exemple image 1000x1000. (À gauche) Le non parameter sharing nous oblige à concevoir une couche cachée de 10^6 neurones, chaque neurone est connecté à 10^6 pixels,(À droite) Avec un noyau 10x10 et une couche cachée de 10^6 neurones, le nombredeparamètresestde10^8[Web20].

- **Equivariant representation :** Dans le cas de la convolution, la particularité du parameter sharing fait que la couche possède une propriété appelée equivariant representation. Pour dire qu'une fonction est équivariante signifie que si l'entrée change, la sortie change de la même manière. Spécifiquement, une fonction $f(x)$ est équivariante à la fonction g si$f(g(x)) = g(f(x))$. La représentation équivariante est le résultat du partage de paramètres. Et parce que la même matrice de poids est utilisée à une position différente sur l'entrée. La sortie est donc invariante au mouvement parallèle, Dans le cas du traitement d'image Cette fonctionnalité peut être très utile lorsque nous nous soucions uniquement de la présence d'une caractéristique et non de sa position. Mais d'un autre côté, cela peut être un gros défaut de CNN car il n'est pas bon pour détecter la position (Et c'est pour ça qu'on doit segmenter le texte à l'entrée).

2.1.2 Architecture :

Une architecture CNN se compose de trois types de couches différentes. D'abord une couche convolutive une couche d'activation non linéaire telle que Rectified Linear Unit (ReLu), une couche pooling et enfin une couche fully connected.

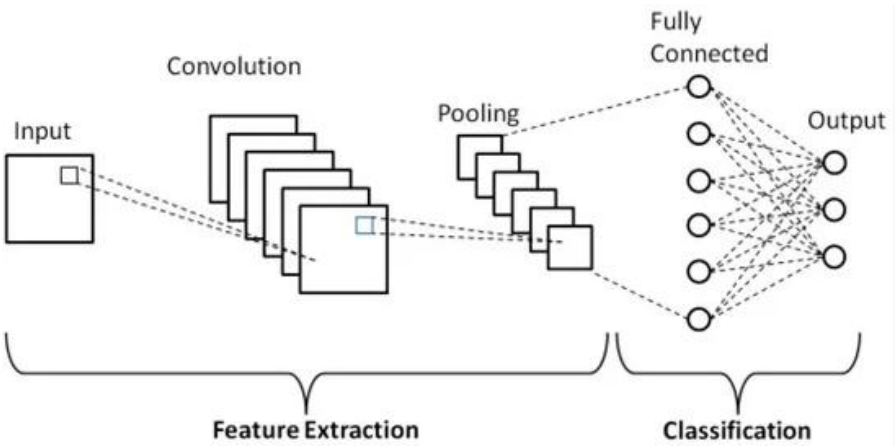


Figure 2.9: Architecture d'un réseau de neurones convolutionel[Web21].

a- Couche convolutive

Une couche de convolution transforme l'image d'entrée afin d'extraire des caractéristiques (featuremap). Dans cette transformation, l'image est convoluée avec un noyau (ou filtre) en anglais kernel.

Un noyau est une petite matrice dont la hauteur et la largeur sont inférieures à l'image à convoluer. Il est également connu sous le nom de matrice de convolution ou masque de convolution.

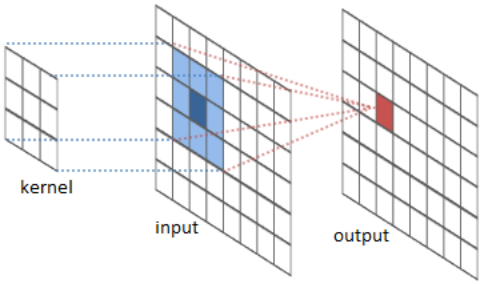


Figure 2.10: Entrée convoluée avec un noyau [Web22].

La convolution discrète sur un seul pixel d'une image implique de prendre ce pixel (illustré en bleu foncé ci-dessus) et de calculer la somme pondérée de ses voisins qui se trouvent dans une fenêtre spécifique pour produire le pixel de sortie (affiché en rouge foncé ci-dessus).). Les poids et la fenêtre sont décrits par le noyau de convolution. Ce processus est répété pour tous les pixels pour produire la sortie finale de la convolution.

Considérons une matrice 5×5 convoluée avec un noyau 3×3 comme indiqué ci-dessous.

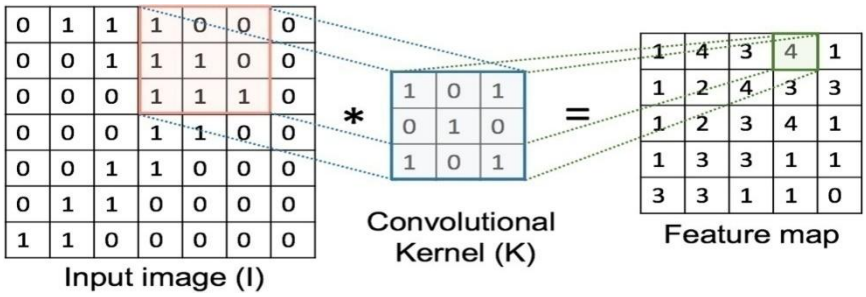


Figure 2.11: Concept de la somme pondérée dans la convolution d'une matrice d'entrée avec un noyau [Web23].

La somme pondérée pour cet élément est:

$(1 \times 1) + (0 \times 0) + (0 \times 1) + (1 \times 0) + (1 \times 1) + (0 \times 0) + (1 \times 1) + (1 \times 0) + (1 \times 1) = 4.$

La valeur de l'élément dans la matrice de sortie (feature map) est donc 4.

Une fonction d'activation est le dernier composant de la couche convolutionnelle utilisé pour augmenter la non-linéarité en sortie.

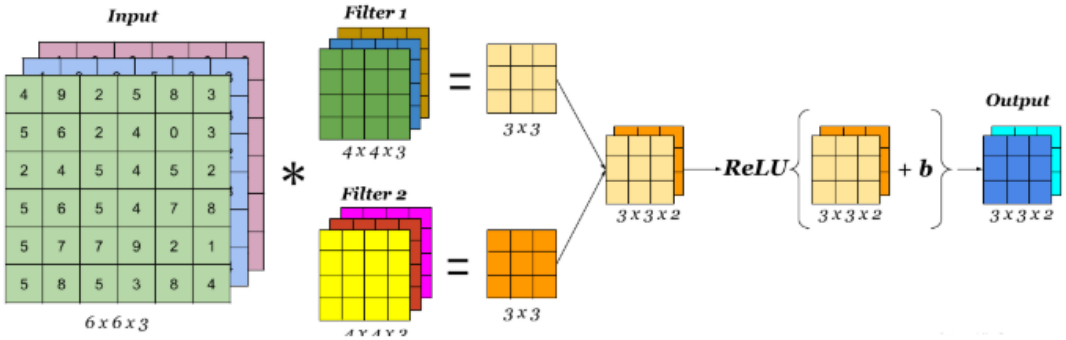


Figure 2.12: Une couche de convolution simple. une image d'entrée 6X6X3 est convoluée avec deux noyaux de taille 4X4X3 pour obtenir une matrice convoluée de taille 3X3X2, à laquelle la fonction d'activation est appliquée pour obtenir la sortie, également appelée featuremap [Web24].

b- Couche de pooling

Le **pooling** est le processus de sous-échantillonnage et de réduction de la taille de la matrice. Un filtre est passé sur les résultats de la couche précédente et sélectionne un nombre dans chaque groupe de valeurs (le maximum, c'est ce qu'on appelle le **max pooling**, et le moyen pour le **average pooling**). Cela permet au réseau de s'entraîner beaucoup plus rapidement, en se concentrant sur les informations les plus importantes dans chaque caractéristique de l'image et de contrôler également le sur-apprentissage.

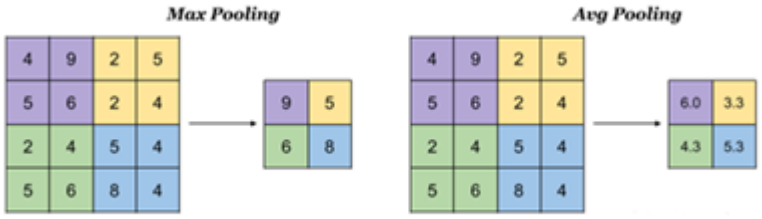


Figure 2.13: Max pooling (àgauche) on prend le maximum de chaque 4 cellules, average pooling (àdroite) on prend le moyen de chaque 4 cellules [Web25].

c- Couche fully connected

Après avoir extrait les caractéristiques des entrées, on attache à la fin du réseau une couche fully connected, la couche fully connected prend comme entrée les caractéristiques extraites et produit un vecteur de N dimensions où N est le nombre de classe et où chaque élément est la probabilité d'appartenance à une classe. Chaque probabilité est calculée à l'aide de la fonction.

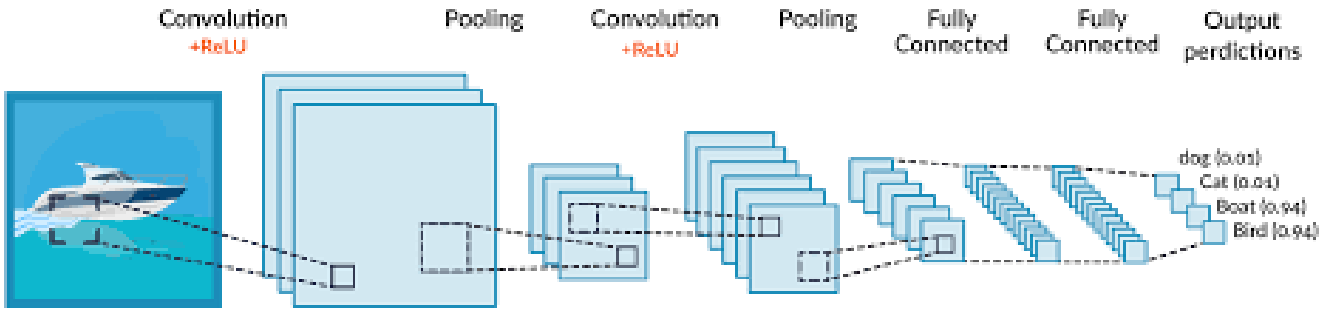


Figure 2.14: Un réseau de neurones convolutifs qui reçoit une image 2D comme entrée et qui est composé d'une couche convolutive, une fonction d'activation non linéaire ReLU , une couche pooling et enfin une couche fully connected avec 4 classes pour la classification [Web26].

2.1.3 L’apprentissage d’un réseau de neurones convolutif CNN

Le processus global de formation du réseau de convolution peut être résumé comme suit:

Étape 1: nous initialisons tous les filtres et paramètres / poids avec des valeurs aléatoires

Étape 2: Le réseau prend une image d’entrainement depuis le data set fourni en entrée, passe par l'étape de propagation directe (feedforward) (convolution, ReLU et pooling et puis la couche fully connected) et trouve les probabilités en sortie pour chaque classe.

Disons que les probabilités en sortie pour une image qui contient un texte sont [0,2, 0.7] avec les classes [ne contient pas du texte, contient du texte] respectivement.

Étant donné que les poids sont attribués au hasard pour le premier exemple d'apprentissage, les probabilités de sortie sont également aléatoires.

Étape 3: Calculez l'erreur totale au niveau de la couche de sortie (sommation sur les 2 classes)

Erreur totale = $\sum$ ½ (probabilité cible - probabilité de sortie) ^{2}.

Étape 4: utilisez la rétro propagation pour calculer les gradients de l'erreur par rapport à tous les poids du réseau et utilisez la descente de gradient pour mettre à jour toutes les valeurs / poids de filtre et les valeurs de paramètre pour minimiser l'erreur de sortie.

Les poids sont ajustés proportionnellement à leur contribution à l'erreur totale.

Lorsque la même image est à nouveau entrée, les probabilités de sortie peuvent désormais être [0,1, 0,9], ce qui est plus proche du vecteur cible [0, 0, 1, 0].

Cela signifie que le réseau a appris à classer correctement cette image particulière en ajustant ses poids / filtres de sorte que l'erreur de sortie soit réduite.

Des paramètres tels que le nombre de filtres, la taille des filtres, l'architecture du réseau, etc. ont tous été fixés avant l'étape 1 et ne changent pas pendant le processus d’entrainement - seules les valeurs de la matrice de filtre et les poids de connexion sont mis à jour.

Étape 5: Répétez les étapes 2 à 4 avec toutes les images du data set d'entraînement.

Apres ces 5 étapes la CNN est entrainé. - cela signifie essentiellement que tous les poids et paramètres du CNN ont maintenant été optimisés pour classer correctement les images de data set d'entraînement.

2.1.4 Applications

Les réseaux de neurones convolutifs ont excellé dans le domaine d'extraction de caractéristiques et la reconnaissance de formes, les CNN’s peut identifier et extraire des caractéristiques même dans des images déformées de faible qualité, c'est pour cela ce type de model est largement utilisé pour des applications telles que :

- Identification des formes [18].
- détection facial [19].
- détection de texte [20].

Une liste de quelques réseaux convolutifs célèbres dans Annexe B.

2.2 Les Réseaux de Neurones Récurrents (RNN)

2.2.1 Inspiration

Les humains ne commencent pas leurs pensées à partir de zéro à chaque fois, par exemple lorsque vous lisez un livre, vous comprenez un certain mot par son contexte dans la phrase, les réseaux CNN habituels échouent à cela car ils ne peuvent gérer qu'une seule entrée indépendante à la fois.

Pourquoi avons-nous des entrées et des sorties dépendantes?

Prenons un exemple où vous voulez prédire le mot suivant dans une phrase:

"Mohamed vit en algérie, il parle couramment ....."

Ce qui fera une bonne prédiction, c'est si vous savez mieux que "Il" est lié à "mohamed" et que le pays dans lequel il vit est "l'Algérie". Dans ce contexte, si le mot à côté de "couramment" commence avec "Ar" et même si le reste "abe" est impossible à identifier a cause d'une image de mauvaise qualité ou des problèmes de détection antérieurs, le mot sera facilement identifié comme "arabe" à l'aide du contexte précédent.

Et c'est là que les réseaux de neurones récurrents sont introduits.

2.2.2 Le Réseau de Neurones Récurrent (RNN)

Le concept des réseaux de neurones récurrents(RNN) est d'utiliser des informations séquentielles [21], ces réseaux de neurones récurrents sont une classe de réseaux de neurones qui permettent aux prédictions antérieures d'être utilisées comme entrées, par le biais d'états cachés.

Voici à quoi ressemble un RNN typique :

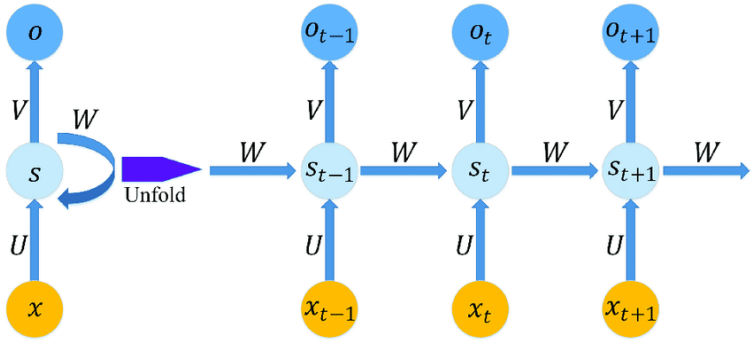


Figure 2.15: Schéma d'un réseau de neurones récurrents à une unité reliant l'entrée et la sortie du réseau. A droite la version « dépliée » de la structure.

Le schéma ci-dessus montre un RNN déroulé. En déroulant, nous signifions simplement qu'on montre le réseau pour la séquence complète. Par exemple, si la séquence qui nous intéresse est une phrase de 5 mots, le réseau serait déroulé en un réseau de neurones de 5 couches, une couche pour chaque mot. Les formules qui régissent les calculs dans un RNN sont les suivantes :

- x_t est l'entrée au moment t.

- *U, V, W* sont les paramètres que le réseau va apprendre des données de l'apprentissage.

- s_t est l'état caché au moment t. C'est la « mémoire » du réseau. s_t est calculé en fonction de l'état caché précédent et de l'entrée à l'étape actuelle :

$$s_t = f(U_{xt} + W_{s_{t-1}}) \quad \textbf{Equation 2.1}$$

Où f est une fonction non linéaire telle que : ReLu ou Hyperbolic tangent ($tanh$).

- o_t est la sortie au moment t. Par exemple, si on veut prédire le prochain mot dans une phrase, ce serait un vecteur de probabilités dans un vocabulaire.

$o_t = softmax(V_{s_t})$ **Equation 2.2**

Quelques remarques importantes :

- On peut considérer l'état caché s_t comme la mémoire du réseau. s_t capture des informations sur ce qui s'est passé dans toutes les étapes précédentes. La sortie o_t est calculée uniquement en fonction de la mémoire au moment t comme mentionné brièvement ci-dessus, c'est un peu plus compliqué dans la pratique car s_t ne peut généralement pas capturer des informations depuis trop longtemps.
- Contrairement à un réseau de neurones traditionnel, qui utilise différents paramètres à chaque couche, un RNN partage les mêmes paramètres (**parameter sharing**) (ici : *U, V, W*) à travers toutes les étapes. Cela reflète le fait que nous effectuons la même tâche à chaque étape, juste avec des entrées différentes. Ce qui réduit le nombre total de paramètres que le réseau devra apprendre.

2.2.3 L'Apprentissage d'un RNN

Pour l'apprentissage d'un RNN, on utilise une version légèrement modifiée de la retro propagation appelé **back propagation through time (BPTT).** Étant donné que les paramètres sont partagés à tout moment dans le réseau, le gradient à chaque sortie dépend non seulement des calculs du moment actuel, mais aussi des étapes précédentes. On applique alors la règle de la chaîne.

On considère o_t la prédiction du réseau au moment t. On traite la sequence complète (ex : une phrase complète) comme un seul exemple d'apprentissage, donc l'erreur totale est la somme des erreurs à chaque moment (chaque mot).

Le but est de calculer le gradient de l'erreur pour les paramètres *U, V, W* et d'apprendre de bons paramètres en utilisant Stochastic Gradient Descent (SGD). Comme on a sommé les

erreurs, on somme également les gradients à chaque étape pour un seul exemple d'apprentissage :

$\frac{\delta E}{\delta W} = \sum_t \frac{\delta E_t}{\delta W}$ **Equation 2.3**

Pour calculer ces gradients, nous utilisons la règle de la chaîne. Par exemple l'erreur au moment $t = 3$.

$\frac{\delta E_3}{\delta V} = \frac{\delta E_3}{\delta o_3}\frac{\delta o_3}{\delta V} = \frac{\delta E_3}{\delta o_3}\frac{\delta o_3}{\delta z_3}\frac{\delta z_3}{\delta V}$ **Equation 2.4**

Avec $z_3 = V s_3$. La remarque importante ici c'est que $\frac{\delta E_3}{\delta V}$ dépend uniquement des valeurs du moment actuel o_3, s_3.

Mais c'est différent pour $\frac{\delta E_3}{\delta W}$ (et pour *U*). Pour comprendre pourquoi, on écrit la règle en chaîne suivante :

$\frac{\delta E_3}{\delta W} = \frac{\delta E_3}{\delta o_3}\frac{\delta o_3}{\delta s_3}\frac{\delta s_3}{\delta W}$ **Equation 2.5**

On note que $s_3 = f(Ux_3 + Ws_2)$ dépend de s_2 qui dépend de W et de s_1 et ainsi de suite. Donc si on prend la dérivée par rapport à W on ne peut pas traiter $s_2$comme une constante, on doit appliquer à nouveau la règle de la chaîne et ce qu'on obtient, c'est ceci :

$\frac{\delta E_3}{\delta W} = \sum_{k=0}^{3} \frac{\delta E_3}{\delta o_3}\frac{\delta o_3}{\delta s_3}\frac{\delta s_3}{\delta s_k}\frac{\delta s_k}{\delta W}$ **Equation 2.6**

On somme la contribution au gradient de chaque moment. En d'autres termes, puisque W est utilisé, à chaque moment jusqu'à la sortie, on doit retro propager le gradient de $t = 3$ vers $t = 0$ à travers tout le réseau.

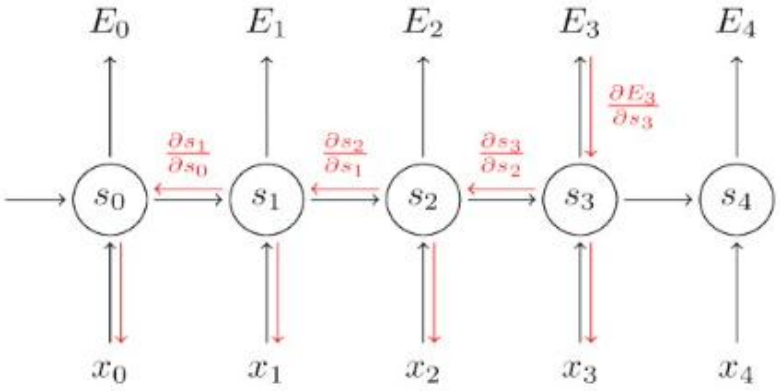


Figure 2.16: Backpropagation Through Time [Web27].

2.2.4 Applications

Les RNN ont connu un grand succès dans de nombreuses tâches de reconnaissance de texte. Les plus grands exploits des RNN ont été accomplis par l'architecture LSTM car ils sont bien meilleurs pour capturer des dépendances à long terme.

- **Modélisation du langage et génération de texte :** tout en prenant note des mots précédents pour prédire les mots suivants dans une séquence de texte.
- **Traduction automatique:** La traduction automatique est similaire à la modélisation de langue dans la mesure où l'entrée est une séquence de mots dans une langue source, nous voulons générer une séquence de mots dans une langue cible, notre sortie ne démarre pas tant que nous n'avons pas la phrase complète car la signification d'un mot dans une phrase peut être différente de la signification de ce mot lorsqu'il est écrit seul.
- **La reconnaissance vocale :** la reconnaissance vocale est l'ensemble des techniques qui consistent à générer du texte écrit à partir d'un fichier vocal.

2.3 Long short term memory (LSTM)

Les RNN ont de la difficulté à apprendre les dépendances à long terme et les interactions entre des mots qui sont éloignés les uns des autres. C'est un problème car parfois la signification d'un mot est déterminée par quelque chose qui a été mentionné plusieurs phrases avant. Dans ce cas précis si la fonction d'activation est une $tanh$ ou bien une sigmoïde alors nous aurons le problème du vanishing gradient.

C'est pour cette raison qu'on se tourne vers la fonction d'activation ReLu qui elle ne souffre pas de ce problème, mais il existe une solution encore plus utilisé c'est l'utilisation des architectures **Long Short term memory (LSTM)** qui peuvent apprendre les dépendances à long terme.

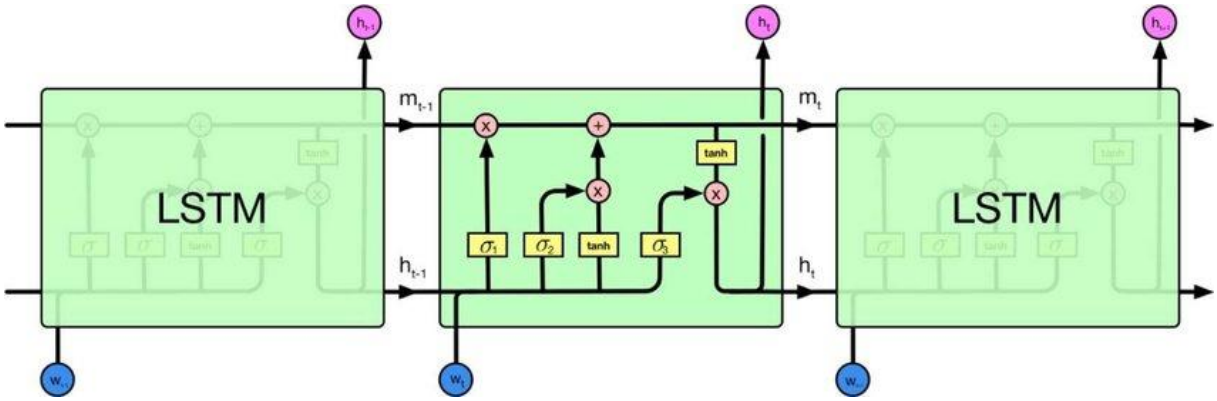


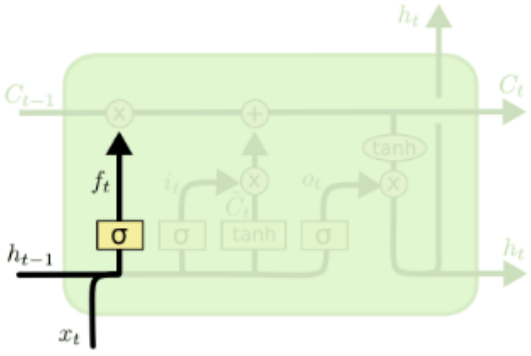
Figure 2.17 : architecture d'un réseau LSTM [Web28].

2.3.1 Fonctionnement des LSTM

LSTM, qui signifie *Long Short-Term Memory*, est une cellule composée de trois « portes » : ce sont des zones de calculs qui régulent le flot d'informations (en réalisant des actions spécifiques). On a également deux types de sorties (nommées états).

- Forget gate (porte d'oubli).
- Input gate (porte d'entrée).
- Output gate (porte de sortie).
- Hidden state (état caché).
- Cell state (état de la cellule).

a- Porte d'oubli (forget gate)

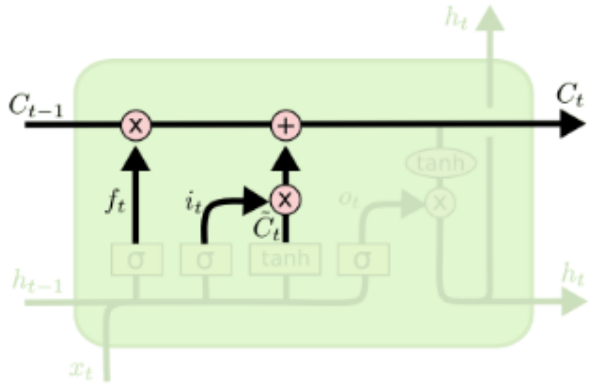


$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

Figure 2.18 : première étape de déroulement d'un LSTM : passage par la porte oubli [Web28].

Cette porte décide de quelle information doit être conservée ou jetée : l'information de l'état caché précédent h_{t-1} est concaténée à la donnée en entrée x_t puis on y applique la fonction sigmoïde afin de normaliser les valeurs entre 0 et 1. Si la sortie du sigmoïde est proche de 0, cela signifie qu'on doit oublier l'information et si on est proche de 1 alors il faut la mémoriser pour la suite.

b- Porte d'entrée (input gate)



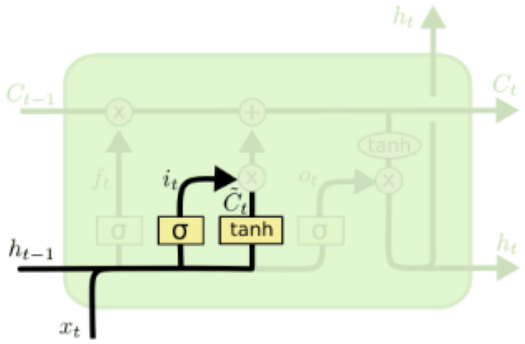
$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

Figure 2.19 : deuxième étape de déroulement d'un LSTM : passage par la porte entrée [Web28].

La porte d'entrée a pour rôle d'extraire l'information de la donnée courante on va appliquer en parallèle une sigmoïde aux deux données concaténées (c_t porte précédente) et une $tanh$.

- Sigmoïde va renvoyer un vecteur pour lequel une coordonnée proche de 0 signifie que la coordonnée en position équivalente dans le vecteur concaténé n'est pas importante. A l'inverse, une coordonnée proche de 1 sera jugée « importante » (i.e. utile pour la prédiction que cherche à faire le LSTM).
- Tanh va simplement normaliser les valeurs (les écraser) entre -1 et 1 pour éviter les problèmes de surcharge de l'ordinateur en calculs.
- Le produit des deux permettra donc de ne garder que les informations importantes, les autres étant quasiment remplacées par 0.

c- Etat de la cellule (cell state)



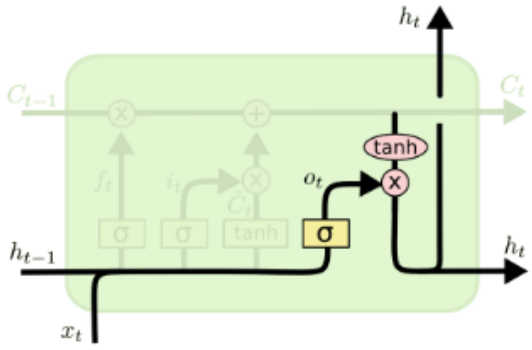
$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

Figure 2.20 : troisième étape de déroulement d'un LSTM : passage par l'état de cellule [Web28].

On parle de l'état de la cellule avant d'aborder la dernière porte (porte de sortie), car la valeur calculée ici est utilisée dedans.

L'état de la cellule se calcule assez simplement à partir de la porte d'oubli et de la porte d'entrée : d'abord on multiplie coordonnée à coordonnée la sortie de l'oubli f_t avec l'ancien état de la cellulec_{t-1}. Cela permet d'oublier certaines informations de l'état précédent qui ne servent pas pour la nouvelle prédiction à faire. Ensuite, on additionne le tout (coordonnée à coordonnée) avec la sortie de la porte d'entrée$i_t * \check{C}_t$, ce qui permet d'enregistrer dans l'état de la cellule ce que le LSTM (parmi les entrées et l'état caché précédent) a jugé pertinent**.**

d- Porte de sortie (output gate)



$$o_t = \sigma\left(W_o \; [h_{t-1}, x_t] \; + \; b_o\right)$$
$$h_t = o_t * \tanh\left(C_t\right)$$

Figure 2.21 : quatrième étape de déroulement d'un LSTM : passage par la porte sortie [Web28].

Dernière étape : la porte de sortie doit décider de quel sera le prochain état caché, qui contient des informations sur les entrées précédentes du réseau et sert aux prédictions.

Pour ce faire, le nouvel état de la cellule calculé juste avant est normalisé entre -1 et 1 grâce à tanh. Le vecteur concaténé de l'entrée courante avec l'état caché précédent passe, pour sa part, dans une fonction sigmoïde dont le but est de décider des informations à conserver (proche de 0 signifie que l'on oublie, et proche de 1 que l'on va conserver cette coordonnée de l'état de la cellule).

Mais il existe un autre inconvénient des LSTM conventionnels est qu'ils ne peuvent utiliser que le contexte précédent. Il existe une nouvelle variante qui devient très populaire, à savoir les RNN bidirectionnels (BRNN). Ils traitent les données dans les deux sens à l'aide de deux couches cachées distinctes. La combinaison de ces deux couches vous donnera des informations complètes sur le contexte. Les BRNN ont été utilisés avec succès dans les modèles de reconnaissance vocale [Web27].

3 L'apprentissage profond

3.1 Introduction

Le processus d'apprentissage dans l'apprentissage profond revient à l'entraînement du réseau neuronal en utilisant des optimiseurs itératifs, qui ne font que conduire la fonction de coût à une très faible valeur. Nous pouvons utiliser différents algorithmes pour effectuer l'apprentissage, mais l'algorithme le plus utilisé est l'algorithme itératif d'optimisation par la descente de gradient. Le processus d'apprentissage revient au problème d'optimisation où il s'agit de minimiser ou de maximiser une fonction $f(x)$. Cette fonction est appelée la fonction objective, les auteurs dans [22] l'ont appelée aussi la fonction de coût, la fonction de perte et la fonction d'erreur. Pendant l'entrainement, l'algorithme tente d'identifier le minimum global sans tomber dans le piège du minimum local. Le schéma suivant réalisé par les auteurs en [22] est une illustration de ce concept.

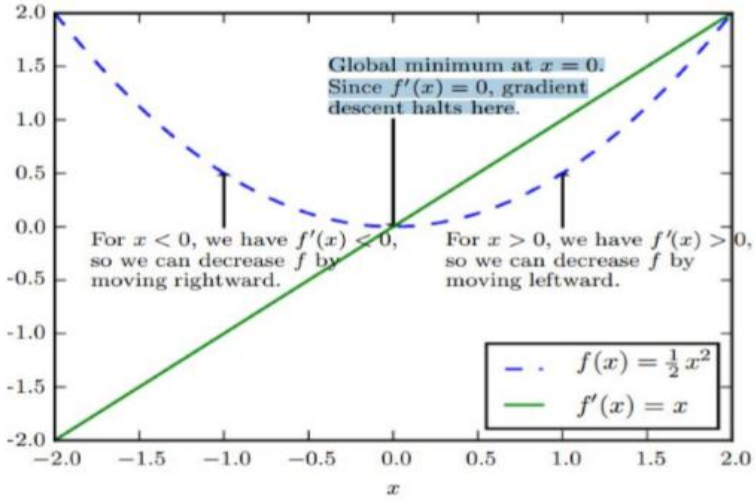


Figure 2.22 : Une illustration du processus de recherche de l'optimum.

3.2 Optimisation

3.2.1 Les variantes de la descente de gradient

La descente de gradient est la méthode la plus célèbre pour optimiser un réseau de neurones [23], où on doit minimiser une fonction objective $f(x)$ caractérisée par les paramètres x en les mettant à jour dansla direction opposée de celle de gradient de la fonction objective.

Il existe trois variantes de cette méthode. En fonction de la quantité de données, nous ferons un compromis entre la précision de la mise à jour des paramètres et le temps d'exécution de cette mise à jour.

-Batch gradient descent.

-Descente de gradient stochastique.

-Mini-batch gradient descent.

3.2.2 Algorithmes d'optimisation de la descente de gradient

On peut citer quelques algorithmes largement utilisés par la communauté d'apprentissage profond pour résoudre les challenges d'optimisation des réseaux de neurones avec la descente de gradient.

-Momentum.

-Nesterov accelerated gradient.

-Adagrad.

-RMSprop.

-Adam optimizer.

4 Les fonctions d’activation

La fonction d’activation est une fonction mathématique appliquée à un signal en sortie d’un neurone artificiel. Le terme de "fonction d’activation" vient de l’équivalent biologique "potentiel d’activation", seuil de stimulation qui, une fois atteint entraîne une réponse du neurone. La fonction d’activation est souvent une fonction non-linéaire. Leur but est de permettre aux réseaux de neurones d’apprendre des fonctions plus complexes qu’une simple régression linéaire car le fait de multiplier les poids d’une couche cachée est juste une transformation linéaire.

Le choix des fonctions d'activation dans les réseaux profonds a un effet significatif sur la dynamique d'entraînement et la performance des tâches. Actuellement, la fonction d'activation largement utilisée est l'unité linéaire rectifiée (ReLU).

Daprès les travaux de [24], la meilleure fonction d'activation découverte, est f (x) = x · sigmoïde (βx), que nous appelons Swish, qui a tendance à mieux fonctionner que ReLU sur des modèles plus profonds dans un certain nombre de jeux de données difficiles.

Quelques fonctions d’activation **sont données par la source suivante** [24].

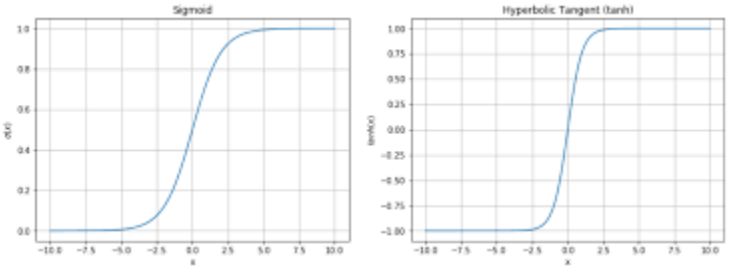


Figure 2.23: (à gauche) Sigmoïde. (À droite) tangente hyperbolique [Web29].

5 L’apprentissage profond pour la détection et la reconnaissance de texte

Dans le passé, la détection et la reconnaissance de texte sont généralement présentées comme deux sous-problèmes indépendants qui sont combinés ensemble pour effectuer la lecture de texte à partir d'images.

Récemment, nous avons vu l'essor des moteurs OCR qui font les deux opérations en même temps, et les efforts pour construire de tels systèmes ont pris un élan considérable en tant que nouvelle tendance dans la communauté des chercheurs.

5.1 L’apprentissage profond dans la détection de texte

La détection de texte peut être placée sous le domaine de la détection d'objets, de sorte que l'utilisation de modèles de détection d'objets peut être adapté pour le domaine de la détection de texte avec quelques petites variations même si la détection de texte a un ensemble différent de caractéristiques et de défis qui nécessitent des méthodologies et des solutions uniques.

5.2 Méthodes de détection du texte

Il existe de nombreuses méthodes dans le domaine de la détection de texte et elles sont généralement classées sous 3 tendances, pipeline simplifié, unité de prédiction différente, cible spécifiée.

- **Pipeline simplifié (simplified pipeline):** est une méthode qui produit les mêmes résultats des pipelines complexes mais avec de meilleures performances en terme d'exécution (côté calcul) en raison du pipeline simplifié.
- **Unité de prédiction différente (different prediction unit):** est une méthode qui tente de prédire le texte à différents niveaux, par exemple au niveau ligne ou au niveau mot ou au niveau caractère ou même au niveau sous-caractère.
- **Cible spécifiée (specific target):** est une méthode qui tentent de détecter une cible spécifique dans une image qui à son tour va aboutir à identifier le texte dans la dite image, par exemple de longues lignes de texte, les longues lignes de texte ont une longue forme rectangulaire spécifique qui peut être facilement détectée dans une image.

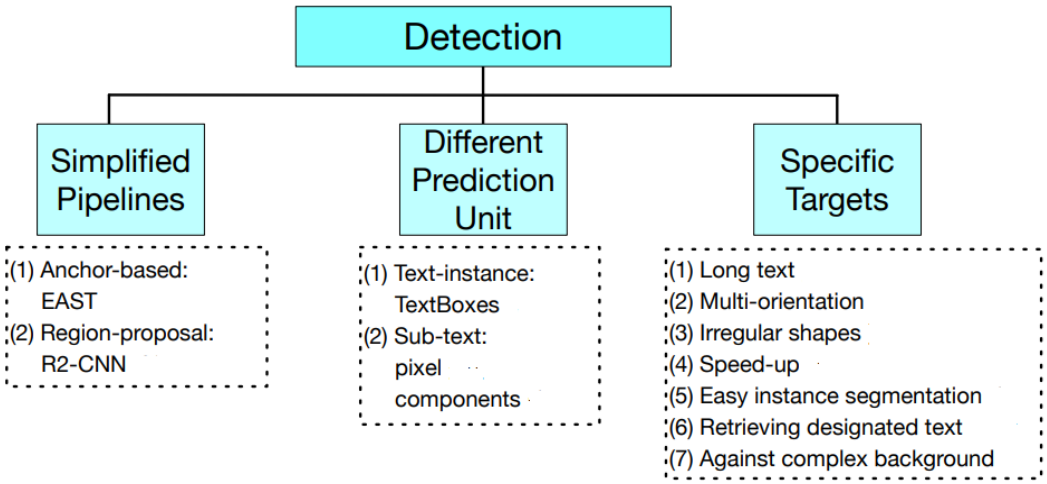


Figure 2.24: Résumé des méthodes de détection dans les 3 tendances [25].

5.3 Modèles de détection d'objets utilisés dans la détection du texte

5.3.1 R-CNN (Regions with CNN)

Pour contourner le problème de la sélection d'un grand nombre de régions, Ross Girshick et al. [26] ont proposé une méthode où nous utilisons la recherche sélective pour extraire seulement 2000 régions de l'image et il les a appelé propositions de régions. Le réseau est composé de trois parties principales, l'extracteur de région, l'extracteur de caractéristiques et enfin le classificateur. Un algorithme de proposition de région extrait ROI, puis chaque région est alimentée en un classificateur et enfin les caractéristiques extraites sont classées[26].

5.3.2 FAST RCNN

Fast R-CNN s'appuie sur des travaux antérieurs pour classer efficacement les propositions d'objets à l'aide de réseaux convolutionnels profonds. Par rapport aux travaux précédents, Fast R-CNN utilise plusieurs innovations pour améliorer la vitesse d'entraînement et de test tout en augmentant la précision de détection.. Fast R-CNN entraine le réseau very deep VGG16 9 fois plus vite que R-CNN, et 213 fois plus vite au test-time [27].

5.3.3 FASTER RCNN

Un réseau de proposition de région (RPN) qui partage des caractéristiques de convolution d'image complètement avec le réseau de détection [28].

5.3.4 **R2CNN (**Rotational Region CNN)

Cette méthode Région de rotation CNN (R2CNN) est proposée pour détecter des textes à orientation arbitraire dans des images de scènes naturelles. R2CNN est basé sur Faster R-CNN [29].

5.3.5 **SSD (**Single ShotMulti Box Detector)

Cette méthode utilise un seul réseau neuronal profond pour détecter des objets dans des images.

Cette approche, nommée SSD, discrétise l'espace de sortie des boîtes englobantes (bounding boxes) en un ensemble de boîtes par défaut sur différents rapports d'aspect (aspect ratio) . Au moment de la prédiction, le réseau génère des scores pour la présence de chaque catégorie d'objets dans chaque boîte par défaut et produit des ajustements de la boîte pour mieux correspondre à la forme de l'objet [30].

5.3.6 **YOLO (**You Only Look Once)

Un seul réseau de neurones prédit des boîtes englobantes et des probabilités de classe directement à partir d'images complètes en une seule évaluation. Étant donné que l'ensemble du pipeline de détection est un réseau unique, le modèle YOLO traite les images en temps réel à 45 images par seconde. Une version plus petite du réseau, Fast YOLO, traite un nombre incroyable de 155 images par seconde [31].

5.4 L’apprentissage profond dans la reconnaissance de texte

L'étape de reconnaissance de texte prend en entrée les cartes de caractéristiques de sortie des réseaux convolutionnels et les régions d’intérêt générées, Cette procédure de reconnaissance de texte doit, produire une séquence prédite correspondant au texte à l'intérieur pour chaque région d’intérêt.

Pour prédire le contenu des régions d’intérêt, on utilise une architecture RNN pour traiter la carte de caractéristiques de gauche à droite séquentiellement et essayé de prédire une étiquette pour chaque emplacement sur cette carte.

5.5 Méthodes de reconnaissance de texte

Les méthodes de reconnaissance de texte sont comme suit :

- **CTC (Connectionist Temporal Classification)-based methods:** La première application de CTC dans le domaine OCR peut être attribuée au système de reconnaissance de texte manuscrite de Graves et al [32].
  CTC calcule la probabilité conditionnelle $P\ (L \mid Y)$, où $Y\ =\ y1, \ldots, yT$ représentent la prédiction par image de RNN et L est la séquence d'étiquettes, de sorte que le réseau peut être entrainé en utilisant uniquement l'étiquette de niveau de séquence comme supervision.

- **Attention-based methods :**Le mécanisme d'attention a d'abord été présenté dans [33] pour améliorer les performances des systèmes de traduction automatique neuronale.Un aspect particulièrement étudié est l'attention visuelle: Ce principe a un impact important sur le calcul neuronal car nous devons sélectionner les informations les plus pertinentes, plutôt que d'utiliser toutes les informations disponibles et c'est ce que fait le mécanisme d'attention qui se concentre sur les parties spécifiques importantes de l'entrée. Le mécanisme d’attention a été appliqué dans la reconnaissance vocale, la traduction, le raisonnement et l'identification visuelle des objets.

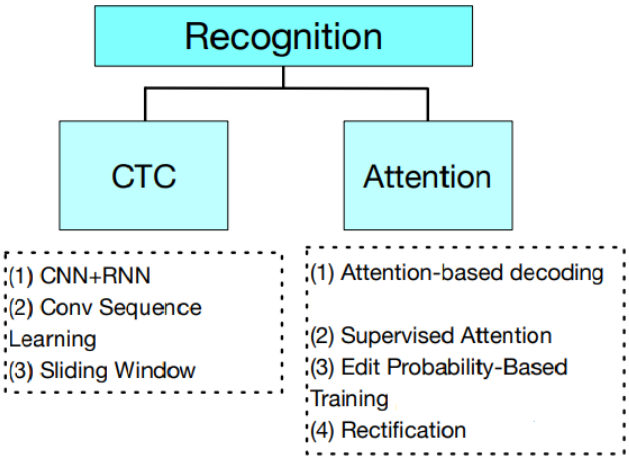


Figure 2.25: Résumé des méthodes de reconnaissance célèbres

Un excellent document [25] résume et va profondément dans les détails de toutes les méthodes de détection et de reconnaissance de texte.

5.6 Modèles de reconnaissance et classification utilisés dans la reconnaissance du texte

5.6.1 LSTM-RNN (long short term mermory recurrent neural networks)

Les réseaux neuronaux récurrents de mémoire à court terme (LSTM-RNN) sont l'un des classificateurs dynamiques les plus puissants connus du public [34].

5.6.2 CRNN (Convolutional Recurrent Neural Networks) .

Une architecture de réseau neuronal, qui intègre l'extraction de caractéristiques, la modélisation de séquence et la transcription dans un cadre unifié [35].

5.6.3 STN (spatial transformer network)

Le Transformateur Spatial, qui permet explicitement la manipulation spatiale des données au sein du réseau neuronal donnant aux réseaux la capacité de transformer activement des cartes des caractéristiques dans l'espace sans aucune supervision d’entrainement supplémentaire ni modification du processus d'optimisation [36].

6 Conclusion

Dans ce chapitre, nous avons parlé du concept d'apprentissage profond et de la façon dont ils sont différents des architectures de ML, nous avons vu quelques architectures communes ainsi que leur fonctionnement et comment ils sont entrainé et optimisés, nous avons également parlé de la façon dont ces architectures sont utilisées dans l'application d'OCR, et nous avons expliqué le pipeline des deux méthodes qui vont être utilisées dans le chapitre suivant.

Chapitre 3 : Implémentation et Résultats.

1 Introduction

Pour implémenter un système de détection et de reconnaissance de texte, nous avons utilisé le modèle de détection EAST **(An Efficient and Accurate SceneText Detector)** pour la détection de texte et le réseau récurrent LSTM (Long short-term memory) tesseract v4 pour la reconnaissance de texte. Nous avons également utilisé le langage python pour la programmation sous l'environnement pycharm, Ce chapitre présente les différents logiciels et les bibliothèques utilisés, les modèles de l'apprentissage profond implémenté ainsi que les résultats obtenus.

2 Structure du projet

Pour commencer, nous donnons une image en entrée au détecteur de texte EAST qui va détecter la présence de texte dans notre image, EAST nous donnera les coordonnées de bounding box estimées autour du texte. Les coordonnées (x,y) de bounding box des régions d'intérêts du texte résultants sont données à l'algorithme de reconnaissance de texte LSTM de tesseract V4 , la sortie du LSTM nous donne nos résultats réels sous forme de chaine de caractères ,qui correspondent à ceux de l'image et qui sont enregistrés dans un fichier texte .txt, voir **figure 3.1**.

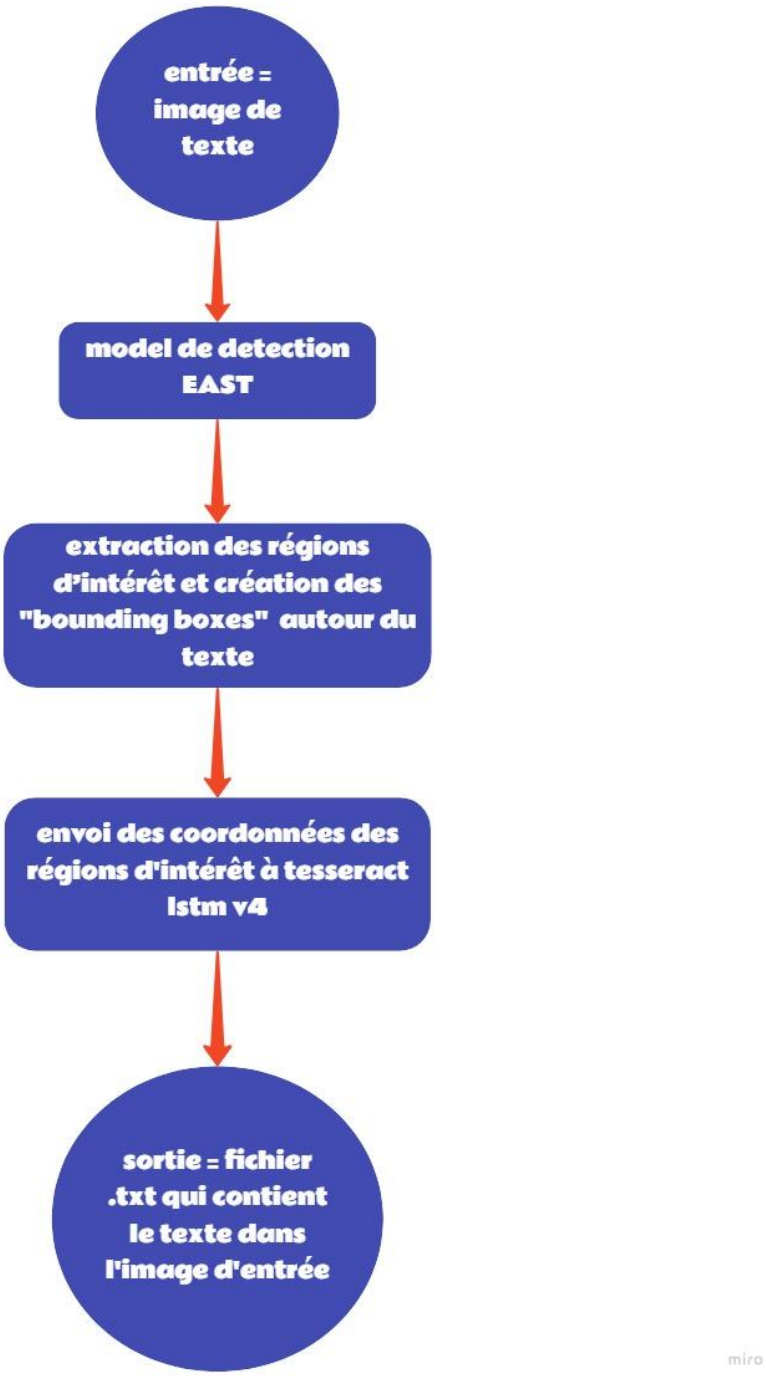


Figure 3.1 : Les étapes de système OCR implémenté.

3 Les outils utilisés

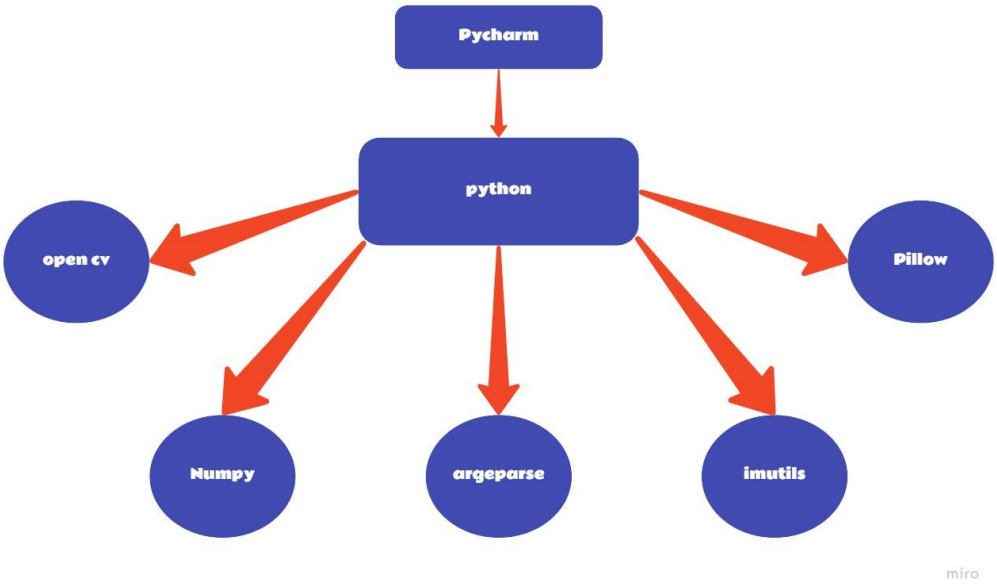


Figure 3.2 : Les bibliothèques utilisées sous l'environnement Pycharm.

3.1 L'environnement Pycharm

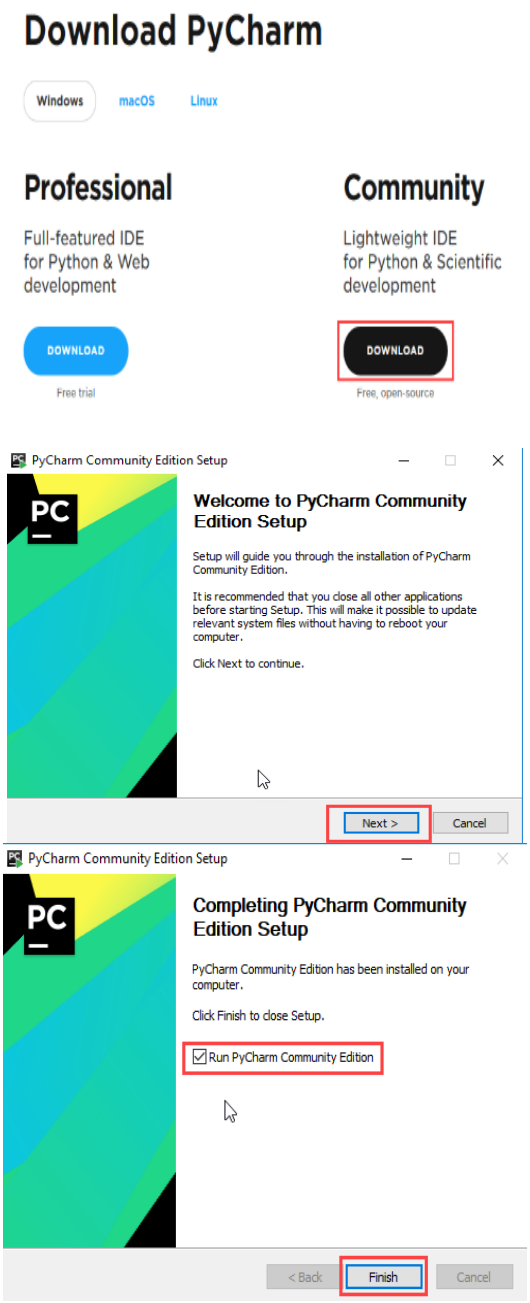
PyCharm est un environnement de développement intégré (IDE) utilisé dans la programmation, spécifiquement pour le langage Python. Il est développé par la société tchèque JetBrains. Il fournit l'analyse de code, le débogage graphique, le test de l'unité intégré, l'intégration avec des systèmes de contrôle de version (VCS), et prend en charge le développement Web avec Django ainsi que la science des données avec Anaconda. PyCharm est multiplateforme, avec les versions Windows, macOS et Linux. L'édition communautaire est publiée sous la licence Apache, et il existe également l'édition professionnelle publiée sous une licence propriétaire avec des fonctionnalités supplémentaires.

Nous avons choisi d'utiliser l'environnement pycharm en raison des nombreuses fonctionnalités offertes :

- La complétion de code.
- L'Interface graphique simplifiée et conviviale.
- Les supports des outils de développement de python (open CV, pillow, tensorflow, numpy,etc.).

L'installation de l'IDE PyCharm se fait par le

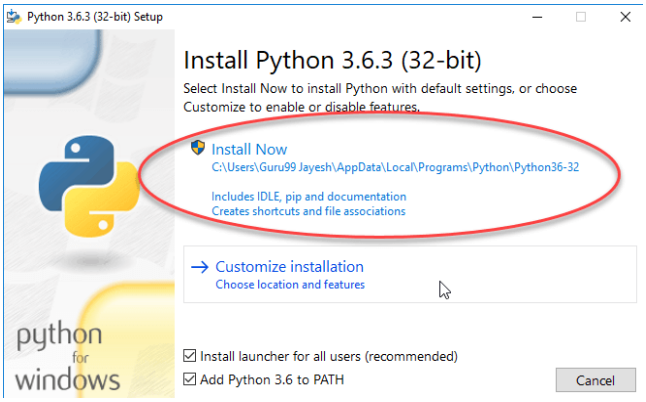
Téléchargement du site officiel.

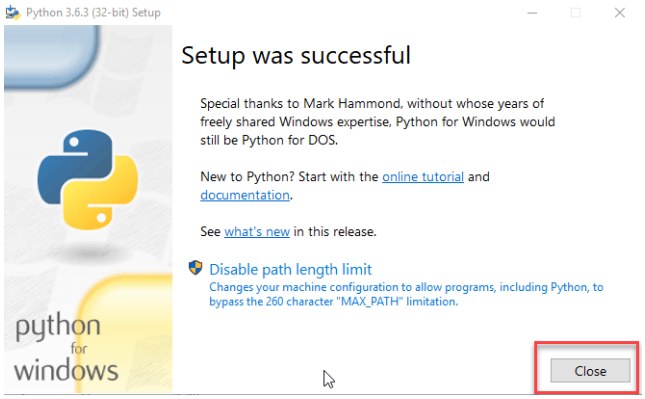


3.2 Le langage de programmation Python

Python est un langage de programmation interprété, de haut niveau et à usage général, Créé par Guido van Rossum et sorti pour la première fois en 1991, le python est le langage le plus utilisé dans le domaine de l'intelligence artificielle et du deep learning, nous avons choisi d'utiliser python parce qu'il supporte les outils de vision artificielle tels que open CV, Pillow et la gestion des données comme theano,tensorflow et numpy.

L'Installation de Python revient à le télécharger du site officiel.





3.3 Installation des bibliothèques et des outils utilisés

Opencv

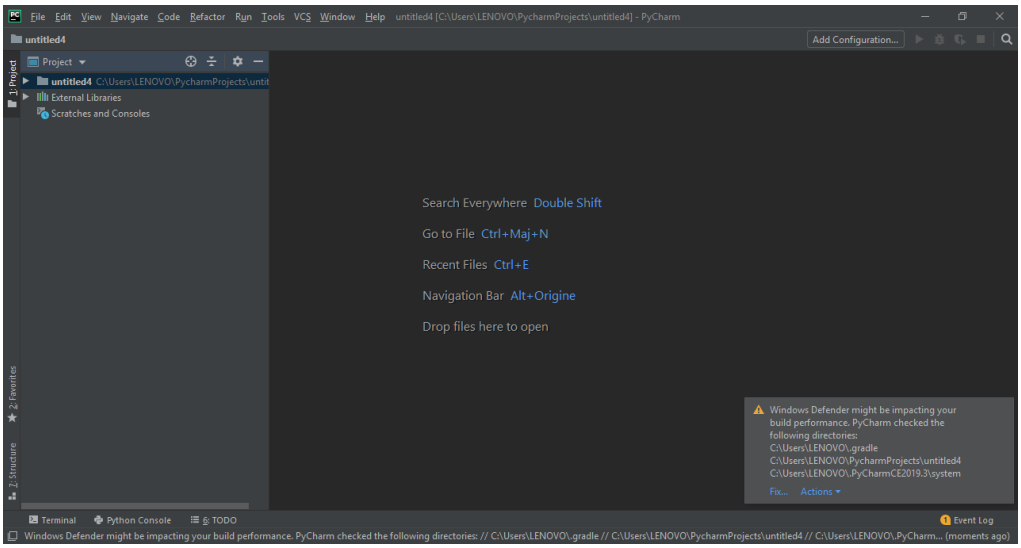
OpenCV (pour Open Computer Vision) est une bibliothèque graphique libre
Initialement développée par Intel, spécialisée dans le traitement d'images en temps réel
La bibliothèque open CV met à disposition de nombreuses fonctionnalités permettant de créer des programmes partant de la manipulation des données brutes jusqu'à la création des interfaces graphiques basiques.

Fonctionnalités :

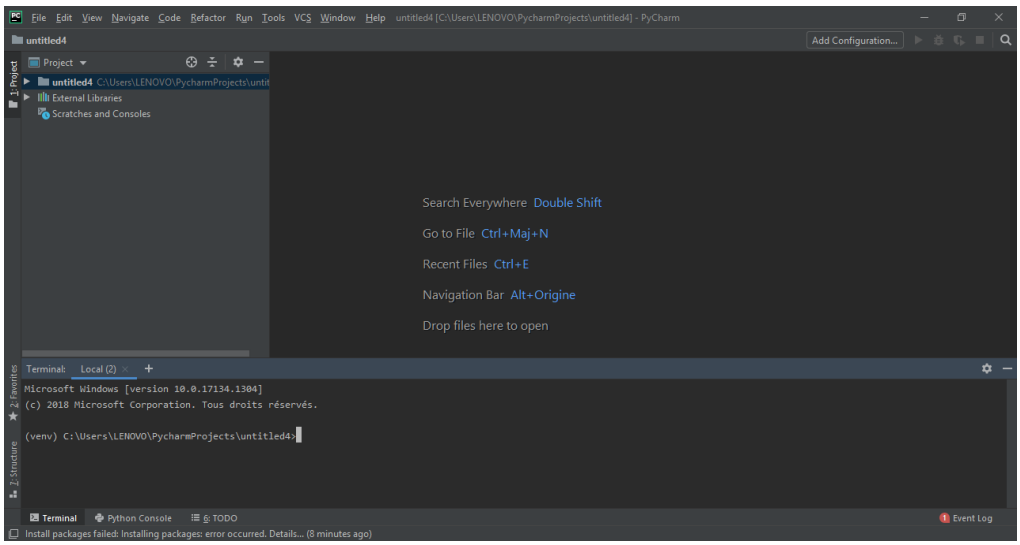
- Traitement d'images et vidéos.
- Algorithmes d'apprentissage (Kmeans, adaboost … etc).
- Calculs matriciels.

Installation d'open CV

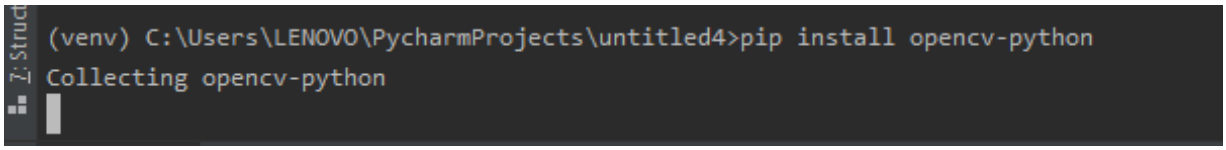
1ere étape : on crée un nouveau projet sous l'environnement IDE de pycharm :



2eme étape : Le terminal de commande est visualisé (coin inférieur gauche)



3eme étape : A travers la commande « pip install opencv-python » on rentre dans l'univers d'openCV-python.

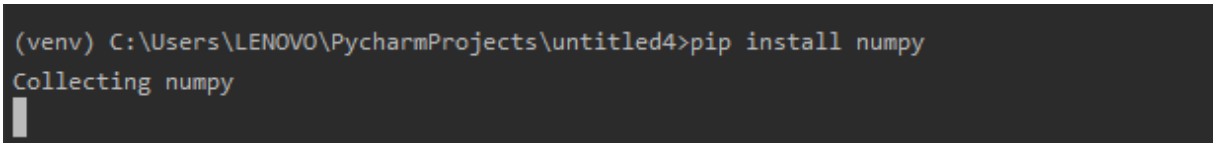


Numpy

NumPy est une bibliothèque du langage de programmation Python, destinée à manipuler des matrices ou tableaux multidimensionnels ainsi que des fonctions mathématiques opérant sur ces tableaux. Elle est très importante dans les applications d'apprentissage profond.

Installation de Numpy

Mêmes étapes précédentes en utilisant la commande « pip install numpy ».



Argparse

Le principe d'argparse est qu'il n'est pas nécessaire d'aller dans le code et d'apporter des modifications au script. Il donne à l'utilisateur la possibilité d'entrer des arguments de ligne de commande offrant une flexibilité, Le module argparse facilite l'écriture des lignes de commande. Le programme définit les arguments dont il a besoin et argparse trouvera comment les analyser à partir de sys.argv. Le module argparse génère également automatiquement des messages d'aide et d'utilisation et génère des erreurs lorsque les utilisateurs donnent au programme des arguments non valides.

Installation d'argparse

Même étapes précédentes mais cette fois la commande est « pip install argparse ».

```
(venv) C:\Users\LENOVO\PycharmProjects\untitled4>pip install argparse
Collecting argparse
```

Imutils

Une série de fonctions pratiques pour rendre les fonctions de traitement d'images de base telles que la translation, la rotation, le redimensionnement, la normalisation, l'affichage des images Matplotlib, le tri des contours, la détection des contours, encore plus facile avec OpenCV.

Installation d'imutils

La commande pour installer imutils est « pip install imutils ».

```
(venv) C:\Users\LENOVO\PycharmProjects\untitled4>pip install imutils
Collecting imutils
```

Pillow

Python Imaging Library (ou PIL) est une bibliothèque de traitement d'images pour le langage de programmation Python. Elle permet d'ouvrir, de manipuler, et de sauvegarder différents formats de fichiers graphiques.

Installation de Pillow

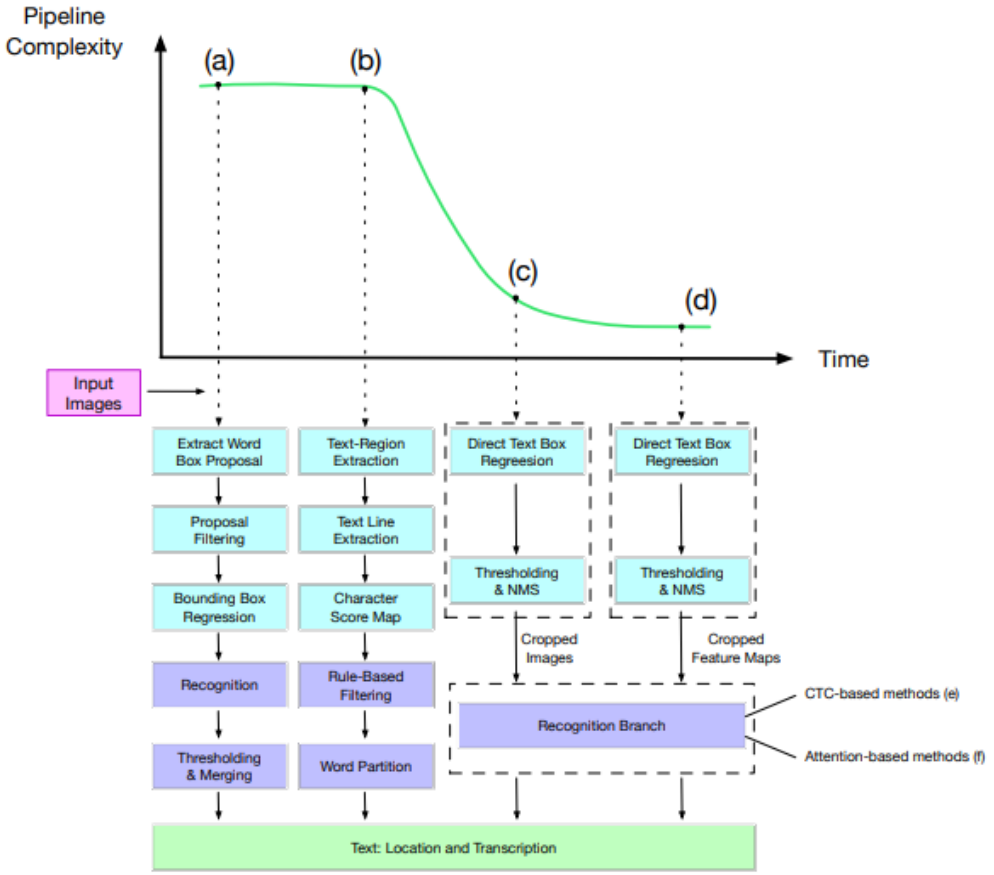
La commande pour installer Pillow est « pip install pillow ».

```
(venv) C:\Users\LENOVO\PycharmProjects\untitled4>pip install pillow
Collecting pillow
```

3.4 Le modèle de la détection

3.4.1 Les modèles de détection:

La **Figure 3.3** montre les pipelines typiques de détection et de reconnaissance de texte de scène. (a) et (b) sont des méthodes représentatives en plusieurs étapes (multi stepmethods) (c) et (d) sont des pipelines simplifiés (simplified pipeline) (c) et (d) contient uniquement une branche de détection, et est donc utilisé avec un modèle de reconnaissance distinct. [25].



.Figure 3.3 : Les pipelines typiques de détection et reconnaissance de texte.

3.4.2 Le modèle East (An Efficient and Accurate Scene Text Detector)

Le modèle East est un pipeline prometteur de détection de régions d'intérêt de texte dans une image. Selon [37], le pipeline EAST (composé d'un réseau entièrement convolutif (FCN) et d'un algorithme de suppression non maximale (NMS) sensible à la localité) est capable de détecter des lignes de texte sans utiliser d'algorithmes traditionnels coûteux et de traiter des images avec une résolution de 1280x720 à environ 16 images par seconde (fps).

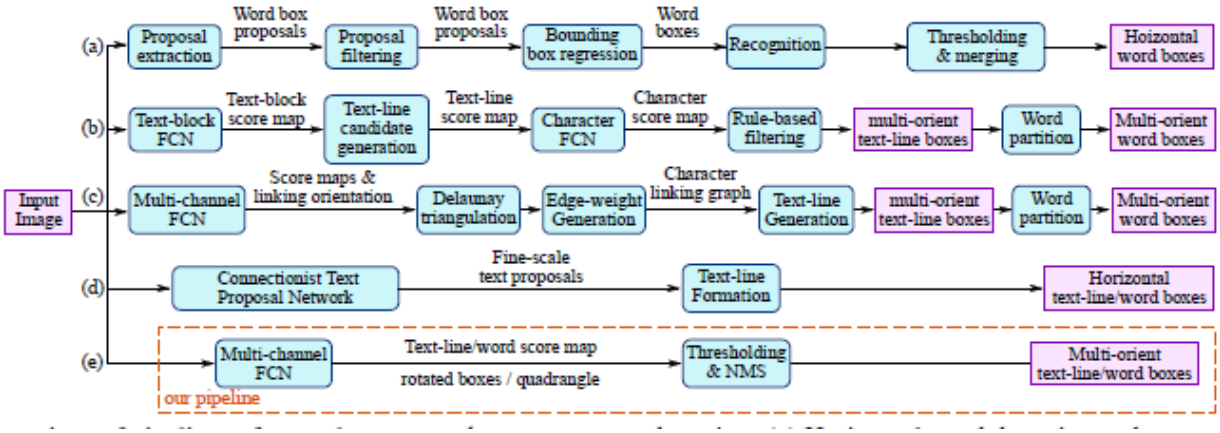


Figure 3.4 : La différence entre le pipline utilisé (e) et les piplines usuels (a),(b),(c),(d) mentionné aussi dans la figure precedente.

- Cette méthode de détection de texte comprend deux étapes: un réseau FCN (Fully-convolutional) et une étape de fusion des bounding boxes estimé NMS. Le FCN produit directement des régions de texte, à l'exclusion des étapes intermédiaires redondantes et longues qui existent dans les pipelines usuels.
- Le pipeline est flexible pour produire des prédictions au niveau du mot ou au niveau de la ligne, dont les formes géométriques peuvent être des rectangles tournés (RBOX) avec certains angles ou des quadrilatères (QUAD), selon les applications spécifiques.
- L'algorithme proposé surpasse considérablement les autres méthodes de précision et de vitesse.

a) Conception du pipeline EAST :

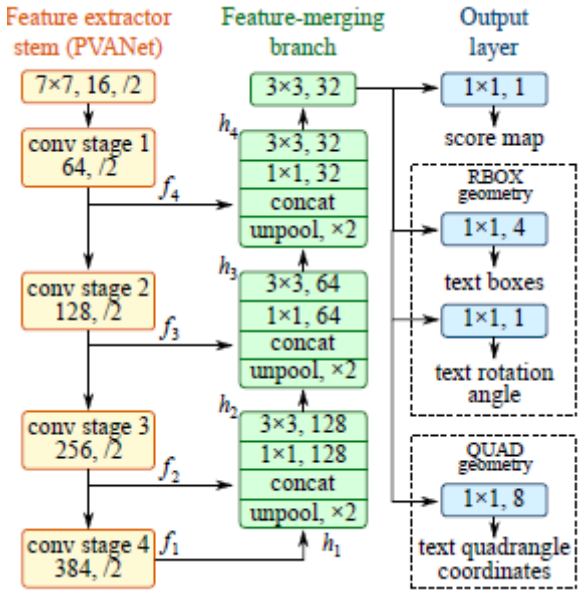


Figure 3.5 : La structure du réseau de détection de texte EAST

Pour créer ce réseau, les développeurs de cet algorithme ont utilisé trois branches combinées en un seul réseau neuronal. Une image introduite dans ce réseau, une chaine d'extraction des caractéristiques (feature extraction channel) avec quatre niveaux de carte de caractéristiques (feature map) notées f_i, et une chaine de fusion des caractéristiques (feature merging channel).

Dans chaque étape de fusion, la carte des caractéristiques de la dernière étape est d'abord introduite dans une couche d'unpooling pour doubler sa taille, puis concaténée avec la carte de caractéristiques d'entités actuelles.h_i Qui est la carte de caractéristiques fusionnée (merged feature map).

La couche de sortie finale contient plusieurs opérations conv1 × 1 pour projeter 32 chaines de cartes de caractéristiques en une chaine de carte de score (score map) $F_s$et une carte de géométrie multichaines F_g La sortie géométrique peut être soit RBOX, soit QUAD.

Le seuillage est ensuite appliqué à la région prédite, où la géométrie dont le score est supérieur au seuil prédéfini est considérée comme valide et enregistrée pour une suppression

non maximale (non maxima supression) ultérieure qui élimine les rectangles à faible score. Les résultats après NMS sont considérés comme la sortie finale du pipeline.

b) Implémentation du modèle EAST

Le modèle est entrainé sur un ensemble de données spécifiques. En utilisant un « freezed model » de tensorflow, le processus permet d'identifier et d'enregistrer tous les paramètres requis (graphique, poids, etc.) dans un seul fichier que nous pouvons facilement utiliser dans notre application.

Un « freezed model » ne contient que les paramètres importants dans l'état final de l'entrainement, généralement un « freezed model » est fourni dans un fichier protobuf. Dans TensorFlow, le fichier protbuf contient la définition du graphique ainsi que les poids du modèle en leur état final. Ainsi, un fichier protobuf est tout ce que nous avons besoin pour pouvoir exécuter un modèle entrainé donné.

3.5 L'OCR Tesseract/Pytesseract pour la reconnaissance

Tesseract est un moteur OCR open source sous licence Apache 2.0 qui aide à lire le texte du document (par exemple, les images pdf, jpg ou png, etc.). Il a été développé par Hewlett Packard dans les années 1980. Et Google a repris le projet à partir de 2006.

Lorsqu'on considère la version Tesseract 4.0, les améliorations suivantes peuvent être notées [38].

- Utilisation du modèle d'apprentissage en profondeur: réseau neuronal récurrent à mémoire à court et long terme (LSTM).
- Comprend un sous-système de réseau neuronal configuré et comprend un identificateur de ligne de texte.
- Amélioration de l'analyse complète de la mise en page, détection de table, détection d'équation, meilleur modèle de langage, recherche de segmentation améliorée et meilleurs modèles manuscrits.

Conception du pipeline tesseract :

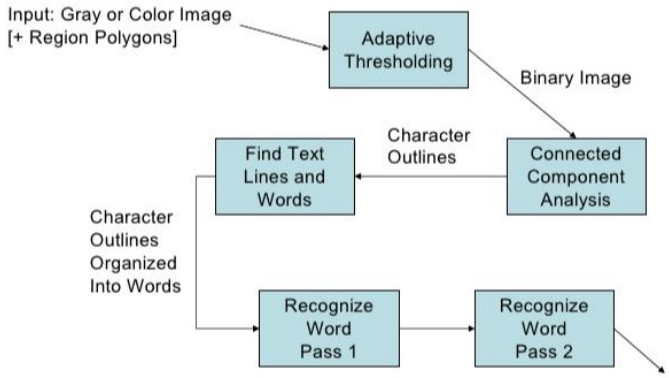


Figure 3.6 : Pipeline de tesseract[Web30].

Au départ, l'entrée est soit en niveaux de gris (grayscale) soit une image couleur. L'entrée est fournie au système et le module adaptive thresholding est appliqué, puis une image binaire sera produite.

La deuxième étape est l'analyse des composants connectés (connected component analysis) qui signifie la recherche de composants distincts dans l'image qui aideront à la localisation du texte.

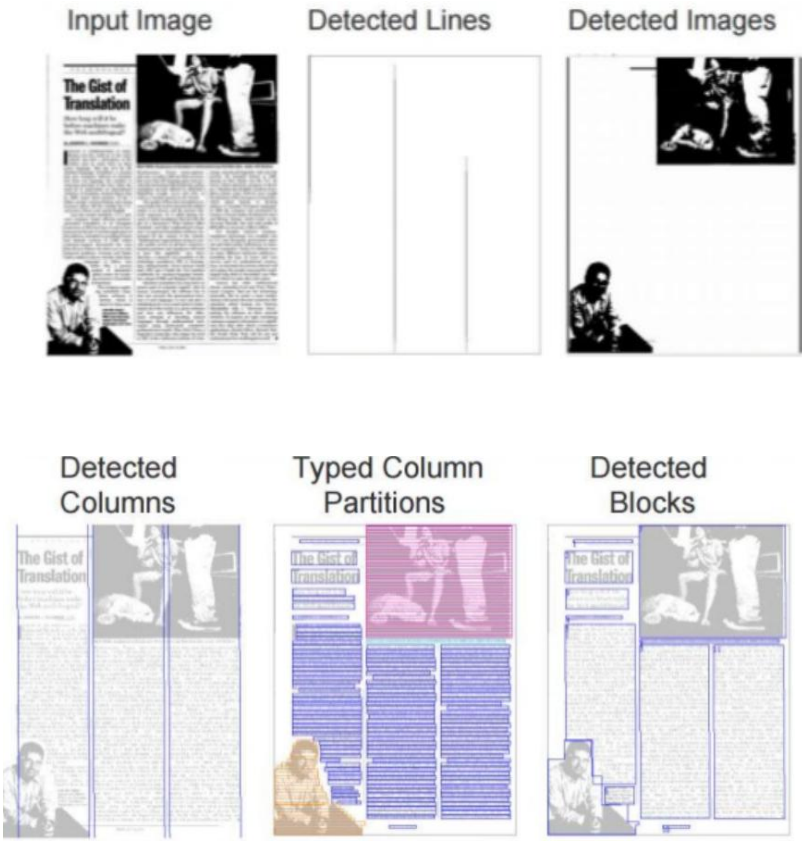


Figure 3.7 Analyse de l'image pour la localisation des régions.

La Figure 3.7 montre la mise en page pour localiser les régions qui contiennent les images et les lignes et la recherche de blocs pour localiser les blocs de texte [Web31].

Après ces deux sous étapes, la zone de texte est identifiée, les caractères sont encore segmentés comme préparation pour l'utilisation dans les prochaines étapes,cela signifie la fin de l'étape d'analyse des composants connectés.

Pour l'étape de reconnaissance, tesseract fait 2 itérations de reconnaissance (adaptive recognition), cela nécessite deux passages dans chaque page de texte, le deuxième passage pour corriger des fautes de classification en haut de page à cause d'un contexte trouvé en bas de même page, cette méthode améliore considérablement les performances en reconnaissance [38].

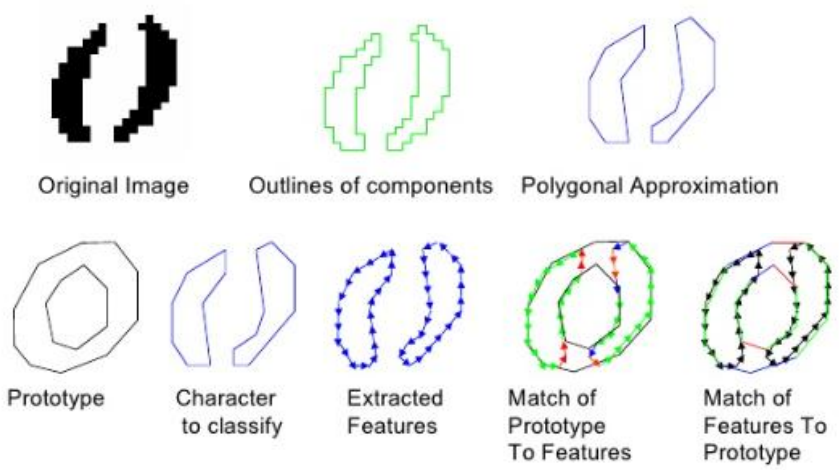


Figure 3.8 : Procédure de classification du caractère « o » dans tesseract [Web30].

Tesseract est un moteur OCR qui intègre l'opération de détection et de reconnaissance de texte, mais il n'est pas vraiment efficace dans la partie détection. Pour qu'il fournisse des résultats satisfaisants, l'image donnée en entrée devra être bien coupé autour du texte et d'une très bonne qualité. Tesseract a alors des difficultés à exécuter l'opération d'OCR sur des images de scènes naturelles.

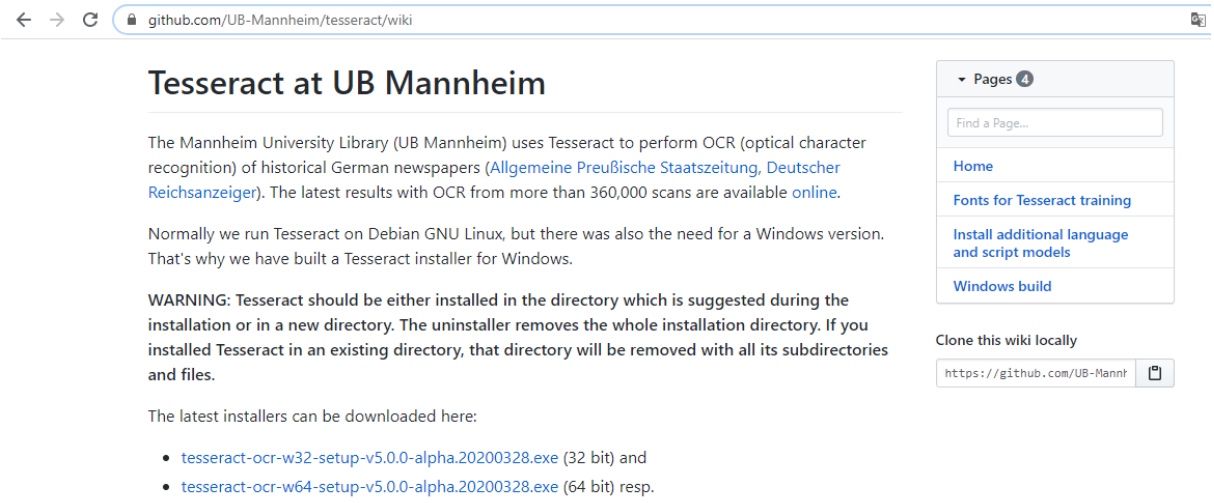
Pytesseract

Python-tesseract (pytesseract) est un outil de reconnaissance optique de caractères (OCR) pour python. Autrement dit, il reconnaîtra et lira le texte intégré aux images.

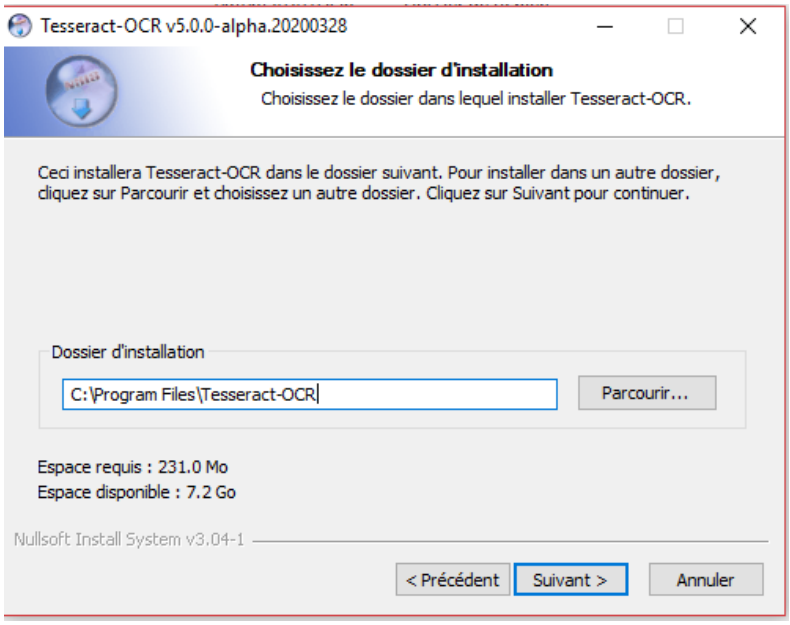
Python-tesseract est un wrapper pour le moteur Tesseract-OCR de Google. Il est également utile comme script autonome pour tesseract, car il peut lire tous les types d'images pris en charge par les bibliothèques d'imagerie Pillow et Leptonica, y compris jpeg, png, gif, bmp, tiff.

Installation de Tesseract et Pytesseract

Nous avons téléchargé Tesseract.



Nous avons installé Tesseract tout en faisant attention au chemin d'installation car nous en aurons besoin.



L’installation de pytesseract dans python se fait de la même manière d’installation des autres bibliothèques, avec la commande « pip install pytesseract ».

```
(venv) C:\Users\LENOVO\PycharmProjects\untitled4>pip install pytesseract
Collecting pytesseract
```

Implémentation de Pytesseract

```
pytesseract.pytesseract.tesseract_cmd = r"C:\Program Files\Tesseract-OCR\tesseract.exe"
```

Il existe 3 paramètres de Tesseract à régler, selon l’application :

Le choix de la langue par le paramètre lang.
Les modes de segmentation de page, par le paramètre psm (exemple de traitement d’images avec une ligne de texte le choix se portera sur le numéro 7 pour psm).

Le mode du moteur OCR par le paramètre oem, qui signifie le type des méthodes utilisés pour la procédure ocr).

- lang

Définirla langue à utiliser. L’anglais est par défaut si aucune n'est spécifiée,

--psm N

Configurer tesseract pour exécuter une seule méthode d’analyse (segmentation) de page suivant l’image en entrée (ex : pour un document on utilise n=1, pour une image de scène naturelle on utilise n=7).

0 = Orientation et détection de script (OSD) uniquement.
1 = Segmentation automatique des pages avec OSD.
2 = Segmentation automatique des pages, mais sans OSD ni OCR.
3 = Segmentation de page entièrement automatique, mais pas d'OSD. (Défaut)
4 = Supposons une seule colonne de texte de tailles variables.
5 = Supposons un seul bloc uniforme de texte aligné verticalement.
6 = Supposons un seul bloc de texte uniforme.
7 = Traitez l'image comme une seule ligne de texte.
8 = Traitez l'image comme un seul mot.
9 = Traitez l'image comme un seul mot dans un cercle.
10 = Traitez l'image comme un seul caractère.

--oem N

Spécifier le mode du moteur OCR. Les options pour N sont:

0 = Tesseract d'origine uniquement.
1 = Réseaux neuronaux LSTM uniquement.
2 = Tesseract + LSTM.
3 = par défaut, basé sur ce qui est disponible.

Voire Annexe C pour plus de détails sur les syntaxes de code de configuration de Tesseract.

4 Résultats des tests

4.1 Préparation de l'environnement de travail

Déclaration des bibliothèques utilisées

```
fromimutils.object_detectionimportnon_max_suppression
import numpyasnp
import pytesseract
import argparse
import cv2
pytesseract.pytesseract.tesseract_cmd = r"C:\Program Files\Tesseract-
OCR\tesseract.exe"
```

4.2 Détection de texte

La fonction sorties_model est définit et permet d'extraire :

Les coordonnées du cadre de sélection d'une région de texte et la probabilité d'une détection de région de texte.

```
defsorties_model(probabilités, geometrie):


(lignes, colonnes) = probabilités.shape[2:4]
rec = []
confid = []
```

Saisir les dimensions du score map puis initialiser deux listes :

rec : stocke les coordonnées du cadre de sélection (x, y) pour les régions de texte.

confid: stocke la probabilité associée à chacun des cadres de sélection de la variable rec.

```
for y in range(0, lignes):
probabilitésD = probabilités[0, 0, y]
xD0 = geometrie[0, 0, y]
xD1 = geometrie[0, 1, y]
xD2 = geometrie[0, 2, y]
xD3 = geometrie[0, 3, y]
anglesD = geometrie[0, 4, y]
```

Faire une boucle par rapport à une seule ligne y (pour extraire les cartes de score et de la géométrie du rectangle (en anglais scores map, geometry map).

L'extraction de la carte de score contient les valeurs de la probabilité de l'existence du texte.

L'extraction des cartes de géométrie (d'une ligne y) des rectangles qui représentes 4 distances entre l'emplacement de pixel et les limites supérieures, droite, inférieure et gauche du rectangle respectivement et l'angle de rotation du rectangle.

```
forxin range(0, colonnes):
if probabilitésD[x] <args["minconfid"]:
continue
```

Faire une boucle (imbriquée) par rapport à x pour fixer une ligne y et boucler par rapport à toutes les colonnes de cette ligne et garder seulement les scores qui sont supérieurs à un certain seuil.

```
(corX, corY) = (x * 4.0, y * 4.0)
h = xD0[x] + xD2[x]
l = xD1[x] + xD3[x]
```

Multiplier les coordonnées par 4 comme la taille de la sortie du model est divisée par 4.

Calculer la hauteur du rectangle où xD0 est la distance entre un pixel et le haut du rectangle, et xD2 est la distance entre le même pixel et le bas du rectangle, la somme des deux donnela hauteur.

Calculer la largeur du rectangle où xD1 est la distance entre un pixel et le côté droit du rectangle, et xD2 est la distance entre le même pixel et le côté gauche du rectangle, la somme des deux donne la largeur.

```
finX = corX + xD1[x]
finY = corY + xD2[x]
debutX = int(finX - l)
debutY = int(finY - h)
```

Trouver les coordonnées du sommet droit du bas (bottom right), et du sommet gauche du haut (upperleft).

```
rec.append((debutX, debutY, finX, finY))
confid.append(probabilitésD[x])
return (rec, confid)
```

Ajout des coordonnées de début et de fin du rectangle et des scores qui dépasse le seuil (les bonnes probabilités). En sortant de la boucle principale, la fonction sorties_model les retourne.

L'utilisation d'argparse nous donne la possibilité d'entrer des arguments à partir de l'invité de cammande.

```
argpar = argparse.ArgumentParser()
```

Créer un objet argpar.

```
argpar.add_argument("-i", "--img", type=str,
help="chemin de l'image d'entrée ")
argpar.add_argument("-east", "--east", type=str,
help="chemin pour introduire le model")
argpar.add_argument("-c", "--minconfid", type=float, default=0.5,
help="probabilitée minimale")
argpar.add_argument("-l", "--largeur", type=int, default=320,
help="plus proche multiple de 32")
argpar.add_argument("-h", "--hauteur", type=int, default=320,
help=" plus proche multiple de 32 ")
argpar.add_argument("-p", "--pad", type=float, default=0.0,
help="ajouter le padding pour une meilleure détection")
ARPR = vars(argpar.parse_args())
```

ARPR contient les valeurs des arguments saisis dans l'invité de commande.

```
img = cv2.imread(ARPR["img"])
original = img.copy()
(originalH, originalL) = img.shape[:2]
(nouvL, nouvH) = (ARPR["largeur"], ARPR["hauteur"])
```

L'image est chargée en mémoire et copiée

La largeur et la hauteur d'origine sont extraites et les nouvelles largeurs et hauteur sont saisies dans ARPR.

```
RL = originalL / float(nouvL)
RH = originalH / float(nouvH)
img = cv2.resize(img, (nouvL, nouvH))
(H, L) = img.shape[:2]
```

Calcul du rapport entre les dimensions d'origine et les nouvelles dimensions.

L'image est ensuite redimensionnée pour la passer dans le model.

```
model = cv2.dnn.readNet(ARPR["east"])
```

Chargement du modèle.

```
(probabilités, geometrie) = model.forward(model)
(rec, confid) = sorties_model(probabilités, geometrie)
```

Les probabilités et géométries sont générés par le modèle, on les introduit dans la fonction sorties_model pour extraire les bonnes probabilités es les rectangles associés.

```
rectangles = non_max_suppression(np.array(rec), probs=confid)
```

Plusieurs rectangles englobant une zone de texte sont générés, d'où l'utilité de l'algorithme NMS pour prendre les régions de texte les plus probables, en éliminant les autres régions qui se chevauchent.

```
résultats = []
```

Cette variable contient les derniers résultats.

```
for(debutX, debutY, finX, finY) in rectangles:
debutX = int(debutX * RL)
debutY = int(debutY * RH)
finX = int(finX * RL)
finY = int(finY * RH)
```

Ensuite, il faut boucler par rapport à la variable rectangle qui contient les coordonnées des rectangles où On calcule des nouvelles coordonnées en fonction des ratios précédemment calculés.

```
région = original[debutY:finY, debutX:finX]
```

Les régions d'intérêt sont alors extraites.

4.3 Reconnaissance de texte

```
configuration = ("-l eng --oem 1 --psm 7")
```

Configuration de Tesseract comme il a été détaillé auparavant.

```
texte = pytesseract.image_to_string(région, config=configuration)
```

Nous appelons pytesseract.image_to_string, en passant les régions d'intérêt et la chaîne de configuration.

```
résultats.append(((debutX, debutY, finX, finY), texte))
```

On y ajoute les résultats.

```
résultats = sorted(résultats, key=lambda r: r[0][0])
```

Puis on ordonne les rectangles par rapport au sommet droit haut.

```
for ((debutX, debutY, finX, finY), text) in résultats:
texte = "".join([c if ord(c) <128 else "" for c in texte]).strip()
    sortie = original.copy()
    cv2.rectangle(sortie, (debutX, debuttY), (finX, finY),
(0, 0, 255), 2)
    cv2.putText(sortie, texte, (debutX, debutY -
20),cv2.FONT_HERSHEY_SIMPLEX,1.2,(0,0,255),3)
```

De là, en bouclant par rapport à la variable résultats, on opère à la suppression des caractères non ASCII du texte car OpenCV ne prend pas en charge les caractères non ASCII dans la fonction cv2.putText.

Affichage des résultats et l'enregistrement dans un fichier .txt

```
text_file = open("ocr.txt", "+a")
text_file.write("%s" % texte)
text_file.close()
    cv2.imshow("résultat", sortie)
    cv2.waitKey(0)
```

Ouvrir un fichier ocr.txt dans le répertoire de notre projet et écrire le résultat obtenu.

Dessiner un rectangle entourant la région d'intérêt et l'affichage de sortie.

Les figures 3.9, 3.10 et 3.11 représentent respectivement :

La procédure de détection.

Le traitement après détection par la fonction « sortie-model ».

L'extraction des régions d'intérêt, reconnaissance et résultats.

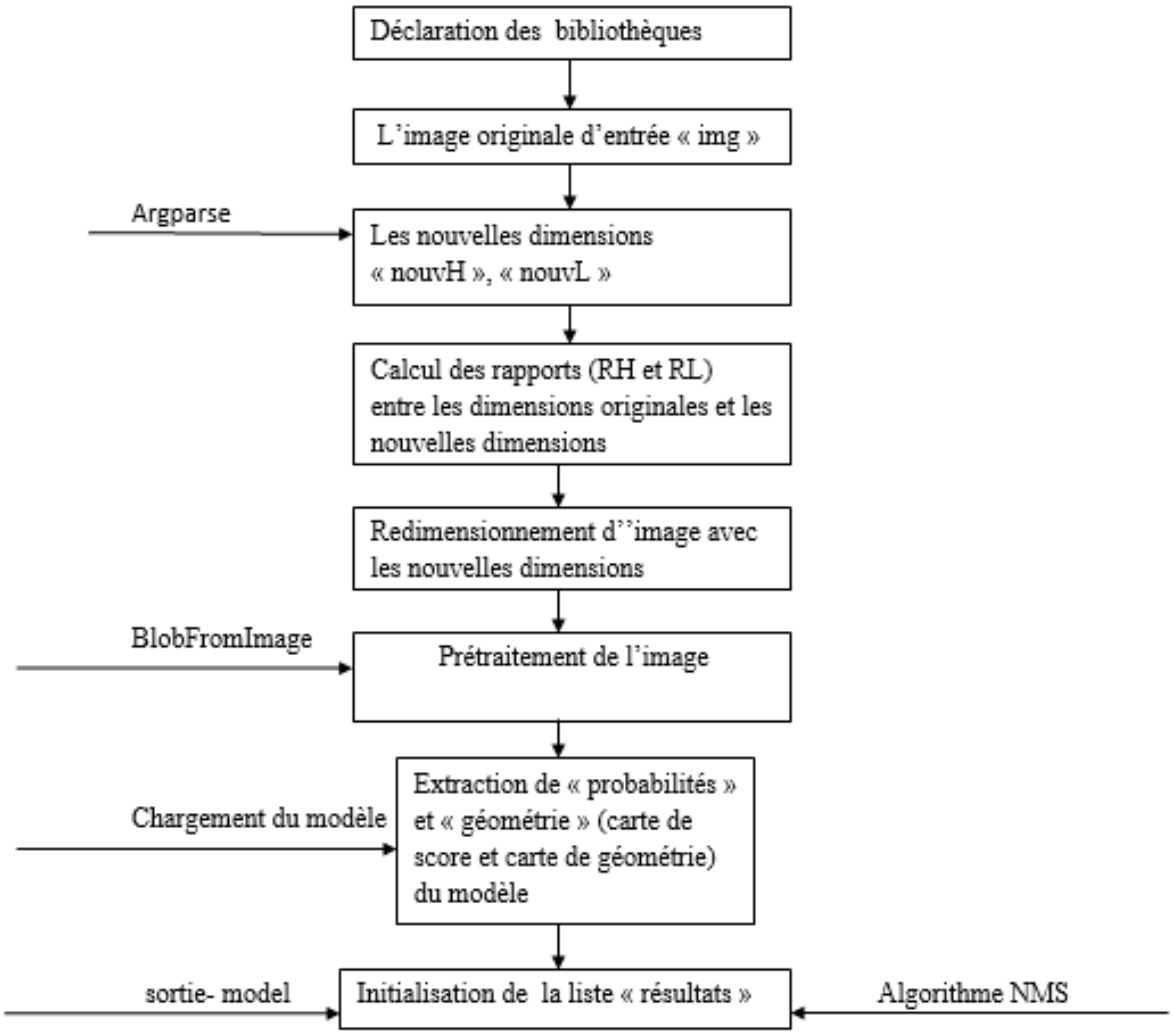


Figure 3.9: Procédure de détection.

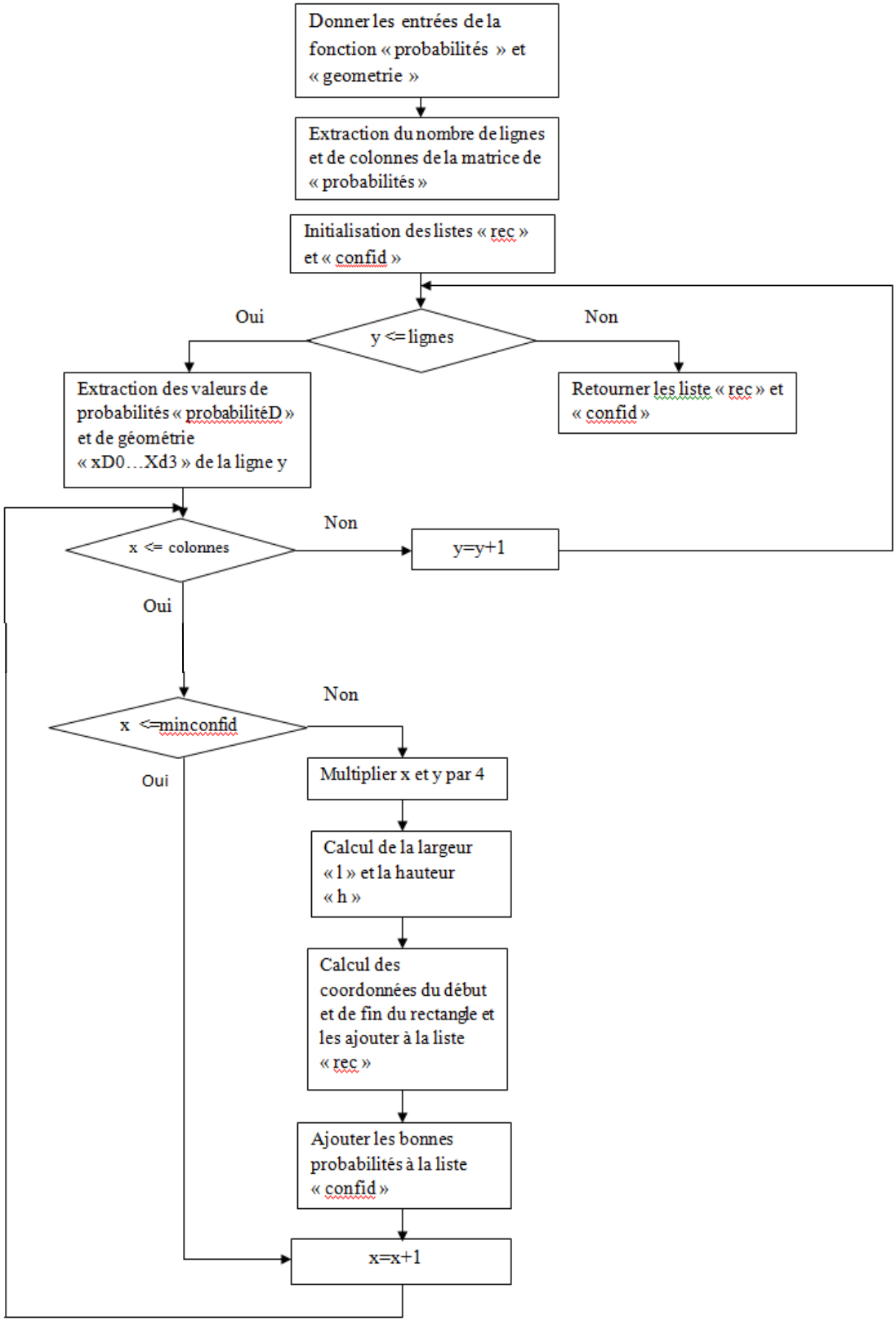


Figure 3.10 : Le traitement après détection par la fonction « sortie-model ».

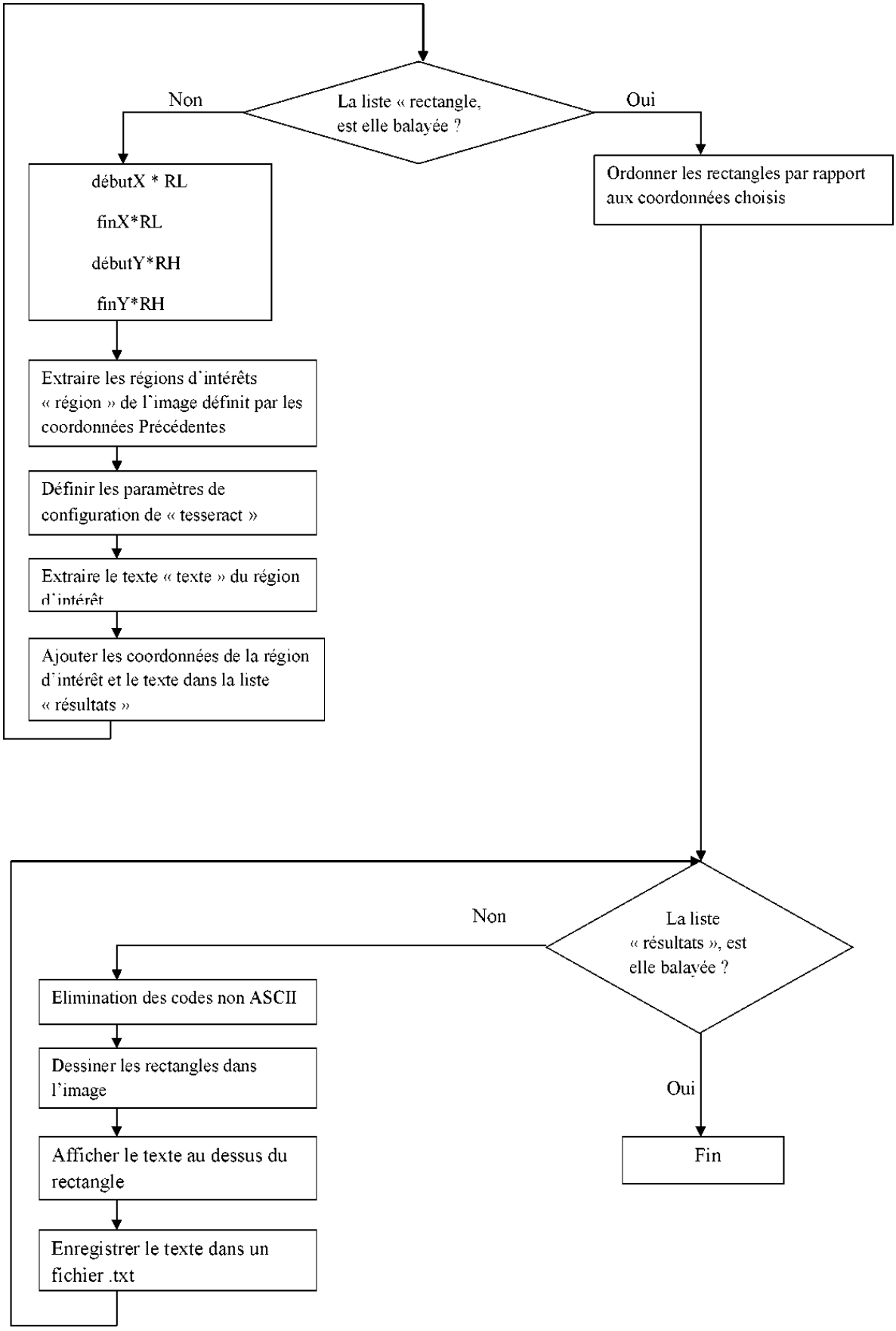


Figure 3.11 : Extraction des régions d'intérêt, reconnaissance du texte et résultats.

4.4 Résultats

Les figures 3.9, figure 3.10, figure 3.11, figure 3.12 et figure 3.13 montrent les résultats de détection et de reconnaissance de notre système.

Figure 3.12 : Une image d'un panneau de signalisation « rappel 90 » détecté et identifié avec la sortie enregistrée dans le fichier .txt.

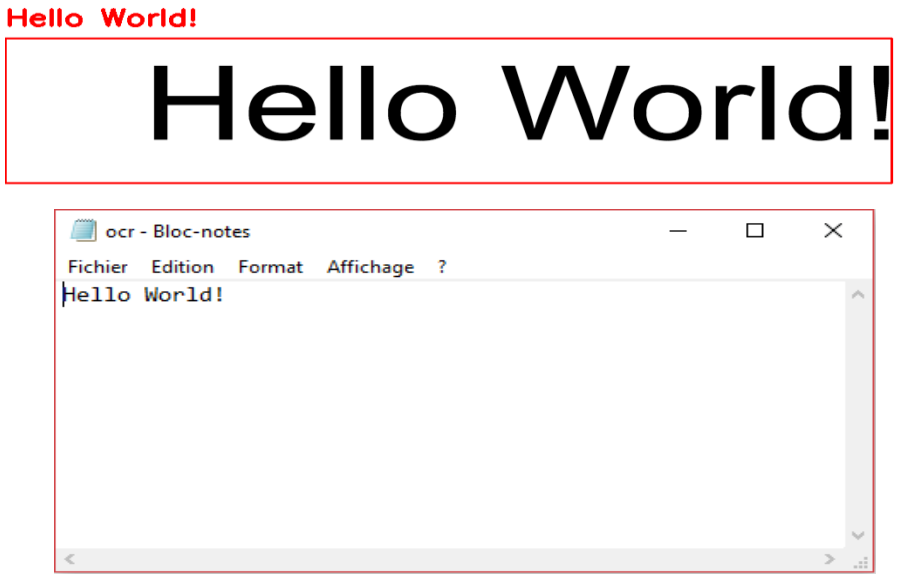


Figure 3.13: Résultat de détection et reconnaissance d'une phrase avec notre système.

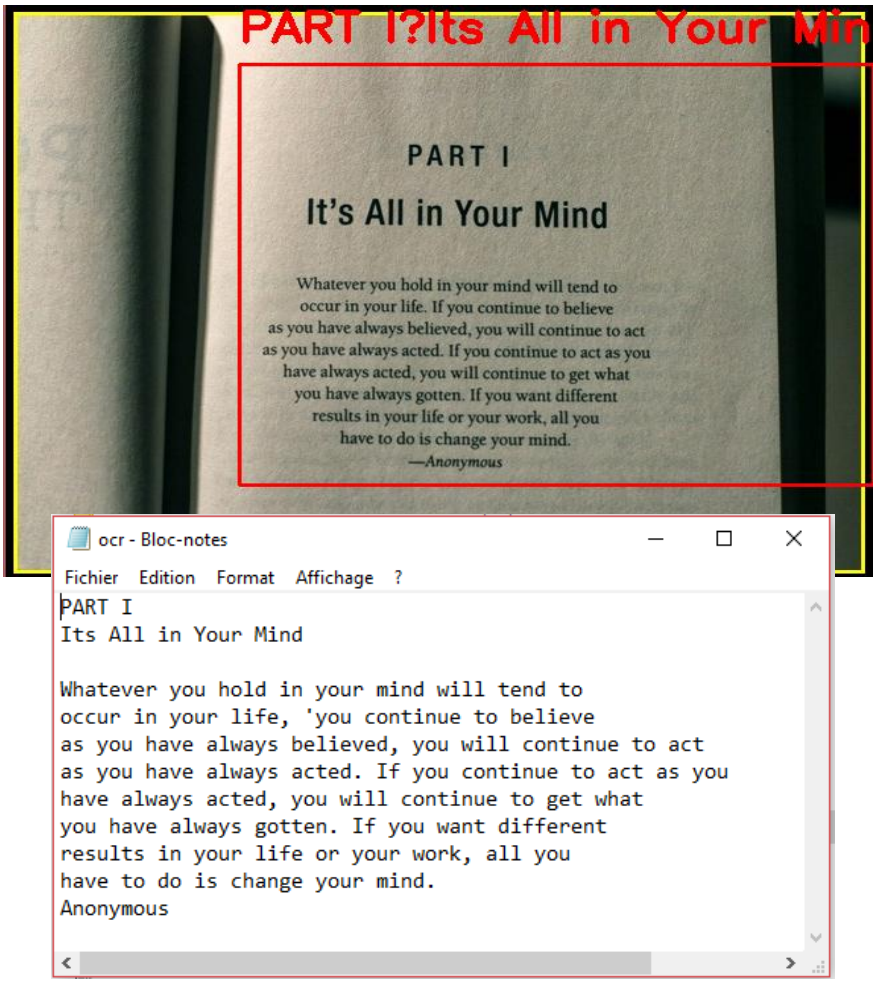


Figure 3.14 : Résultat de détection et reconnaissance d'un paragraphe avec notre système.

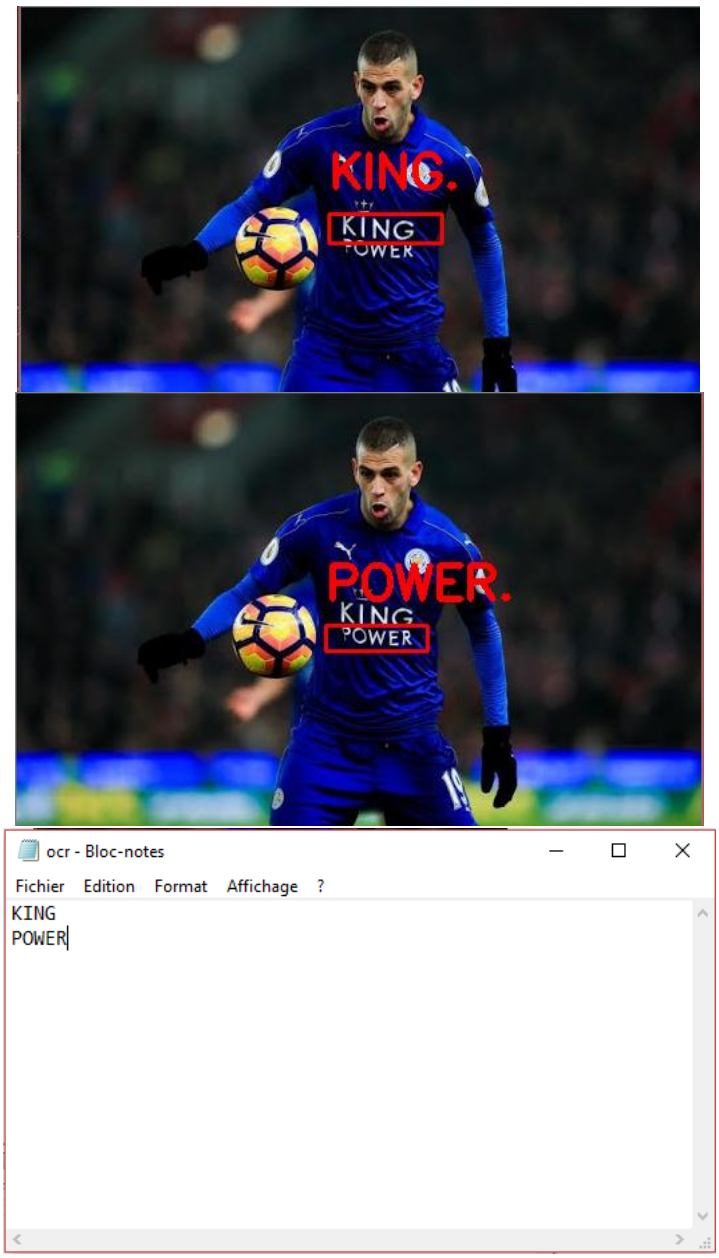


Figure 3.15: Résultat de détection et reconnaissance d'une image de scène naturelle avec notre système.


CORCAIGH
CORCAIGH
CORK
6
AN LONGFORT
LONGFORD
5
CORK
CORCAIGH
CORK
6
AN LONGFORT
LONGFORD
5
CORCAIGH
CORK
6
AN
AN LONGFORT
LONGFORD
5
CORCAIGH
CORK
6
LONGFORT
AN LONGFORT
LONGFORD
5

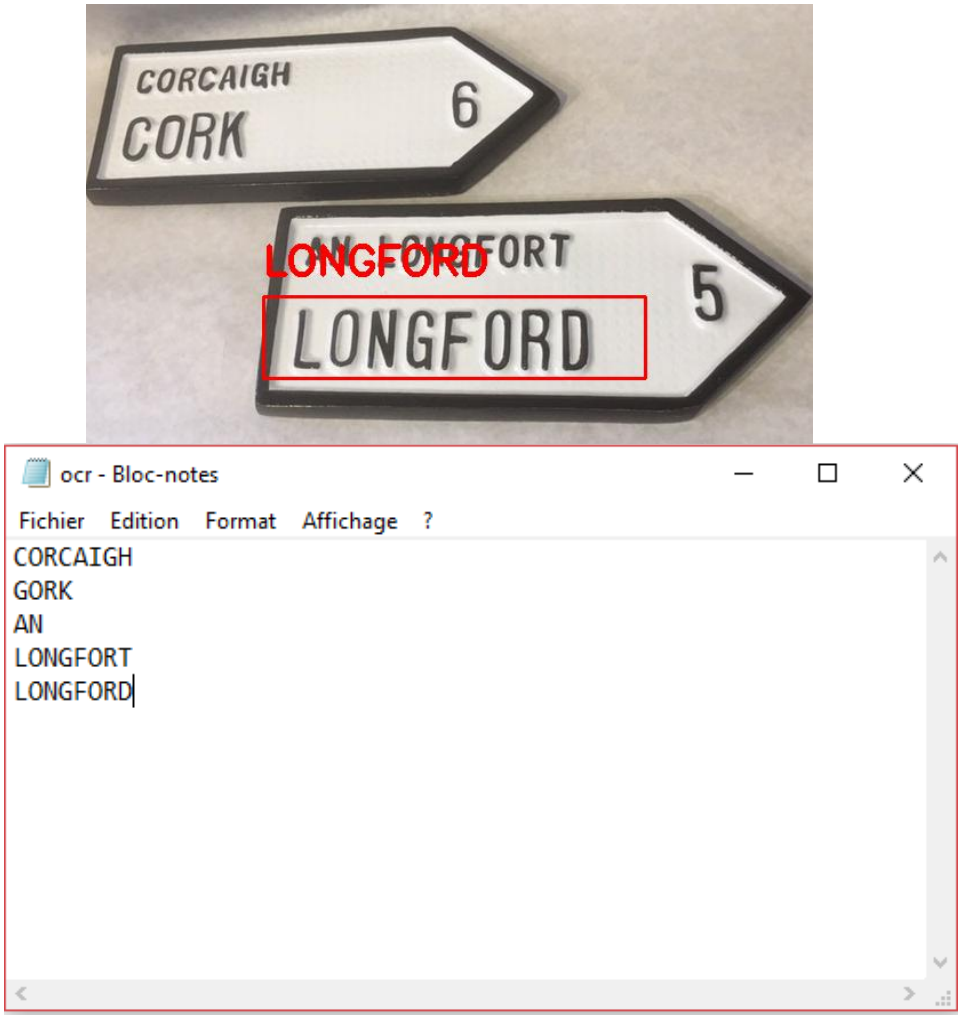


Figure 3.16 : Résultat de détection et reconnaissance de plusieurs panneaux de signalisation avec notre système.

4.5 Comparaison avec un système OCR en ligne

Pour la comparaison on a utilisé un système OCR en ligne appelé « FREE OCR ONLINE » disponible sur [Web32].

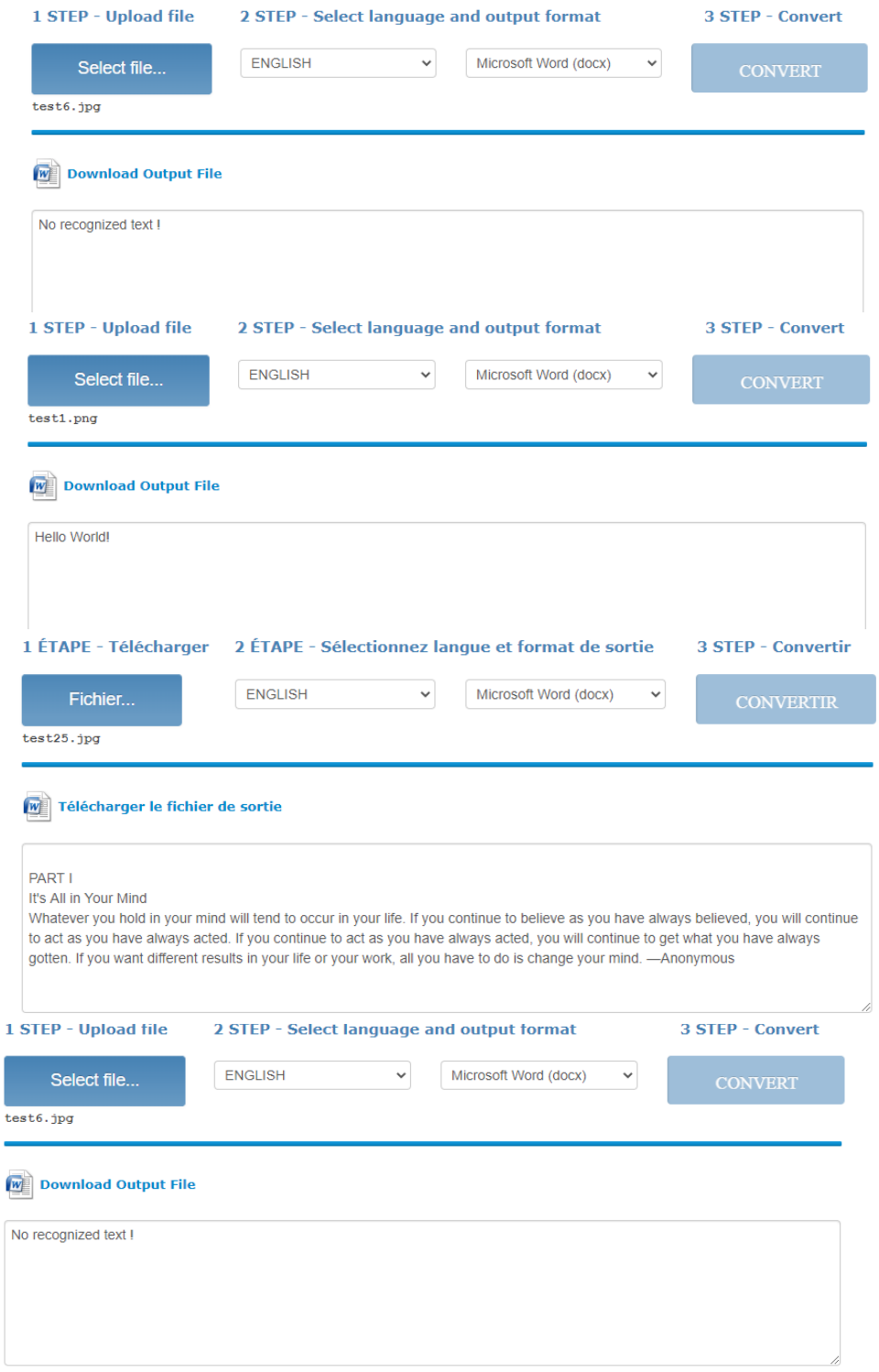


Figure 3.17 : Résultats des tests précédents respectifs avec OCR Online.

Résultats de comparaison :

- Notre système a fonctionné aussi bien que le système en ligne et a même dépassé les performances du système en ligne dans la détection et la reconnaissance d'images de scènes naturelles.

- Le seul inconvénient est que nous devons réinitialiser toutes les configurations en fonction de chaque document (une image d'une scène naturelle nécessite une configuration du système différente de celle d'une image d'un document PDF), mais le système en ligne est plus automatisé.

Limitations de ce système

Le système OCR proposé peut bien traiter certaines images, mais ne parvient pas à en traiter d'autres. Il y a des raisons principales à l'échec de ce processus de reconnaissance de texte:

- Le texte est tordu ou pivoté. Les polices de texte sont loin de celles entrainé au modèle Tesseract. Même si Tesseract V4 est plus puissant et précis que la v3, le modèle d'apprentissage profond est toujours limité par les données d'entraînement. Si votre police de texte est trop éloignée de la police de données d'entraînement, Tesseract peut ne pas être en mesure de reconnaitre le texte.
- Tesseract suppose toujours que l'image d'entrée a été correctement nettoyée. Lorsque nous effectuons la reconnaissance de texte sur des images de scènes naturelles, cette hypothèse n'est pas toujours exacte.
- Il ne fonctionne pas bien avec les images affectées par des artefacts, y compris l'occlusion partielle, la perspective déformée et l'arrière-plan complexe.
- Il n'est pas capable de reconnaître l'écriture manuscrite.
- Si un document contient des autres langues que celles indiquées dans les arguments lang, les résultats peuvent être médiocres. Cela signifie que vous devez réinitialiser les paramètres en fonction de chaque document.
- Il n'est pas capable d'analyser l'ordre de lecture naturel des documents. Par exemple, il peut ne pas reconnaître qu'un document contient deux colonnes et peut essayer de joindre du texte sur plusieurs colonnes.

5 Conclusion :

On a réussi à implémenter un système de détection et de reconnaissance de texte en utilisant lemodèle EAST pour la détection et LSTM Tesseract v4 pour la reconnaissance Nous avons comparé les performances de notre système à un système OCR en ligne et nous avons fait changer les paramètres un à un afin de comprendre l'impact de chacun sur la performance.

Conclusion générale

Dans ce travail, nous avons exploré le domaine d'OCR qui, comme tous les autres domaines du traitement d'images, a connu une évolution majeure avec l'avancement de l'intelligence artificielle et plus particulièrement du deep learning. La détection et la reconnaissance du texte dans des images de scènes naturelles sont alors devenues plus précises et plus efficaces.

Nous avons essayé de comprendre les concepts de ce domaine et de trouver les meilleurs algorithmes qui atteignent l'état de l'art dans la détection et la reconnaissance du texte.

Nous avons présenté un certain nombre de méthodes, de définitions et d'algorithmes concernant le domaine d'OCR dont certains d'entre eux ont été réutilisés pour réaliser notre travail. .

Afin d'aboutir aux résultats voulus :

- Nous avons implémenté le modèle EAST de détection de texte dans des images de scènes naturelles et de l'optimiser afin d'avoir une meilleure détection.
- Nous avons testé le moteur d'OCR tesseract et exploré ses paramètres afin de le combiner avec le modèle de détection EAST
- Une fois notre système réalisé, on a pris beaucoup de temps dans l'optimisation et le réglage des paramètres afin d'obtenir de bons résultats.

Ce travail nous a permis de faire le premier pas vers l'apprentissage profond, un des champs les plus importants de l'intelligence artificielle, d'ailleurs certains le considèrent comme étant le seul digne de faire partie de l'IA.

Le potentiel de ce travail est donc son pouvoir d'être embarqué dans un système comme android pour le rendre plus accessible dans le monde professionnel.

Bibliographie

[1] Horacio Cachinho. « Reconnaissance d'éléments textuels ou graphiques dans les documents juridiques numérisés».Projet Recherche & Développement 2016-2017. Ecole Polytechnique de l'Université François Rabelais de Tours, Département Informatique. 2 Avril 2017 .

[2]Dr. Chang Shu. « Image Formation ». COMP 4900C Winter 2008.

[3] M.A.Balafar. "REVIEW OF NOISE REDUCING ALGORITHMS FOR BRAIN MRI IMAGES". International Journal on "Technical and Physical Problems of Engineering" (IJTPE) Published by International Organization of IOTPE.December 2012, Issue 13,Volume 4, Number 4,Pages 54-59.

[4] Kun-yu, Tien ,HoomanSamani, JUI HUA, LUI . "A Survey on Image Processing in Noisy Environment by Fuzzy Logic, Image Fusion, Neural Network, and Non-Local Means".

[5]SukumarMaji. "Image skew detection and correction in regular images and document images". Department of Computer Science and Engineering National Institute of Technology Rourkela Rourkela-769 008, Odisha, India. February' 2015.

[6] WIESŁAW CHMIELNICKI1, KATARZYNA STĄPOR. "Investigation of Normalization Techniques and Their Impact on a Recognition Rate in Handwritten Numeral Recognition".Faculty of Physics, Astronomy and Applied Computer Science, Jagiellonian University, Reymonta 4, 30-059 Kraków,Institute of Computer Science, Silesian Technical University,VOLUME19. 2010.

[7] Xiangjian He, Lihong Zheng, Qiang Wu, WenjingJia, BijanSamali and MarimuthuPalaniswami. "Segmentation of Characters on Car License Plates". Faculty of Engineering and Information Technology University of Technology, Sydney PO Box 123, Broadway NSW 2007, Australia. Charles Sturt University WaggaWagga Campus NSW, Australia. Department of Electrical and Electronic Engineering The University of Melbourne. Victoria, 3010, Australia.

[8] DeepikaGhai, Neelu Jain. "Text Extraction from Document Images- A Review". E & EC Deptt. PEC University of Technology Chandigarh, India. E & EC Deptt. PEC University of Technology Chandigarh, India. International Journal of Computer Applications (0975 – 8887) Volume 84 – No 3, December 2013.

[9] Olivier Augereau. "Reconnaissance et classification d'images de documents". 2013.

[10] Stefano Ferilli, Fabio Leuzzi, Fulvio Rotella etFloriana Esposito. A run length smoothing-based algorithm for non-manhattan document segmentation.

[11] David G. Lowe. Distinctive Image Features from Scale-Invariant Keypoints. 2004.

[12]Andreas Kuhn, True Price , Jan-Michael Frahm , Helmut Mayer.” Down to Earth: Using Semantics for Robust Hypothesis Selection for the Five-Point Algorithm”.Bundeswehr University Munich. The University of North Carolina at Chapel Hill.

[13] Herbert Bay, Andreas Ess, TinneTuytelaarset Luc Van Gool. Speeded-Up RobustFeatures (SURF). 2008.

[14] Chokri FERKOUS. « Extraction des connaissances pour la segmentation d'images mammographique ». THESE Présentée en vue de l'obtention du diplôme de Doctorat en Sciences. UNIVERSITY BADJI MOKHTAR-ANNABA. Faculté des Sciences de l'Ingéniorat. Département d'Informatique. Année : 2016/2017.

[15] Mohamadally Hasan et Fomani Boris. SVM : Machines à Vecteurs de Support ou Séparateurs à Vastes Marges. 2006.

[16] ZafeiriosFountas. “Spiking Neural Networks for Human-like Avatar Control in a Simulated Environment”. Submitted in partial fulfilment of the requirements for the MSc Degree inComputing Science of Imperial College London. Septembre 2011.

[17] Masakazu Matsugu, Katsuhiko Mori, Yusuke Mitari, Yuji Kaneda.” Subject independent facial expression recognition with robust face detection using a convolutional neural network”. Canon Research Center, 5-1, Morinosato-Wakamiya, Atsugi 243-0193, Japan. Neural Networks 16 (2003) 555–559. 2003 Special issue.

[18] Zhong-Qiu Zhao, Member, IEEE, Peng Zheng, Shou-tao Xu, and Xindong Wu, Fellow, IEEE. “Object Detection with Deep Learning: A Review”. 16 Apr 2019.

[19] Kalinovskii I.A, Spitsyn V.G. Tomsk Polytechnic University Tomsk, Russia. “Compact Convolutional Neural Network Cascade for Face Detection”.

[20] Tao Wang, David J. Wu, Adam Coates, Andrew Y. Ng. “End-to-End Text Recognition with Convolutional Neural Networks”. Stanford University, 353 Serra Mall, Stanford, CA 94305.

[21] Wei Bao , Jun Yue, Yulei Rao. “A deep learning framework for financial time series using stacked autoencoders and longshort term memory”. Business School, Central South.
University, Changsha, China, Institute of Remote Sensing andGeographic Information System, Peking University, Beijing, China.2017.

[22] Deep Learning. Ian Goodfellow, YoshuaBengio, Aaron Courville. MIT Press. 2016.

[23] RiccoRakotomalala. « Principe de la descente de gradient pour l'apprentissage supervisé Application à la régression linéaire et la régression logistique ». Université Lumière Lyon 2.

[24] PrajitRamachandran ,BarretZoph, Quoc V. Le. Google Brain.« SEARCHING FOR ACTIVATION FUNCTIONS”.27 Oct 2017.

[25] Shangbang Long, Xin He, Cong Yao. “Scene Text Detection and Recognition: The Deep Learning Era”. 5 Sep 2019.

[26] Ross Girshick, Jeff Donahue, Trevor Darrell,Jitendra Malik. UC Berkele." Rich feature hierarchies for accurate object detection and semantic segmentation Tech report (v5)". 22 Oct 2014.

[27] Ross Girshick, Microsoft Research. "Fast R-CNN". 27 Sept 2015.

[28] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks". 6 Jan 2016.

[29] Yingying Jiang, Xiangyu Zhu, Xiaobing Wang, Shuli Yang, Wei Li, Hua Wang, Pei Fu and Zhenbo Luo. "R 2CNN: Rotational Region CNN for Orientation Robust Scene Text Detection". Samsung R&D Institute China - Beijing. 30 June 2017.

[30] Wei Liu ,DragomirAnguelov , DumitruErhan , Christian Szegedy , Scott Reed, Cheng-Yang Fu , Alexander C. Berg . "SSD: Single Shot MultiBoxDetector"UNC Chapel Hill Zoox Inc. Google Inc, University of Michigan, Ann-Arbor. 29 Dec 2016.

[31] Joseph Redmon , Santosh Divvala, Ross Girshick , Ali Farhadi. "You Only Look Once: Unified, Real-Time Object Detection". University of Washington, Allen Institute for AI, Facebook AI Research.9 May 2016.

[32] Alex Graves, Santiago Fernandez, Marcus Liwicki, Horst Bunke, JurgenSchmidhuber. "Unconstrained Online Handwriting Recognition with Recurrent Neural Networks". TUM, Germany. IDSIA, Switzerland. University of Bern, Switzerland. University of Bern, Switzerland. IDSIA, Switzerland and TUM, Germany.

[33] DzmitryBahdanau, KyungHyunCho ,YoshuaBengio. "NEURAL MACHINE TRANSLATION BY JOINTLY LEARNING TO ALIGN AND TRANSLATE". Jacobs University Bremen, Germany. Universite de Montréal. 19 May2016.

[34] Sepp Hochreiter, J□urgenSchmidhuber. "LONG SHORT-TERM MEMORY". Fakult□atf□urInformatikTechnischeUniversit□atM□unchen 80290 M□unchen, Germany, IDSIA Corso Elvezia 36 6900 Lugano, Switzerland. Neural Computation 9(8):1735{1780, 1997.

[35] Baoguang Shi, Xiang Bai and Cong Yao."An End-to-End Trainable Neural Network for Image-based Sequence Recognition and Its Application to Scene Text Recognition". School of Electronic Information and Communications, Huazhong University of Science and Technology, Wuhan, China. 21 Jul 2015.

[36] Max Jaderberg , Karen Simonyan, Andrew Zisserman, KorayKavukcuoglu. « Spatial Transformer Networks ». Google DeepMind, London, UK. 4 Feb 2016.

[37] Xinyu Zhou, Cong Yao, He Wen, Yuzhi Wang, Shuchang Zhou, Weiran He, and Jiajun Liang. ." EAST: An Efficient and Accurate Scene Text Detector". Megvii Technology Inc., Beijing, China.10 Jul 2017.

[38] Ray Smith, Google Inc. "An Overview of the Tesseract OCR Engine".

[39]] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," Proceedings of the IEEE, vol. 86, no. 11, pp. 2278–2324, 1998.

[40] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” in Advances in neural information processing systems, pp. 1097–1105, 2012.

[41] Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, “ImageNet Large Scale Visual Recognition Challenge,” International Journal of Computer Vision (IJCV), vol. 115, no. 3, pp. 211–252, 2015).

[42] M. D. Zeiler and R. Fergus, “Visualizing and understanding convolutional networks,” in European conference on computer vision, pp. 818–833, Springer, 2014.

[43] C.Szegedy,W.Liu,Y.Jia,P.Sermanet,S.Reed,D.Anguelov,D.Erhan,V.Vanhoucke,andA.Rabinovich , “Going deeper with convolutions,” in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 1–9, 2015.

[44] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. Alemi, “Inception-v4, inception-resnet and the impact of residual connections on learning,” arXiv preprint arXiv :1602.07261, 2016.

[45] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 770–778, 2016.

[46] Curtis P Langlotz, Bibb Allen, Bradley J. Erickson, JayashreeKalpathy-Cramer "A Roadmap for Foundational Research on Artificial Intelligence in Medical Imaging: From the 2018 NIH/RSNA/ACR/The Academy Workshop" Radiology 291(3):190613April 2019.

[47] Matthias Lee python-tesseract, OCR, Python.

Webographie

[Web1] https://www.abbyy.co.il/?categoryId=72050&itemId=168963

[Web2] https://brohrer.github.io/convert_rgb_to_grayscale.html.

[Web3]http://homepages.inf.ed.ac.uk/rbf/HIPR2/threshld.htm

[Web4] http://homepages.inf.ed.ac.uk/rbf/HIPR2/adpthrsh.htm

[Web5]https://www.slideshare.net/yaldaak/otsu-binarization

[Web6] https://www.mdpi.com/2076-3417/10/7/2236

[Web7] https://medium.com/@susmithreddyvedere/segmentation-in-ocr-10de176cf373,

[Web8] http://crblpocr.blogspot.com/2007/06/run-length-smoothing-algorithm-rlsa.html

[Web9] https://rocbo.lautre.net/typo/site/anatomie.htm

[Web10] https://sites.google.com/site/iprinceps/Home/augmented-reality-and-artoolkit/what-i-ve-done/object-recognition-with-sift-and-fast

[Web11] Olivier Augereau. SIFT et SURF. 2011. http://www.olivier- augereau. com/blog/?p=73

[Web12] http://cedric.cnam.fr/vertigo/Cours/ml2/coursSVMLineaires.html

[Web13] https://www.how-ocr-works.com/OCR/omnifont-OCR.html

Chapitreee 2

[Web14]https://www.bial-r.com/2019/05/22/comprendre-le-machine-learning-et-le-deep-learning/

[Web15] https://sciences-cognitives.fr/les-neurones-pour-apprendre/

[Web16] https://medium.com/towards-artificial-intelligence/deep-learning-series-chapter-1-introduction-to-deep-learning-d790feb974e2.

[Web17] https://mc.ai/how-to-collect-your-deep-learning-dataset/

[Web18] https://lawtomated.com/a-i-technical-machine-vs-deep-learning/

[Web19] https://mc.ai/deeplearning-overview-of-convolution-neural-network/

[Web20] https://ireneli.eu/2016/02/03/deep-learning-05-talk-about-convolutional-neural-network%EF%BC%88cnn%EF%BC%89/

[Web21] https://www.mdpi.com/2076-3417/9/21/4500

[Web22] https://jeanvitor.com/convolution-parallel-algorithm-python/

[Web23] https://onlinelibrary.wiley.com/doi/full/10.1002/cyto.a.23701.

[Web24] https://indoml.com/2018/03/07/student-notes-convolutional-neural-networks-cnn-introduction/

[Web25] https://towardsdatascience.com/beginners-guide-to-understanding-convolutional-neural-networks-ae9ed58bb17d

[Web26] https://missinglink.ai/guides/convolutional-neural-networks/convolutional-neural-network-tutorial-basic-advanced/

[Web27] https://acadgild.com/blog/recurrent-neural-network-tutorial

[Web28] http://colah.github.io/posts/2015-08-Understanding-LSTMs/

[Web29] https://www.oreilly.com/library/view/mastering-machine-learning/9781788621113/913c98cf-b765-4840-92e1-d873df659c6a.xhtml

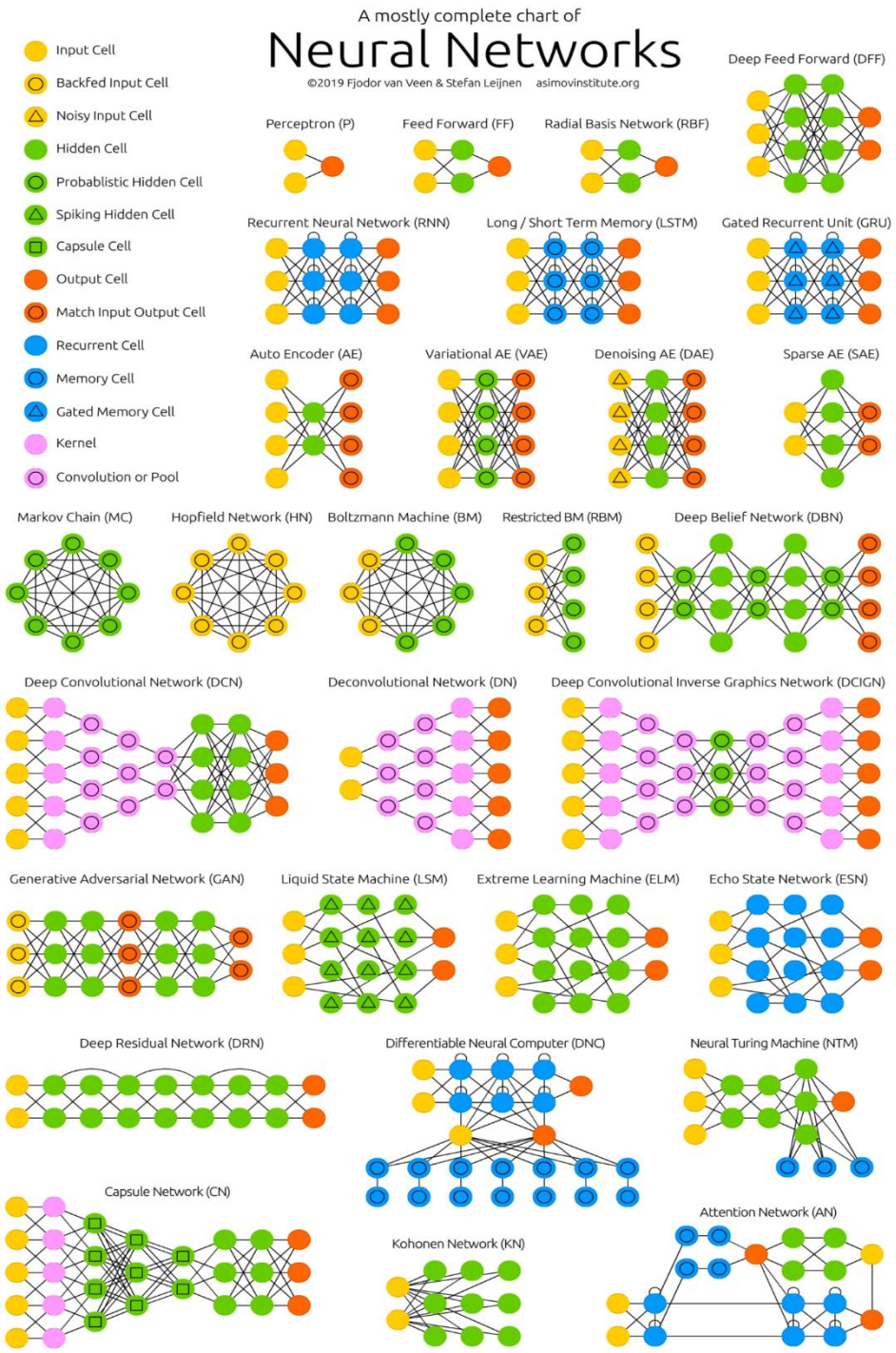
[Web30] https://www.slideshare.net/oscon2007/os-raysmith

[Web31] https://www.slideshare.net/microlife/ocr-processing-with-deep-learning-apply-to-vietnamese-documents

[Web32] https://www.onlineocr.net/

[Web33] https://www.asimovinstitute.org/neural-network-zoo/

Annexe A.



Un résumé des types d'architectures de réseaux de neurones [Web33]

Annexe B.

Quelques réseaux convolutifs célèbres

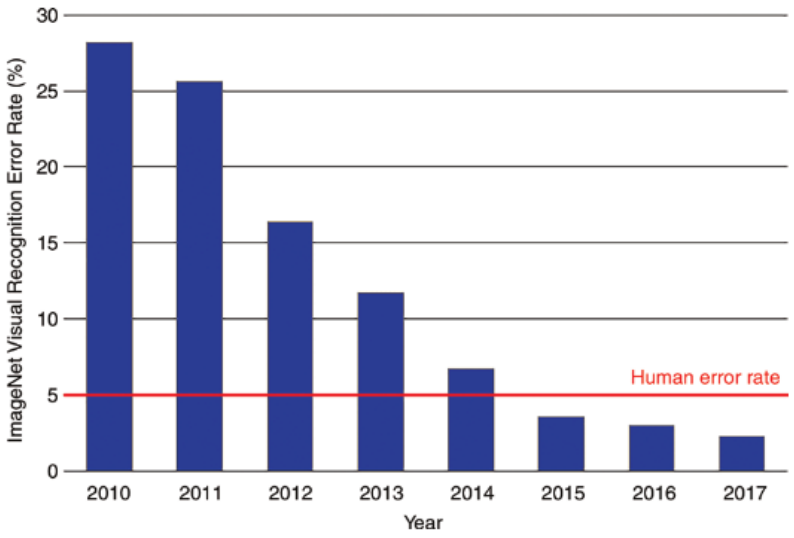
LeNet[39] : Les premières applications réussies des réseaux convolutifs ont été développées par Yann LeCun dans les années 1990. Parmi ceux-ci, le plus connu est l'architecture LeNet utilisée pour lire les codes postaux, les chiffres, etc.

AlexNet[40]: Le premier travail qui a popularisé les réseaux convolutifs dans la vision par ordinateur était AlexNet, développé par Alex Krizhevsky, Ilya Sutskever et Geoff Hinton. AlexNet a été soumis au défi ImageNet ILSVRC, [41] en 2012 et a nettement surpassé ses concurrents. Le réseau avait une architecture très similaire à LeNet, mais était plus profond, plus grand et comportait descouchesconvolutives empilées les unes sur les autres (auparavant, il était commun de ne disposer que d'une seule couche convolutifs toujours immédiatement suivie d'une couche de pooling).

ZFnet[42]: Le vainqueur de ILSVRC challenge 2013 était un réseau convolutif de Matthew Zeiler et Rob Fergus. Il est devenu ZFNet (abréviation de Zeiler et Fergus Net). C'était une amélioration de AlexNet en ajustant les hyper-paramètres de l'architecture, en particulier en élargissant la taille des couches convolutifs et en réduisant la taille du noyau sur la première couche.

GoogLeNet[43] : Le vainqueur de ILSVRC challenge 2014 était un réseau convolutif de Szegedy et al. De Google. Sa principale contribution a été le développement d'un module inception qui a considérablement réduit le nombre de paramètres dans le réseau (4M, par rapport à AlexNet avec 60M). En outre, ce module utilise le **global AVG pooling** au lieu du PMC à la fin des réseaux, ce qui élimine une grande quantité de paramètres. Il existe également plusieurs versions de GoogLeNet, parmi elles, Inception-v4[44]

ResNet[45] : Residual network développé par Kaiming He et al. A été le vainqueur de ILSVRC 2015. Il présente des sauts de connexion et une forte utilisation de la batch normalisation. Il utilise aussi le **global AVG pooling** au lieu du PMC à la fin.



Taux d'erreur dans ImageNet Visual recognition Challenge. Le Deep Learning dépasse la performance humaine [46]

Annexe C.

L’usage de pytesseract

```
fromPILimportImage

exceptImportError:

importImage

importpytesseract
```

Si vous n'avez pas d'exécutable tesseract dans votre PATH, incluez les éléments suivants:

```
pytesseract.pytesseract.tesseract_cmd=r'<full_path_to_your_tesseract_executable>'
```

Exemple tesseract_cmd = r'C: \ Program Files (x86) \ Tesseract-OCR \ tesseract '

Image simple à enchaîner

```
print(pytesseract.image_to_string(Image.open('test.png')))
```

Image de texte en français à chaîne de caractères

```
print(pytesseract.image_to_string(Image.open('test-european.jpg'),lang='fra'))
```

Afin de contourner les conversions d'images de pytesseract, utilisez simplement un chemin d'image relatif ou absolu

REMARQUE: dans ce cas, vous devez fournir des images prises en charge par tesseract ou tesseract renverra une erreur

```
print(pytesseract.image_to_string('test.png'))
```

Traitement par lots avec un seul fichier contenant la liste de plusieurs chemins de fichier image

```
print(pytesseract.image_to_string('images.txt'))
```

Timeout / terminer le travail tesseract après une période de temps

```
print(pytesseract.image_to_string('test.jpg',timeout=2))# Timeout after 2
seconds


print(pytesseract.image_to_string('test.jpg',timeout=0.5))# Timeout after
half a second


exceptRuntimeErrorastimeout_error:
```

Obtenez des estimations de bounding boxes

```
print(pytesseract.image_to_boxes(Image.open('test.png')))
```

Obtenez des données détaillées, y compris des rectangle, des confidences, des numéros de ligne et de page

```
print(pytesseract.image_to_data(Image.open('test.png')))
```

Obtenez des informations sur l'orientation et la détection des scripts

```
print(pytesseract.image_to_osd(Image.open('test.png')))
```

Obtenez un searchablePDF le type pdf est en byte par défaut

```
pdf=pytesseract.image_to_pdf_or_hocr('test.png',extension='pdf')

withopen('test.pdf','w+b')asf:

f.write(pdf)
```

Obtenir la sortie HOCR

```
hocr=pytesseract.image_to_pdf_or_hocr('test.png',extension='hocr')
```

Une explication encore plus détaillée ce trouve dans [47]