

MINISTÈRE DE L'ENSEIGNEMENT SUPÉRIEUR ET DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITÉ ABDELHAMID IBN BADIS - MOSTAGANEM



**Faculté des Sciences Exactes et d'Informatique**

**Département de Mathématiques et informatique**

**Filière : Informatique**

**RAPPORT DE MINI-PROJET**

Option : **Ingénierie des Systèmes d'Information**

**THÈME :**

**Reconnaissance De L'écriture manuscrits  
avec les réseaux de neurones**

Etudiants : « Mohamed el hadi Boukhatem »

« Iyad Makrerougrass »

Encadrant : « Reda Djebbara »

Année Universitaire 2021-2022

## **Résumé**

La numérisation massive de documents papier a fait apparaître le besoin d'avoir des systèmes de reconnaissance de l'écriture extrêmement performants. La numérisation de ces documents permet d'effectuer des opérations telles que des recherches ou l'extraction d'informations. La reconnaissance de l'écriture et en particulier l'écriture manuscrite ne sont pas encore au niveau de performance de l'homme sur des documents complexes, ce qui restreint ou nuit à certaines.

Notre travail compare différentes technologies et architectures permettant d'effectuer la reconnaissance de l'écriture manuscrite et fournir un système basé sur les réseaux de neurones artificiels précisément les réseaux de neurones récurrents (RNN).

### **Mots-clés:**

Apprentissage profond, reconnaissance de l'écriture, réseaux de neurones, RNN, CNN.

## **Abstract**

The massive digitization of paper documents has revealed the need for extremely efficient handwriting recognition systems. The digitization of these documents makes it possible to carry out operations such as research or the extraction of information. handwriting recognition and in particular handwriting are not yet at the level of human performance on complex documents, which restricts or impairs some.

Our work compares different technologies and architectures to perform handwriting recognition and provide a system based on artificial neural networks, precisely recurrent neural networks (RNN).

### **Keywords:**

Deep learning, handwriting recognition, neural networks, RNN, CNN.

## Dédicaces

*C'est avec grand respect et gratitude que je tiens à exprimer toute ma reconnaissance et ma sympathie et dédier ce travail modeste à mes parents MOKHTAR, NACERA à mon grand-père KADDOUR et FOUZIA , les mots ne sauraient exprimer l'immense et profonde gratitude que je leur témoigne ici pour leurs précieux soutien, pour leurs patience, pour avoir crus en moi, pour leurs sourires réconfortants et pour leurs sacrifices qui m'ont permis d'atteindre cette étape dans ma vie et qu'ils n'ont jamais cessé de consentir pour mon instruction et mon bien être dieu me les gardes et les protèges.*

- *Mes frères et sœurs, Mes oncles KACEM et MANSOUR et mon cousin SEDDIK.*
  - *Dédicace à mon binôme IYAD et à mon amie YASMINE*
  - *Tous mes chers amis, mes meilleurs amis IDIR, RAMZI, GHILES, HENDI, IBRAHIM, FARID ..*
- *Toute ma famille et surtout mon Grand-père ABDELLAH et ma grand-mère KHAIRA décédés.*
- *Tous ceux qui ont participé de près ou de loin à la réalisation de ce travail.*
  - *Tous mes enseignants tout au long des cycles de mes études.*
  - *Toute la promotion 2021/2022 de ISI.*

**BOUKHATEM Mohamed**

*C'est avec grand respect et gratitude que je tiens à exprimer toute ma reconnaissance et ma sympathie et dédier ce travail modeste à :*

- *À ma maman qui m'a soutenu et encouragé durant ces années d'études, qu'elle trouve ici le témoignage de ma profonde reconnaissance et à mon père ALLAH Yarhmou qui aurait été si fier de ce que son fils a accompli.*

- *À mes deux grands frères RAMY et WANISS et ma grand-mère que dieu la protège.*

*Grâce à leur présence, soutien moral permanent et leur amour infini tout au long de ma vie que j'ai pu accomplir ce travail qui est le fruit de leurs sacrifices qu'ils ont consentis pour mon éducation et ma formation.*

- *À tous mes amis qui m'ont toujours encouragé, surtout mon binôme avec lequel j'ai partagé des moments inoubliables, et à qui je souhaite plus de succès et de réussite dans leur vie.*

- *À mon binôme MOHAMED BOUKHATEM pour sa bonne collaboration et son dévouement.*

**MAKREROUGRASS Iyad**

## Remerciements

*Nous remercions ALLAH de nous avoir donné la force et la capacité qui nous ont menées à ce niveau.*

*Nous tenons à exprimer notre profonde gratitude et reconnaissance à Mr. MOUMEN et Mr. ZAHMANI, nous vous remercions d'avoir bien assuré la direction et l'encadrement de notre travail malgré vos nombreuses occupations, nous vous remercions pour vos qualités humaines, votre précieuse attention, implication et vos précieux conseils et orientations qui ont mené à l'aboutissement de ce travail.*

*Nous remercions aussi les membres de jury pour avoir bien voulu donner de leur temps pour lire et juger notre projet.*

*Merci aussi à tous nos amis, nos collègues, et à tous ceux qui nous ont aidé de près ou de loin. Nous leur exprimons notre profonde sympathie et leur souhaitons beaucoup de bien.*

*Nous n'oublierons pas non plus de remercier toutes les personnes que nous avons pu côtoyer pendant ces cinq ans à la faculté des sciences exactes et d'informatique pour leur soutien moral et amical.*

## Liste des figures

|  |    |
|--|----|
| FIGURE 1 - AI & ML .....   | 5  |
| FIGURE 2 - RÉSEAU DE NEURONES SIMPLE .....   | 7  |
| FIGURE 3 - FONCTION SIGMOÏDE .....   | 7  |
| FIGURE 4 - COMPOSANTS D'UN PERCEPTRON .....  | 8  |
| FIGURE 5 - RÉSEAU DE NEURONES MULTICOUCHES .....   | 9  |
| FIGURE 6 - RÉSEAU DE NEURONES RÉCURRENT .....  | 11 |
| FIGURE 7 - FONCTION MATHÉMATIQUE.....  | 11 |
| FIGURE 8 : L'ANNOTATION LORSQUE LES CARACTÈRES OCCUPENT UN OU PLUS D'UN PAS<br>DE TEMPS..... | 12 |
| FIGURE 9 : MATRICE DE SORTIE DU RÉSEAU DE NEURONES .....                                     | 13 |
| FIGURE 10 - RECONNAISSANCE D'ÉCRITURE MANUSCRITE .....                                       | 15 |
| FIGURE 11 - LA STRUCTURE DE CNN .....  | 19 |
| FIGURE 12 - LA STRUCTURE DE RNN ET BRNN .....  | 21 |
| FIGURE 13 - FLUX DE CALCUL DU MODÈLE BRNN .....  | 22 |
| FIGURE 14 - LA STRUCTURE DE LSTM .....   | 24 |
| FIGURE 15 - SCHÉMA DU MÉCANISME D'ATTENTION .....  | 26 |
| FIGURE 16 - MODÈLE LSTM AVEC ATTENTION.....  | 26 |
| FIGURE 17 - FLUX DE CALCUL DU MODÈLE BLSTM .....   | 28 |
| FIGURE 18 - DÉPENDANCES ET ORDRE DE CALCUL DU MDLSTM .....                                   | 30 |
| FIGURE 19 - LA STRUCTURE DE QRNN .....   | 32 |
| FIGURE 20 - L'ARCHITECTURE GÉNÉRALE DU MODÈLE.....   | 38 |
| FIGURE 21 - FORMULAIRE CONTIENT TEXTES SEGMENTÉS .....                                       | 39 |
| FIGURE 22 - FICHIER WORDS.TXT.....   | 40 |
| FIGURE 23 - AFFICHAGE DU TERMINAL PENDANT LE PRÉTRAITEMENT .....                             | 41 |
| FIGURE 24 - ÉCHANTILLONS AVANT LE REDIMENSIONNEMENT .....                                    | 42 |
| FIGURE 25 - ÉCHANTILLONS APRÈS LE REDIMENSIONNEMENT .....                                    | 42 |
| FIGURE 26 - RÉSUMÉ DU MODÈLE .....   | 44 |
| FIGURE 27 - PRÉDICTIONS AVEC LES DONNÉES DE TEST .....                                       | 46 |
| FIGURE 28 - PRÉDICTIONS AVEC NOUVELLES IMAGES .....  | 47 |

## Liste des tableaux

|  |    |
|--|----|
| TABLEAU 1 - COMPARAISON ENTRE LSTM ET ALSTM .....                            | 27 |
| TABLEAU 2 - LES RÉSULTATS DES ENTRAÎNEMENTS SUR IAM .....                    | 31 |
| TABLEAU 3 - RÉSULTAT SUR L'ANALYSE DE SENTIMENT D'UNE CRITIQUE DE FILM ..... | 33 |
| TABLEAU 4 - RÉSULTAT SUR UNE TRADUCTION D'ALLEMAND EN ANGLAIS .....          | 33 |

## Liste des abréviations

| ABRÉVIATION | EXPRESSION COMPLÈTE                      | PAGE |
|-------------|--|------|
| AI          | ARTIFICIAL INTELLIGENCE                  | 5    |
| RNN         | RECURRENT NEURAL NETWORK                 | 7    |
| CNN         | CONVOLUTIONAL NEURAL NETWORK             | 8    |
| ML          | MACHINE LARNING                          | 5    |
| CSV         | COMMA SEPARATED VALUES                   | 9    |
| BRNN        | BIDIRECTIONAL RECURRENT NEURAL NETWORK   | 14   |
| MLP         | MULTILAYER PERCEPTRON                    | 14   |
| TDNN        | TIME DELAY NEURAL NETWORK                | 14   |
| IDS         | INTRUSION DETECTION SYSTEM               | 17   |
| LSTM        | LONG SHORT TERM MEMORY                   | 17   |
| ALSTM       | ATTENTION LONG SHORT-TERM MEMORY         | 19   |
| BLSTM       | BIDIRECTIONAL LONG SHORT-TERM MEMORY     | 22   |
| MDLSTM      | MULTI-DIMENSIONAL LONG SHORT-TERM MEMORY | 24   |
| WER         | WORD ERROR RATE                          | 25   |
| CER         | CHARACTER ERROR RATE                     | 25   |
| IAM         | HANDWRITING DATABASE                     | 25   |
| QRNN        | QUASI-RECURRENT NEURAL NETWORKS          | 26   |
| CNTK        | MICROSOFT COGNITIVE TOOLKIT              | 30   |
| API         | APPLICATION PROGRAMMING INTERFACE        | 30   |
| CPU         | CENTRAL PROCESSING UNIT                  | 33   |
| GPU         | GRAPHICS PROCESSING UNIT                 | 33   |



# Table des matières

|  |    |
|--|----|
| INTRODUCTION GÉNÉRALE .....  | 2  |
| CHAPITRE 1 APPRENTISSAGE MACHINE, RÉSEAUX DE NEURONES & RECONNAISSANCE MANUSCRITE..... | 4  |
| 1.1    INTRODUCTION .....  | 4  |
| 1.2    APPRENTISSAGE MACHINE.....  | 4  |
| 1.2.1    RÉGRESSION .....  | 5  |
| 1.2.2    CLASSIFICATION .....  | 5  |
| 1.2.3    CLUSTERING.....   | 6  |
| 1.2.4    REGROUPEMENT APPRENTISSAGE PAR RENFORCEMENT .....                             | 6  |
| 1.2.5    TRAITEMENT DU LANGAGE NATUREL .....   | 6  |
| 1.3    RÉSEAUX DE NEURONES ARTIFICIELS .....   | 7  |
| 1.4    RÉSEAUX DE NEURONES CONVOLUTIFS (CNN) .....                                     | 10 |
| 1.5    RÉSEAUX DE NEURONES RÉCURRENTS (RNN) .....                                      | 10 |
| 1.6    CLASSIFICATION TEMPORELLE CONNEXIONNISTE (CTC) .....                            | 12 |
| 1.6.1    AVANTAGES DU CTC.....   | 12 |
| 1.6.2    MÉCANISME DU CTC.....   | 13 |
| 1.7    RECONNAISSANCE MANUSCRITE .....   | 14 |
| 1.7.1    HORS-LIGNE.....   | 14 |
| 1.7.2    ENLIGNE .....   | 15 |
| 1.7.3    UTILISATION DES RNN DANS LA RECONNAISSANCE MANUSCRITE OFFLINE.....            | 15 |
| 1.8    CONCLUSION.....   | 16 |
| CHAPITRE 2 RÉSEAUX DE NEURONES RÉCURRENTS & CONVOLUTIFS .....                          | 17 |
| 2.1    INTRODUCTION .....  | 17 |
| 2.2    RÉSEAUX DE NEURONES CONVOLUTIFS .....   | 17 |
| 2.2.1    COUCHE CONVOLUTIVE .....  | 18 |
| 2.2.2    COUCHE DE POOLING .....   | 18 |
| 2.2.3    COUCHE RELU.....  | 18 |
| 2.2.4    COUCHE FULLY CONNECTED .....  | 18 |
| 2.2.5    COUCHE LOSS .....   | 19 |
| 2.2.6    IMPORTANCE DE LA CONVOLUTION.....   | 19 |
| 2.3    RÉSEAUX DE NEURONES RÉCURRENTS .....  | 20 |
| 2.3.1    BRNN .....  | 20 |
| 2.3.2    LSTM.....   | 23 |
| 2.3.3    ALSTM.....  | 25 |

|                                 |   |    |
|---------------------------------|---|----|
| 2.3.4                           | BLSTM .....                                     | 28 |
| 2.3.5                           | MDLSTM .....                                    | 29 |
| 2.3.6                           | QRNN .....                                      | 32 |
| 2.4                             | CONCLUSION.....                                 | 34 |
| CHAPITRE 3 IMPLÉMENTATION ..... |   | 35 |
| 3.1                             | INTRODUCTION .....                              | 35 |
| 3.2                             | ENVIRONNEMENTS D'IMPLÉMENTATION .....           | 35 |
| 3.2.1                           | TENSORFLOW.....                                 | 35 |
| 3.2.2                           | KERAS .....                                     | 36 |
| 3.2.3                           | PANDAS.....                                     | 36 |
| 3.2.4                           | NUMPY .....                                     | 36 |
| 3.2.5                           | MATPLOTLIB .....                                | 37 |
| 3.2.6                           | PILLOW .....                                    | 37 |
| 3.2.7                           | JUPYTER NOTEBOOK .....                          | 37 |
| 3.3                             | ÉTAPES DE CONSTRUCTION GÉNÉRALES DU MODÈLE..... | 38 |
| 3.3.1                           | COLLECTER LES DONNÉES .....                     | 38 |
| 3.3.2                           | PRÉTRAITEMENT.....                              | 39 |
| 3.3.3                           | CONSTRUIRE LE MODÈLE .....                      | 42 |
| 3.3.4                           | COMPILER LE MODÈLE.....                         | 45 |
| 3.3.5                           | ENTRAINER LE MODÈLE .....                       | 45 |
| 3.3.6                           | ÉVALUER LE MODÈLE .....                         | 46 |
| 3.3.7                           | PRÉDICTION DU MODÈLE .....                      | 46 |
| 3.4                             | CONCLUSION.....                                 | 47 |
| CONCLUSION GÉNÉRALE.....        |   | 48 |
| BIBLIOGRAPHIE .....             |   | 50 |

# Introduction Générale

Pendant plusieurs millénaires, l'écriture joue un rôle important voire indispensable dans la vie de l'être humain, l'homme a commencé à écrire ces premiers mots sur des murs ou des rochers pour transmettre des idées ou pour des besoins de comptabilité c'est-à-dire compter les possessions et les biens manufacturés, ce qui lui a permis à se développer et de trouver plus rapidement de nouvelles technologies qui vont l'aider dans la vie de tous les jours pour nous retrouver aussi avancé dans notre temps.

A l'ère numérique, où tout le monde utilise des téléphones et des ordinateurs pour transmettre l'information, les chercheurs et scientifiques ont logiquement commencé à développer une nouvelle technologie de reconnaissance de l'écriture pour numériser les documents manuscrits grâce à des techniques d'apprentissage machine et l'intelligence artificielle (IA).

La reconnaissance de l'écriture manuscrite est l'une des plus vieilles problématiques qui a été posée à l'intelligence artificielle ce qui montre que c'est un véritable défi à la fois technique et scientifique. En effet, reconnaître et comprendre l'écriture met en jeu toutes les composantes de l'IA : il faut visualiser une image et détecter le texte (ce qui suppose de disposer de méthodes de perception visuelle), suivre le tracé de l'écriture puis reconnaître les caractères (grâce à des algorithmes de reconnaissance de formes) et enfin reconnaître les mots et les phrases (par le traitement automatique de la langue) pour aller jusqu'à les comprendre (via une modélisation sémantique).

Dans ce mémoire nous allons nous focaliser sur deux types spéciaux de réseaux de neurones (sous-catégories de l'apprentissage machine) : les RNN et CNN.

Les RNN (réseau de neurones récurrent) sont adaptés pour fonctionner avec des données de séries chronologiques ou des données impliquant des séquences. Les RNN ont le concept de mémoire qui les aide à stocker les états ce qui les rend utile dans le traitement de l'image.

Les CNN (réseau de neurones convolutif) sont plus couramment appliqués pour analyser l'imagerie visuelle le traitement du langage naturel et d'autres problèmes complexes de classification d'images, notamment la reconnaissance d'images, en utilisant une technique spéciale appelée convolution.

Notre travail consiste à mettre en œuvre un système de reconnaissance de l'écriture manuscrite qui a un temps d'exécution et un ratio d'erreur le plus minimum possible avec CNN et les différentes architectures RNN explorées plus précisément les BLSTM (Bidirectional long short term memory). Ce rapport est divisé en 3 chapitres :

Le chapitre 1 nous introduit à des technologies de base reliés à notre travail : l'apprentissage Machine et les réseaux de neurones artificiels et met en avant le concept de la reconnaissance de l'écriture manuscrite et expose la relation entre eux.

Il explore brièvement les CNN et RNN et leurs modes de fonctionnement, vu que notre travail tourne autour de ce dernier et pour nous conduire au 2ème chapitre.

Le chapitre 2 analyse plus en détail la structure des réseaux de neurones convolutifs et les différents types d'architectures des réseaux de neurones récurrents en précisant leur structure générale, leur but et enfin leur efficacité pour mettre au clair la meilleure architecture la plus adaptée au travail que nous voulons accomplir. Pour cette raison, le chapitre se conclut en choisissant une des architectures explorées qui est l'architecture BLSTM.

Le chapitre 3 qui est un chapitre d'implémentation, décrit les différentes étapes à suivre pour construire notre algorithme et met en avant les technologies et les outils (Python, TensorFlow, Jupyter Notebook etc...) que nous pouvons utiliser lors du processus de création de notre système.

# **Chapitre 1**

## **Apprentissage machine, réseaux de neurones & reconnaissance manuscrite**

### **1.1 Introduction**

La Reconnaissance d'écriture manuscrite a pour but de transformer un texte écrit en un texte numérique avec un ensemble des techniques, généralement les réseaux de neurones et particulièrement les réseaux de neurones récurrents. [1]

### **1.2 Apprentissage Machine**

Également appelé apprentissage automatique est une forme d'intelligence artificielle (IA) qui permet à un système d'apprendre à partir des données et non à l'aide d'une programmation explicite.

L'apprentissage automatique n'est pas un processus simple. Au fur et à mesure que les algorithmes ingèrent les données de formation, il devient possible de créer des modèles plus précis basés sur ces données.

Un modèle de Machine Learning est le résultat généré lorsque on entraîne notre algorithme d'apprentissage automatique avec des données. Après l'entraînement, lorsque on fournisse des données en entrée à un modèle, on reçoit un résultat en sortie. [1]

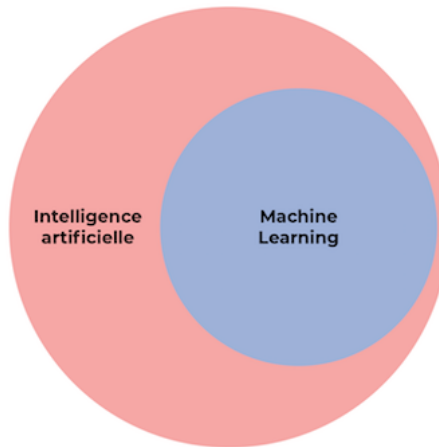


Figure 1 - AI & ML

Il existe plusieurs méthodes d'apprentissage automatique parmi eux il y a 5 méthodes populaires :

### 1.2.1 Régression

Les méthodes de régression entrent dans la catégorie des ML supervisés. Ils aident à prédire ou à expliquer une valeur numérique particulière sur la base d'un ensemble de données antérieures, par exemple en prédisant le prix d'une propriété sur la base de données de tarification antérieures pour des propriétés similaires. [2]

### 1.2.2 Classification

Les méthodes de classification prédisent ou expliquent une valeur de classe. Par exemple, ils peuvent aider à prédire si un client en ligne achètera ou non un produit. La sortie peut être oui ou non : acheteur ou non acheteur. Mais les méthodes de classification ne se limitent pas à deux classes. Par exemple, une méthode de classification pourrait aider à évaluer si une image donnée contient une voiture ou un camion. Dans ce cas, la sortie aura 3 valeurs différentes :

- 1) l'image contient une voiture,

- 2) l'image contient un camion,
- 3) l'image ne contient ni voiture ni camion. [2]

### **1.2.3 Clustering**

Avec les méthodes de clustering, nous entrons dans la catégorie des ML non supervisés car leur objectif est de regrouper ou de regrouper des observations ayant des caractéristiques similaires. Les méthodes de clustering n'utilisent pas les informations de sortie pour l'entraînement, mais laissent plutôt l'algorithme définir la sortie. Dans les méthodes de clustering, nous ne pouvons utiliser que des visualisations pour inspecter la qualité de la solution. [2]

### **1.2.4 Regroupement Apprentissage par renforcement**

Imaginez une souris dans un labyrinthe essayant de trouver des morceaux de fromage cachés. Plus nous exposons la souris au labyrinthe, mieux elle trouve le fromage. Au début, la souris peut se déplacer au hasard, mais après un certain temps, l'expérience de la souris l'aide à comprendre quelles actions la rapprochent du fromage. [2]

### **1.2.5 Traitement du langage naturel**

Un pourcentage énorme des données et des connaissances du monde se trouve dans une forme de langage humain. De toute évidence, les ordinateurs ne peuvent pas encore comprendre pleinement le texte humain, mais nous pouvons les entraîner à effectuer certaines tâches. Par exemple, nous pouvons former nos téléphones à la saisie semi-automatique de nos messages texte ou à la correction des mots mal orthographiés. Nous pouvons même apprendre à une machine à avoir une simple conversation avec un humain. [2]

### 1.3 Réseaux de Neurones Artificiels

Les réseaux de neurones, communément appelés des réseaux de neurones artificiels sont des imitations simples des fonctions d'un neurone dans le cerveau humain pour résoudre des problématiques d'apprentissage de la machine *Machine Learning*. [2]

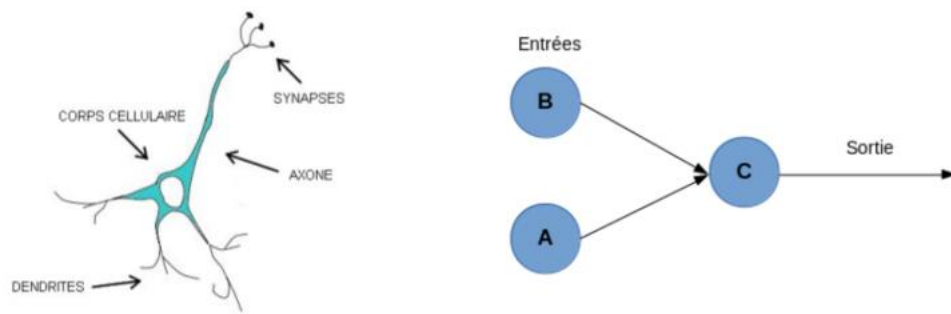


Figure 2 - Réseau de Neurones Simple

Le neurone est une unité qui est exprimée généralement par une fonction sigmoïde. [2]

$$f(x) = \frac{1}{1 + e^{-x}}$$

Figure 3 - Fonction Sigmoïde



Voici les différents composants d'un perceptron :

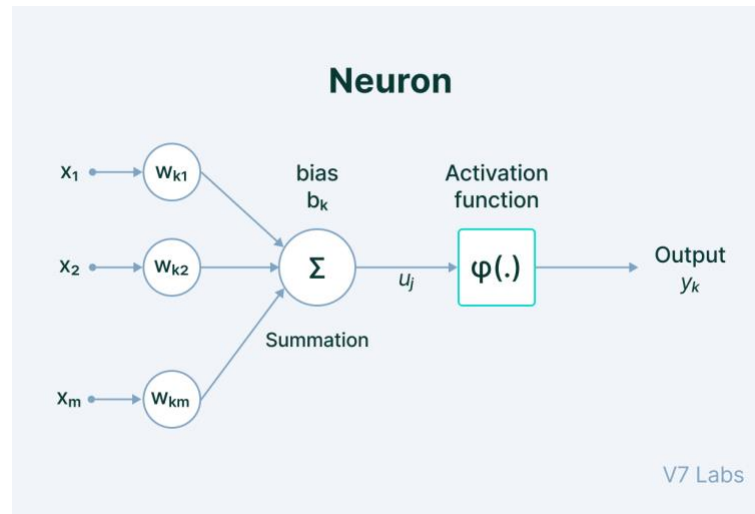


Figure 4 - Composants d'un perceptron

**Entrée :** C'est l'ensemble des caractéristiques qui sont introduites dans le modèle pour le processus d'apprentissage. Par exemple, un tableau de valeurs de pixels appartenant à une image.

**Poids :** Sa fonction principale est de donner de l'importance aux caractéristiques qui contribuent le plus à l'apprentissage. Il le fait en introduisant une multiplication scalaire entre la valeur d'entrée et la matrice de poids. [3]

**Fonction de transfert :** Le travail de la fonction de transfert consiste à combiner plusieurs entrées en une valeur de sortie afin que la fonction d'activation puisse être appliquée. Elle se fait par une simple sommation de toutes les entrées de la fonction de transfert

**Fonction d'activation :** Elle introduit la non-linéarité dans le fonctionnement des perceptrons pour considérer une linéarité variable avec les entrées. Sans cela, la sortie ne serait qu'une combinaison linéaire de valeurs d'entrée et ne pourrait pas introduire de non-linéarité dans le réseau.

**Biais :** Le rôle du biais est de décaler la valeur produite par la fonction d'activation. Son rôle est similaire à celui d'une constante dans une fonction linéaire. [3]

Voici les composants principaux d'un Réseau de neurones multicouches:

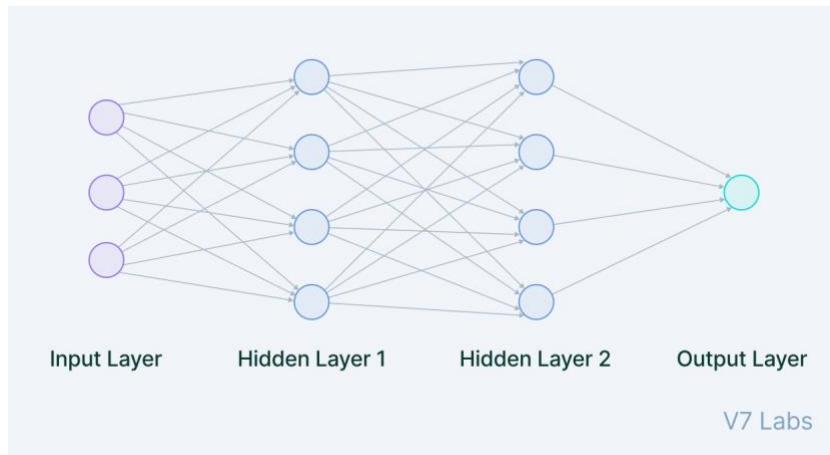


Figure 5 - Réseau de neurones multicouches

**Couche d'entrée :** les données que nous alimentons au modèle sont chargées dans la couche d'entrée à partir de sources externes comme un fichier CSV. C'est la seule couche visible dans l'architecture complète du réseau neuronal qui transmet toutes les informations du monde extérieur sans aucun calcul.

**Couches cachées :** Ce sont des couches intermédiaires qui effectuent tous les calculs et extraient les caractéristiques des données.

Il peut y avoir plusieurs couches cachées interconnectées qui tiennent compte de la recherche de différentes caractéristiques cachées dans les données.

**Couche de sortie :** La couche de sortie prend les données des couches cachées précédentes et aboutit à une prédiction finale basée sur les apprentissages du modèle. C'est la couche la plus importante où nous obtenons le résultat final. [3]

Les domaines d'application des réseaux neuronaux sont souvent caractérisés par une relation entrée-sortie de la donnée d'information :

- La reconnaissance d'image
- Les classifications de textes ou d'images
- Identification d'objets
- Prédiction de données
- Filtrage d'un set de données [2]

## 1.4 Réseaux de Neurones Convolutifs (CNN)

Les réseaux convolutifs sont une forme particulière de réseau neuronal multicouches dont l'architecture des connexions est inspirée de celle du cortex visuel des mammifères. Les CNN sont capables de catégoriser les informations des plus simples aux plus complexes, ils sont caractérisés par leurs premières couches convolutionnelles. [4]

## 1.5 Réseaux de Neurones Récurrents (RNN)

Les réseaux de neurones récurrents ont le pouvoir de se souvenir de ce qu'ils ont appris dans le passé et de l'appliquer dans les prédictions futures. [3]

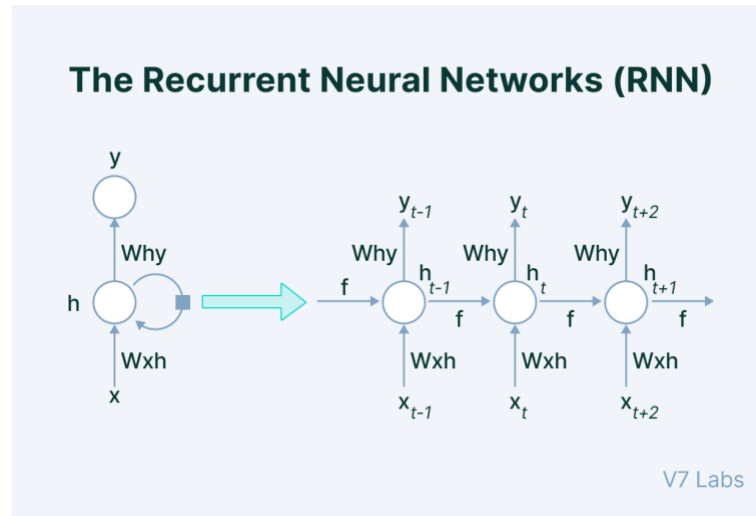


Figure 6 - Réseau de neurones récurrent

L'entrée se présente sous la forme de données séquentielles qui sont introduites dans le RNN, qui a un état interne caché qui est mis à jour chaque fois qu'il lit la séquence de données suivante dans l'entrée. L'état caché interne sera renvoyé au modèle. Le RNN produit une sortie à chaque horodatage. [3]

La représentation mathématique est donnée ci-dessous :

$$h_t = f_w(h_{t-1}, x_t)$$

○ new state      ○ Some function with parameters W  
○ old state      ○ Input vector at some time step

V7 Labs

Figure 7 - Fonction Mathématique

## 1.6 Classification temporelle connexionniste (CTC)

CTC est un type de sortie de réseau neuronal utile pour résoudre les problèmes de séquence tels que l'écriture manuscrite et la reconnaissance vocale. L'utilisation de CTC garantit que l'on n'a pas besoin d'un ensemble de données aligné, ce qui rend le processus de formation plus simple.

Dans le cas de la création d'un OCR (Optical Character Reader), les CRNN produisent un score de caractère pour chaque pas de temps, qui est représenté par une matrice que Nous devons l'utiliser pour :

- Former le réseau de neurones, (calculer loss)
- Décodage de la sortie du réseau de neurones

CTC aide à réaliser les deux tâches.

### 1.6.1 Avantages du CTC

- CTC est formulé de telle manière qu'il ne nécessite que le texte qui apparaît dans l'image. Nous pouvons ignorer à la fois la largeur et la position des caractères dans une image.
- Il n'est pas nécessaire de post-traiter la sortie de l'opération CTC ! En utilisant des techniques de décodage, nous pouvons obtenir directement le résultat du réseau.

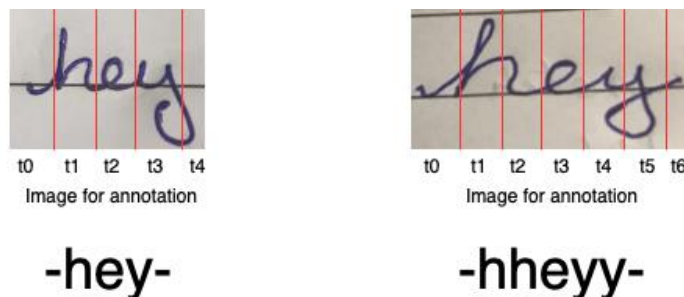


Figure 8 : l'annotation lorsque les caractères occupent un ou plus d'un pas de temps

## 1.6.2 Mécanisme du CTC

### 1.6.2.1 Encodage du texte

CTC fusionne tous les caractères répétés en un seul caractère. Par exemple, si le mot dans l'image est "hé" où "h" prend trois pas de temps, "e" et "y" prennent un pas de temps chacun. Ensuite, la sortie du réseau utilisant CTC sera "hhhey", qui, selon notre schéma de codage, est réduite à "hey"

### 1.6.2.2 Calcul des pertes

Pour entrainer le CRNN, nous devons calculer Loss en fonction de l'image et de son étiquette. Nous obtenons une matrice du score pour chaque caractere à chaque pas de temps du CRNN. Pour calculer Loss, tous les scores des alignements possibles de la vérité terrain sont additionnés. De cette manière, l'emplacement du caractère dans l'image n'est pas significatif.

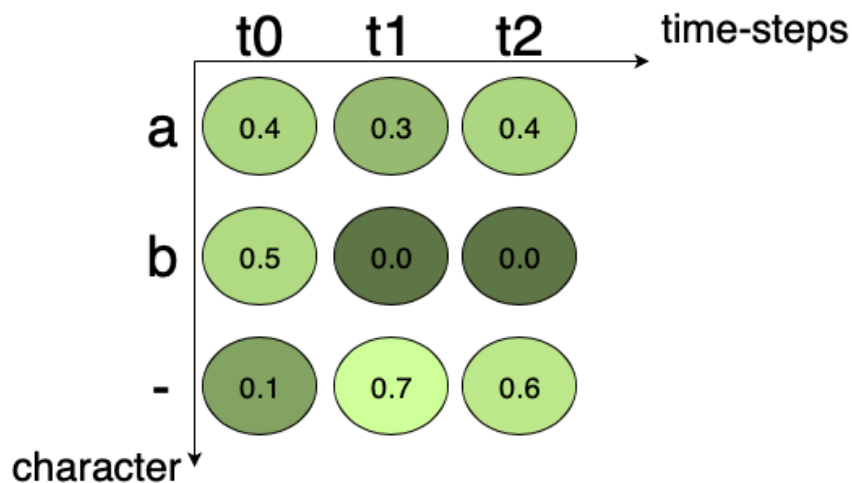


Figure 9 : Matrice de sortie du réseau de neurones

### 1.6.2.3 Décodage

Il se compose des deux étapes suivantes :

- Calcule le meilleur chemin en considérant le caractère avec la probabilité maximale à chaque pas de temps.
- Cette étape consiste à supprimer les blancs et les caractères en double, ce qui donne le texte réel.

Par exemple, considérons la matrice de la Fig.9 Si nous considérons le premier pas de temps  $t_0$ , alors le caractère avec la probabilité maximale est « b ». Pour  $t_1$  et  $t_2$ , le caractère avec la probabilité maximale est '-' et '-' respectivement. Ainsi, le texte de sortie selon l'algorithme du meilleur chemin pour la matrice après décodage est "b".

## 1.7 Reconnaissance Manuscrite

La reconnaissance manuscrite est un traitement informatique qui a pour but de traduire un texte écrit en un texte codé numériquement. La reconnaissance de l'écriture manuscrite fait appel à le Traitement par Vision et au Traitement Automatique de la Langue.

[1]

Il faut distinguer deux reconnaissances distinctes, avec des problématiques et des solutions différentes :

### 1.7.1 Hors-ligne

La reconnaissance hors-ligne travaille sur un instantané d'encre numérique (sur une image). C'est le cas notamment de la reconnaissance optique de caractères. Dans ce contexte, il est impossible de savoir comment ont été tracés les différents motifs. Il est seulement possible d'extraire des formes à partir de l'image, en s'appuyant sur les technologies de reconnaissance de forme. [5]

### 1.7.2 Enligne

Dans le cadre de la reconnaissance en ligne, l'échantillon d'encre est constitué d'un ensemble de coordonnées ordonnées dans le temps. Il est ainsi possible de suivre le tracé, de connaître les posés et levés de stylo et éventuellement l'inclinaison et la vitesse. Il faut évidemment un matériel spécifique pour saisir un tel échantillon, c'est le cas notamment des stylos numériques ou des stylets sur agendas électroniques ou sur les tablettes tactiles. [5]

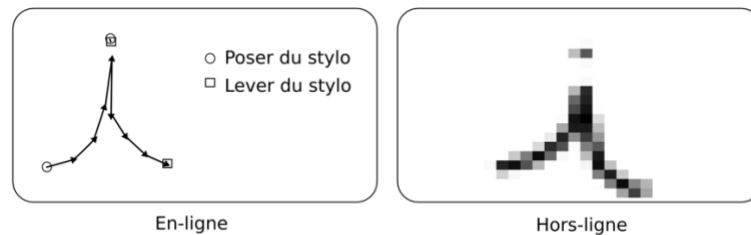


Figure 10 - Reconnaissance d'écriture Manuscrite

### 1.7.3 Utilisation des RNN dans la reconnaissance manuscrite offline

RNN est un type particulier de réseau de neurones caractérisé par la présence de nœuds auto-connectés. Ces nœuds permettent au réseau de mémoriser et de garder une trace des entrées précédentes, ce qui lui permet de stocker et d'accéder aux informations sur de longues périodes de temps. Cette capacité particulière permet au RNN d'apprendre le contexte des étiquettes ce qui s'avère être un gros avantage dans la reconnaissance de l'écriture manuscrite puisque les caractères des mots ne peuvent pas être considérés comme des composants réellement indépendants. Afin d'utiliser les RNN pour la reconnaissance de l'écriture manuscrite, les images contenant du texte manuscrit sont interprétées comme des séquences temporelles le long d'un ou plusieurs axes. [5]



## **1.8 Conclusion**

Il existe plusieurs types de réseaux de neurones avec différentes architectures qui peuvent être utilisées pour la reconnaissance manuscrite, et que dans le chapitre suivant, ces différentes architectures vont être étudiées en détail.

## Chapitre 2

# Réseaux de Neurones Récurrents & Convolutifs

### 2.1 Introduction

Afin de régler des problèmes de performance des réseaux de neurones, les scientifiques ont développés de nouvelles techniques et architectures pour améliorer ces modèles et avoir des résultats plus précis.

### 2.2 Réseaux de Neurones Convolutifs

Les réseaux neurones convolutifs consistent en un empilage multicouche de neurones, des fonctions mathématiques à plusieurs paramètres ajustables, qui prétraitent de petites quantités d'informations. Les réseaux convolutifs sont caractérisé par leurs premières couches convolutionelles (généralement une à trois).

Elles comportent par conséquent deux parties bien distinctes :

**Partie convolutive :** Son objectif final est d'extraire des caractéristiques propres à chaque image en les compressant de façon à réduire leur taille initiale. En résumé, l'image fournie en entrée passe à travers une succession de filtres, créant par la même occasion de nouvelles images appelées cartes de convolutions. Enfin, les cartes de convolutions obtenues sont concaténées dans un vecteur de caractéristiques appelé code CNN.

**Partie classification :** Le code CNN obtenu en sortie de la partie convolutive est fourni en entrée dans une deuxième partie, constituée de couches entièrement connectées appelées perceptron multicouche (MLP pour « Multi Layers Perceptron »). Le rôle de cette partie est de combiner les caractéristiques du code CNN afin de classer l'image. [24]

### 2.2.1 Couche Convulsive

Les paramètres de la couche consistent en un ensemble de filtres apprenants (ou noyaux), qui ont un petit champ récepteur, mais s'étendent sur toute la profondeur du volume d'entrée. Pendant le passage vers l'avant, chaque filtre est convolué sur la largeur et la hauteur du volume d'entrée, calculant le produit scalaire entre les entrées du filtre et l'entrée, produisant une carte d'activation bidimensionnelle de ce filtre. En conséquence, le réseau apprend des filtres qui s'activent lorsqu'il détecte un type spécifique de caractéristique à une certaine position spatiale dans l'entrée.

### 2.2.2 Couche de Pooling

Un autre concept important des CNN est la mise en commun, qui est une forme de sous-échantillonnage non linéaire. Il existe plusieurs fonctions non linéaires pour implémenter la mise en commun, la mise en commun maximale étant la plus courante. Il partitionne l'image d'entrée en un ensemble de rectangles et, pour chacune de ces sous-régions, génère le maximum.

### 2.2.3 Couche ReLU

ReLU est l'abréviation d'unité linéaire rectifiée, qui applique la fonction d'activation non saturante  $f(x) = \max(0, x)$

### 2.2.4 Couche Fully connected

Après plusieurs couches convolutionnelles et max pooling, la classification finale se fait via des couches entièrement connectées. Les neurones d'une couche entièrement connectée ont des connexions à toutes les activations de la couche précédente, comme on le voit dans les réseaux de neurones artificiels réguliers (non convolutifs). Leurs activations

peuvent ainsi être calculées comme une transformation affine, avec une multiplication matricielle suivie d'un décalage de biais

### 2.2.5 Couche Loss

La couche de perte, ou fonction de perte, spécifie comment la formation pénalise l'écart entre la sortie prédite du réseau et les véritables étiquettes de données (lors de l'apprentissage supervisé). Diverses fonctions de perte peuvent être utilisées, selon la tâche spécifique. [25]

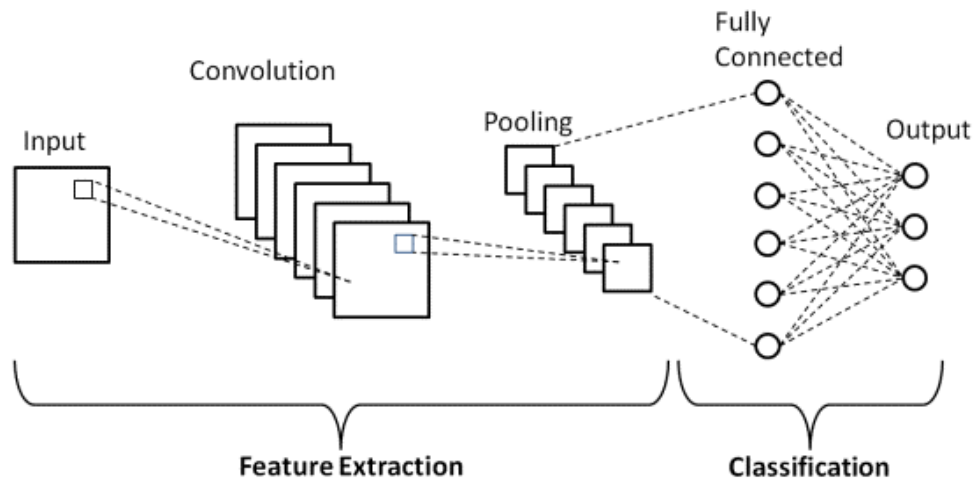


Figure 11 - La structure de CNN

### 2.2.6 Importance de la convolution

La convolution est une opération mathématique simple généralement utilisée pour le traitement et la reconnaissance d'images. Sur une image, son effet s'assimile à un filtrage dont voici le fonctionnement :

- Premièrement, on définit la taille de la fenêtre de filtre située en haut à gauche.

- La fenêtre de filtre, représentant la « feature », se déplace progressivement de la gauche vers la droite d'un certain nombre de cases défini au préalable (le pas) jusqu'à arriver au bout de l'image.
- À chaque portion d'image rencontrée, un calcul de convolution s'effectue permettant d'obtenir en sortie une carte d'activation ou « feature map » qui indique où sont localisées les « features » dans l'image

Un CNN a des couches cachées de couches convolutives qui forment la base des ConvNets. Les « features » font référence à des détails infimes dans les données d'image comme les bords, les bordures, les formes, les textures, les objets, les cercles, etc .

À un niveau supérieur, les couches convolutives détectent ces motifs dans les données d'image à l'aide de filtres. Les détails de niveau supérieur sont pris en charge par les premières couches convolutives [9]. Plus le réseau est profond, plus la recherche de modèles devient sophistiquée. Par exemple, dans les couches ultérieures plutôt que sur les bords et les formes simples, les filtres peuvent détecter des objets spécifiques comme des yeux ou des oreilles, et éventuellement un chat, un chien.

## **2.3 Réseaux de Neurones Récurrents**

### **2.3.1 BRNN**

#### **2.3.1.1 Structure générale**

Les réseaux de neurones récurrents bidirectionnels (BRNN) connectent deux couches cachées de directions opposées à la même sortie. Avec cette forme d'apprentissage profond génératif, la couche de sortie peut obtenir simultanément des informations sur les états passés (en arrière) et futurs (en avant). Inventés en 1997 par Schuster et Paliwal, les BRNN ont été introduits pour augmenter la quantité d'informations d'entrée disponibles sur le réseau. Par exemple, le perceptron multicouche (MLP) et le réseau de neurones à retardement (TDNN) ont des limites sur la flexibilité des données d'entrée, car ils nécessitent que leurs

données d'entrée soient fixes. Les réseaux de neurones récurrents standard (RNN) ont également des restrictions car les futures informations d'entrée ne peuvent pas être atteintes à partir de l'état actuel. Au contraire, les BRNN n'ont pas besoin de fixer leurs données d'entrée. De plus, leurs futures informations d'entrée sont accessibles à partir de l'état actuel. [5]

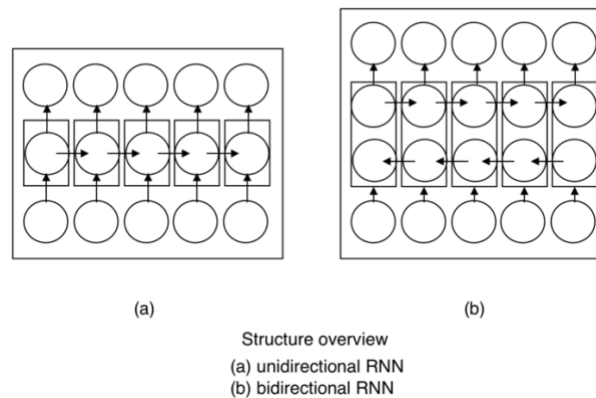


Figure 12 - La structure de RNN et BRNN

Les BRNN peuvent être entraînés à l'aide d'algorithmes similaires aux RNN, car les deux neurones directionnels n'ont aucune interaction. Cependant, lorsque la rétro-propagation dans le temps est appliquée, des processus supplémentaires sont nécessaires car la mise à jour des couches d'entrée et de sortie ne peut pas être effectuée en même temps. Les procédures générales d'entraînement sont les suivantes : pour la passe avant, les états avant et arrière sont passés en premier, puis les neurones de sortie sont passés. Pour le passage en arrière, les neurones de sortie sont passés en premier, puis les états avant et arrière sont passés ensuite. Une fois les passes avant et arrière effectuées, les poids sont mis à jour. [5]

### 2.3.1.2 But de L'architecture

Dans le cas de nombreuses tâches de classification de séquences, pour prendre une décision à un instant donné il est intéressant de connaître le passé et tout ou partie du futur de

la séquence. Les RNN standards traitent les séquences dans l'ordre temporel et n'ont donc pas accès à l'information future.

Pour s'attaquer à ce problème, on peut rajouter de l'information future dans les données d'entrée ou introduire un délai entre un vecteur d'entrée et sa cible pour que le RNN ait le temps de traiter un peu d'information future avant d'avoir à prendre sa décision. Cependant, ces différentes approches rajoutent des contraintes sur l'apprentissage soit en augmentant le nombre de paramètres dans la couche d'entrée, soit en forçant le RNN à apprendre le délai choisi pour donner sa réponse au bon moment. Et dans les deux cas on ne résout pas pour autant le problème puisqu'on introduit des hyper- paramètres (nombre de vecteurs d'entrées à agréger ou délai de réponse) qui peuvent s'avérer difficiles à régler. De plus, on ne supprime en aucun cas la dissymétrie entre le passé et le futur pour ce qui est du contexte exploitable, alors Schuster et Paliwal ont introduit une solution élégante appelée réseau de neurones récurrent bidirectionnel (bidirectional Recurrent Neural Network - BiRNN) qui consiste à présenter chaque séquence à traiter à deux RNN de même type mais avec des paramètres différents :

le premier traite la séquence dans l'ordre naturel  $t : 1 \rightarrow t_f$ ,

le second traite la séquence dans l'ordre inverse  $t : t_f \rightarrow 1$ . [6]

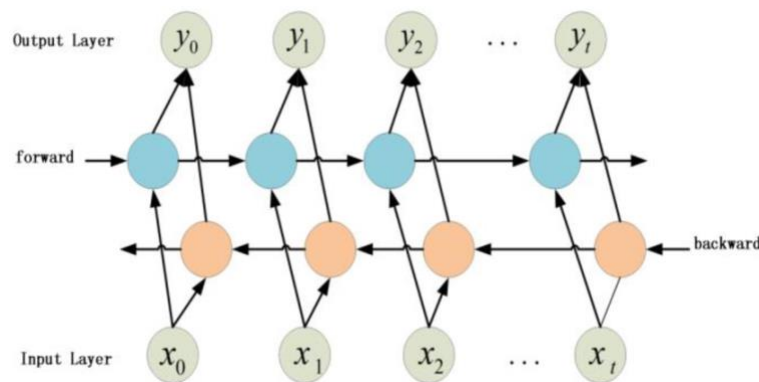


Figure 13 - Flux de calcul du modèle BRNN

### **2.3.1.3 Efficacité**

Les RNN unidirectionnels incluent un historique de phrases illimité, ils sont toujours limités dans le nombre de futurs mots sources qu'ils incluent. Les modèles bidirectionnels offrent un moyen flexible d'inclure également un contexte futur illimité qui, contrairement aux modèles unidirectionnels retardés, ne nécessite aucun réglage pour déterminer la quantité de retard [7]

Les RNN bidirectionnels qui peuvent être entraînés en utilisant toutes les informations d'entrée disponibles dans le passé et le futur, ont été utilisés pour atténuer l'effet du problème des séquences longues et améliorer les performances prédictives. [8]

Les BRNN sont particulièrement utiles lorsque le contexte de l'entrée est nécessaire. Par exemple, en reconnaissance d'écriture manuscrite, les performances peuvent être améliorées par la connaissance des lettres situées avant et après la lettre courante. [5]

## **2.3.2 LSTM**

### **2.3.2.1 Structure générale**

La mémoire à long court terme (LSTM) est une architecture de réseau de neurones récurrents artificiels (RNN) utilisée dans le domaine de l'apprentissage en profondeur. Contrairement aux réseaux de neurones à réaction standard, LSTM a des connexions de rétroaction. Il peut traiter non seulement des points de données uniques (tels que des images), mais également des séquences entières de données (telles que de la parole ou de la vidéo). Par exemple, LSTM est applicable à des tâches telles que la reconnaissance d'écriture manuscrite non segmentée et connectée, la reconnaissance vocale et la détection d'anomalies dans le trafic réseau ou les IDS (systèmes de détection d'intrusion). [11]



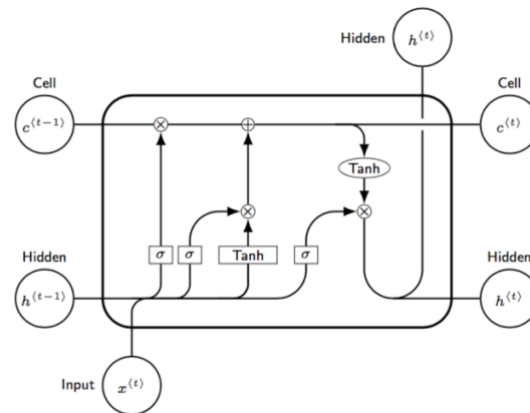


Figure 14 - La Structure de LSTM

Une unité LSTM commune est composée d'une cellule, d'une porte d'entrée, d'une porte de sortie et d'une porte d'oubli. La cellule mémorise les valeurs sur des intervalles de temps arbitraires et les trois portes régulent le flux d'informations entrant et sortant de la cellule.

Les réseaux LSTM sont bien adaptés à la classification, au traitement et à la réalisation de prédictions basées sur des données de séries chronologiques, car il peut y avoir des décalages de durée inconnue entre les événements importants d'une série chronologique.

Un RNN utilisant des unités LSTM peut être entraîné de manière supervisée, sur un ensemble de séquences d'entraînement, à l'aide d'un algorithme d'optimisation, comme la descente de gradient, combiné à une rétropropagation dans le temps pour calculer les gradients nécessaires au cours du processus d'optimisation, afin de modifier chaque poids du réseau LSTM proportionnellement à la dérivée de l'erreur (au niveau de la couche de sortie du réseau LSTM) par rapport au poids correspondant. [11]

### 2.3.2.2 But de L'architecture

Les LSTM ont été développés pour traiter le problème de gradient de fuite qui peut être rencontré lors de la formation de RNN traditionnels. L'insensibilité relative à la longueur

de l'espace est un avantage de LSTM par rapport aux RNN, aux modèles de Markov cachés et à d'autres méthodes d'apprentissage de séquences dans de nombreuses applications. [11]

### **2.3.2.3 Efficacité**

Un problème avec l'utilisation de la descente de gradient pour les RNN standard est que les gradients d'erreur disparaissent de manière exponentielle rapidement avec la taille du décalage temporel entre les événements importants.

Avec les unités LSTM, lorsque les valeurs d'erreur sont rétro-propagées à partir de la couche de sortie, l'erreur reste dans la cellule de l'unité LSTM. Ce "carrousel d'erreurs" renvoie en permanence l'erreur à chacune des portes de l'unité LSTM, jusqu'à ce qu'elles apprennent à couper la valeur. [11]

## **2.3.3 ALSTM**

### **2.3.3.1 Structure générale**

ALSTM est un modèle LSTM classique avec un mécanisme qu'on appelle "attention". Le mécanisme Attention est basé sur le mécanisme d'attention visuelle trouvé dans l'observation humaine. Ce mécanisme aide le modèle à se concentrer sur les informations saillantes. Le but de la couche d'attention est de permettre au modèle d'accorder plus d'attention aux informations importantes proposées par un modèle d'attention anticipée réduit [16]

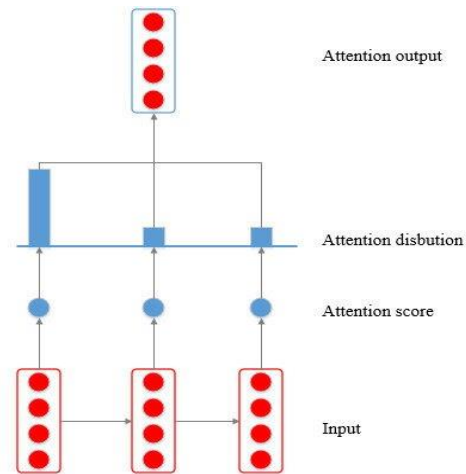


Figure 15 - Schéma du mécanisme d'Attention

Le système montré dans la figure au-dessus est ajouté comme une couche au modèle LSTM ce qui donne :

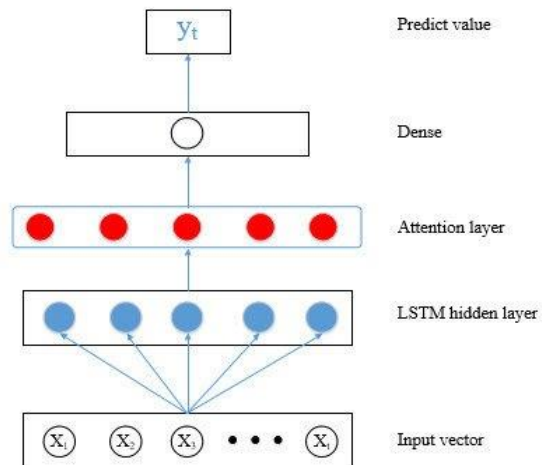


Figure 16 - Modèle LSTM avec Attention

### 2.3.3.2 But de L'architecture

Les mécanismes d'attention ont apporté et apporteront un changement radical dans l'apprentissage automatique et ce fut le cas aussi pour le modèle ALSTM.

Bien que le modèle classique fonctionne bien, il a des limitations qui rendent difficile l'exécution d'une tâche complexe et gère les calculs de manière moins intelligente que l'architecture ALSTM comme par exemple :

- Le modèle classique n'effectue que la perception, représentant un mappage entre les entrées et les sorties.
- Le modèle classique n'effectue pas de raisonnement séquentiel. Ce processus essentiel est basé sur la perception et la mémoire par l'attention et guide les étapes du modèle d'apprentissage automatique de manière consciente et interprétable.

### 2.3.3.3 Efficacité

Les scientifiques ont mis en place différents tests et comparaison pour voir si le ALSTM peut surpasser le modèle classique :

Comparaison entre LSTM et ALSTM à faire des prédictions des actions du marché de la bourse de 5 grandes entreprises :

Tableau 1 - Comparaison entre LSTM et ALSTM

|       | Intel | Wells Fargo | Amazon | Agilent | BE    |
|-------|-------|-------------|--------|---------|-------|
| LSTM  | 0.573 | 0.457       | 0.490  | 0.457   | 0.470 |
| ALSTM | 0.588 | 0.603       | 0.328  | 0.443   | 0.493 |

On voit dans le tableau ci-dessus qu'effectivement ALSTM est plus précis que le modèle classique.

## 2.3.4 BLSTM

### 2.3.4.1 Structure générale

Le LSTM bidirectionnel est un LSTM optimisé, qui peut lire des séquences d'entrée des deux côtés. Cette structure permet à LSTM d'apprendre des motifs séquentiels dans les deux sens. Après le traitement, chaque pas de temps  $t$  de BLSTM générera deux états cachés :

- **htf** : état caché pour la couche avant au pas de temps  $t$ .
- **htb** : état caché pour la couche arrière au pas de temps  $t$ .

Les modèles BLSTM peuvent être divisés en deux types en termes de manière dont ils génèrent leurs états cachés :

- **Processus plusieurs-à-plusieurs** : le modèle concaténerait htf et htb et produirait une instruction à chaque pas de temps.
- **Processus plusieurs-à-un** : le modèle généreraient qu'une seule instruction, qui est déterminée par les états cachés produits par les deux couches directionnelles au dernier pas de temps.

Pour les situations qui nécessitent une sortie à chaque pas de temps, la structure plusieurs-à-plusieurs est plus appropriée. Au contraire, lorsque la situation exige que le modèle produise une sortie généralisée, la structure plusieurs-à-un serait préalable. [9]

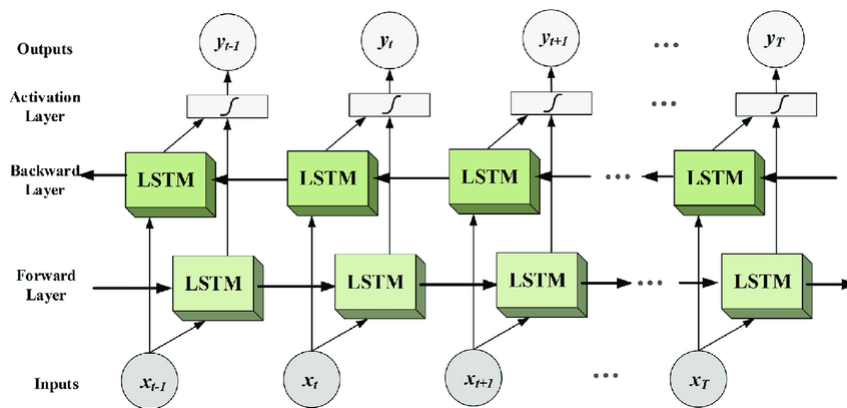


Figure 17 - Flux de calcul du modèle BLSTM

#### **2.3.4.2 But de L'architecture**

La version LSTM de la structure BRNN est appelée LSTM bidirectionnelle (BLSTM). Cette version peut améliorer les performances du modèle LSTM dans les processus de classification. Contrairement à la structure LSTM standard, deux réseaux LSTM différents sont entraînés pour des entrées séquentielles dans l'architecture BLSTM. [12]

#### **2.3.4.3 Efficacité**

Les structures BLSTM peuvent donner de meilleurs résultats que d'autres structures de réseau, selon la zone à problème. Elles sont des réussites significatives dans les tâches de traitement de la parole où le contenu est important. [12]

Les BLSTM combinent un autre LSTM qui recule dans la séquence et est donc capable d'apprendre une représentation qui dépend à la fois du passé et du futur mais qui est la plus sensible aux valeurs d'entrée autour de ce moment particulier. [13]

### **2.3.5 MDLSTM**

#### **2.3.5.1 Structure générale**

Le modèle LSTM multidimensionnel (MDLSTM) a des connexions récurrentes dans plusieurs dimensions au contraire des LSTM normaux qui n'utilisent qu'une récurrence sur une dimension (l'axe des x d'une image ou l'axe du temps pour l'audio). Par exemple, dans les unités 2D LSTM (utilisées dans les systèmes de reconnaissance d'écriture manuscrite de pointe), la valeur cachée dépend des valeurs des axes x et y précédents de l'image ce qui permet à ce dernier de travailler directement sur des images d'entrée brutes et de traiter des données bidimensionnelles de taille illimitée, capturant potentiellement les dépendances à long terme sur les deux axes. L'architecture MDLSTM s'est prouvée d'être extrêmement

efficace dans la reconnaissance d'écriture manuscrite mais en même temps demande une puissance de calculs considérable ce qui la désavantage par rapport aux autres architectures. [18]

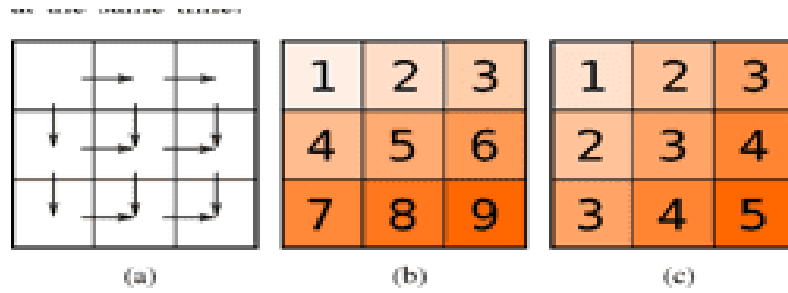


Figure 18 - Dépendances et ordre de calcul du MDLSTM

(a) Les flèches entrantes vers un pixel ont leur origine au niveau des pixels nécessaires au calcul du pixel actuel.

(b) Ordre de calcul naïf : les nombres indiquent l'ordre de calcul qui va de haut en bas, de gauche à droite, un pixel à la fois.

(c) Ordre de calcul en diagonale : tous les pixels sur une diagonale commune sont calculés en même temps.

### 2.3.5.2 But de L'architecture

Le modèle MDLSTM est mis en place pour améliorer la fidélité de la technologie de reconnaissance. En effet, les scientifiques sont en constante recherche pour trouver la solution optimale pour gagner du temps tout en minimisant les erreurs qui peuvent être commises, ce qui a permis à cette nouvelle architecture de naître. [18]

### 2.3.5.3 Efficacité

Récemment, des concours de reconnaissance d'écriture manuscrite ont été remportés par les MDLSTM-RNN et très récemment, les réseaux MDLSTM ont également donné des résultats prometteurs pour la reconnaissance vocale

Tableau 2 - les résultats des entraînements sur IAM

| Système      | WER (word error rate) |      | CER (character error rate) |      |
|--------------|-----------------------|------|----------------------------|------|
|              | dev                   | eval | dev                        | eval |
| Doetsch      | 8.4                   | 12.2 | 2.5                        | 4.7  |
| Voigtlaender | 8.7                   | 12.7 | 2.6                        | 4.8  |
| Pham         | 11.2                  | 13.6 | 3.7                        | 5.1  |

Le tableau compare les résultats des entraînements obtenus par les chercheurs sur la base de données IAM (images texte manuscrit) des scores WER et CER :

- **Doetsch** a utilisé une architecture de réseau LSTM.
- **Voigtlaender** a utilisé des réseaux LSTM et a appliqué un critère d'entraînement discriminant en séquence.
- **Pham** a utilisé un modèle optique MDLSTM plus petit et aucune approche de vocabulaire ouvert pour la reconnaissance, et donc on peut conclure que le modèle MDLSTM est le plus fiable. [18]



## 2.3.6 QRNN

### 2.3.6.1 Structure générale

Abréviation pour "Quasi-Recurrent Neural Networks" permet de faire une approche de la modélisation de séquences neuronales qui alterne des couches convolutifs, qui s'appliquent en parallèle à travers les pas de temps, et une fonction de pooling récurrente minimaliste qui s'applique en parallèle à travers les canaux. Malgré l'absence de couches récurrentes entraînaables. Une révolution par rapport aux vieux modèles où chaque neurone dépend des autres neurones qui s'est montré d'être coûteux en temps et en ressources.

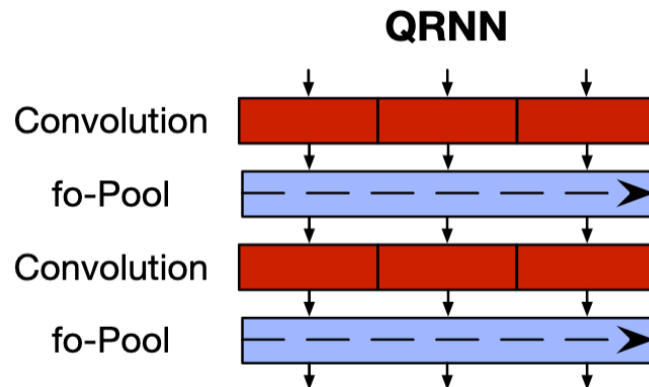


Figure 19 - La structure de QRNN

La partie en "rouge" dans la figure représente les calculs les plus compliqués et lents et qui peuvent être traités en même temps.

La partie en "bleu" représente les fonctions les plus simples et rapides et qui corrigent ce que la partie rouge a eu faux

### 2.3.6.2 But de L'architecture

Pendant longtemps les chercheurs ont supposé que la meilleure manière de traiter les textes est de copier la manière avec laquelle l'être humain lit, c'est à dire mot par mot du

début à la fin mais qui s'est avéré d'être faux. En effet on peut traiter tout à la fois (comme par exemple une image) puis petit à petit comprendre le contexte, d'où l'introduction de cette nouvelle architecture. Elle a été mise en place pour résoudre certaines problématiques que pouvait rencontrer le modèle RNN classique afin de les résoudre de manière plus rapide et plus précise notamment pour les très longues séquences de données.

### 2.3.6.3 Efficacité

D'après plusieurs tests effectués avec des moyens de calcul égaux. Le modèle QRNN s'est montré jusqu'à 16 fois plus rapide que les autres architectures précédentes notamment dans les traductions de texte, l'analyse des sentiments et les prédictions. [15]

Comparaison entre un modèle LSTM et un modèle QRNN :

Tableau 3 - Résultat sur l'analyse de sentiment d'une critique de film

| Modèle | Temps de traitement(s) | Précision |
|--------|------------------------|-----------|
| LSTM   | 480                    | 90.9      |
| QRNN   | 150                    | 91.4      |

Tableau 4 - Résultat sur une traduction d'allemand en anglais

| Modèle | Temps de traitement(h) | Score test BLEU |
|--------|------------------------|-----------------|
| LSTM   | 4.2                    | 16.53           |
| QRNN   | 1                      | 19.41           |

## 2.4 Conclusion

Après être familiarisé avec les différents types d'architecture neuronales, nous avons choisie de nous focaliser sur les CNN standards et l'architecture BLSTM qui a réglé le problème de la descente de gradient pour les RNN standard, les résultats satisfaisants obtenue lors de notre étude et la disponibilité des ressources pour travailler de manière plus facile et fluide ont influencé notre décision.

Dans le prochain chapitre nous allons étudier ce modèle plus profondément et en détail sur le fonctionnement de ces mécanismes.

# Chapitre 3

## Implémentation

### 3.1 Introduction

Pour construire un Modèle pour notre système de reconnaissance, nous allons explorer des étapes importantes et nécessaires pour mener à bien le travail.

En plus, examiner les différents environnements et outils pour l'implémentation et choisir les meilleures options est une étape fondamentale dans le processus de création de notre système d'où l'importance de ce chapitre.

### 3.2 Environnements d'implémentation

En ce qui suit, les outils ML que la plupart des gens utilisent et qui peuvent être d'une grande aide :

#### 3.2.1 TensorFlow

TensorFlow est une bibliothèque open source à grande échelle. Elle utilise à la fois des modèles de ML et de réseau de neurones.

TensorFlow est un outil d'apprentissage automatique compatible avec Python. Il fonctionne à la fois sur CPU et GPU. En outre, TensorFlow entraîne et exécute des modèles de réseaux de neurones.

### 3.2.2 Keras

Keras est un excellent outil ML si vous êtes débutant. Il s'agit d'une API de réseau neuronal avancée. Il s'exécute au-dessus de TensorFlow, Theano, CNTK. Il peut créer à la fois CNN et RNN ou leur combinaison.

La bibliothèque est très conviviale et facile à utiliser. Sa conception est une API spécialement destinée aux humains plutôt qu'aux machines. Keras est l'un des outils d'apprentissage automatique les plus utilisés pour les débutants. C'est également l'un des meilleurs outils d'apprentissage automatique.

### 3.2.3 Pandas

Pandas est l'une des bibliothèques les plus basiques et les plus faciles à pratiquer en ML. Il est généralement utilisé pour Python. Ils gèrent la manipulation des données et d'autres fonctions liées aux données. Il fournit des structures de données rapides et efficaces. Ces structures de données rendent les données structurées et chronologiques très faciles à utiliser. Son objectif est de devenir l'outil de manipulation de données le plus avancé au monde.

### 3.2.4 Numpy

Numpy est un outil d'apprentissage automatique utilisé dans les calculs scientifiques. Des bibliothèques plus avancées comme Tensorflow et Theano fonctionnent sur NumPy.

Pour les calculs basés sur les mathématiques, cette bibliothèque vient en main. Il dispose d'un tableau à N dimensions, d'autres outils sophistiqués pour le calcul des données.

Numpy est un outil basé sur Python. Il s'agit d'un stockage multidimensionnel de données génériques. Par conséquent, nous l'utilisons également dans diverses bases de données.

### 3.2.5 Matplotlib

Matplotlib est une bibliothèque de tracé de graphes utilisée en Python. Avec cela, nous pouvons dessiner des histogrammes, des graphes.

La seule différence est que Matplotlib est utilisé pour le dessin 2D. Alors que la colle peut également être utilisée pour la 3D, elle peut également prendre des images en entrée.

Il est construit sur des tableaux Numpy. L'avantage est qu'il peut prendre une grande quantité de données.

### 3.2.6 Pillow

Pillow est une bibliothèque d'imagerie Python qui y ajoute des capacités de traitement d'image. Elle fournit une prise en charge étendue des formats de fichiers, une représentation interne efficace et des capacités de traitement d'image assez puissantes. La bibliothèque d'images principale est conçue pour un accès rapide aux données stockées dans quelques formats de pixels de base. Il devrait fournir une base solide pour un outil général de traitement d'image.

### 3.2.7 Jupyter Notebook

Le bloc-notes Jupyter est l'une des plateformes/outils d'apprentissage automatique les plus utilisés dans l'industrie. C'est une plateforme de traitement très efficace et rapide. Elle prend en charge trois langages, qui sont Julia, Python et R. En combinant les trois, nous obtenons le nom Jupyter. Le Jupyter Notebook est idéal pour le codage Python. Nous pouvons utiliser Jupyter pour exécuter vos codes ML.

### 3.3 Étapes de construction générales du Modèle

Le processus de construction comprend généralement 5 parties nécessaires pour former un algorithme pour réaliser le système de reconnaissance manuscrite. [19]

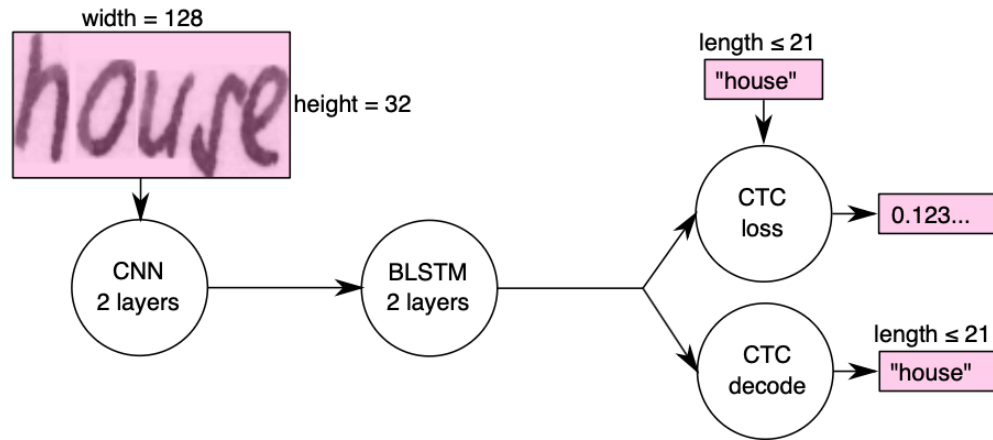


Figure 20 - L'architecture générale du modèle

#### 3.3.1 Collecter les données

La première étape consiste à acquérir des images de documents papier. De cette façon, une image originale peut être capturée et stockée. La plupart des documents papier sont en noir et blanc. En d'autres termes, il doit remplacer chaque pixel d'une image par un pixel noir ou blanc, C'est une méthode de segmentation d'images. [19]

La plupart des scientifiques utilisent des bases de données de large volume pour construire les modèles d'apprentissage comme La base de données IAM qui contient 13 353 images de lignes de texte anglais manuscrites créées par 657 écrivains totalisant 1 539 pages manuscrites comprenant 115 320 mots. La base de données est étiquetée et isolées au niveau de la phrase, de la ligne et du mot. [20]

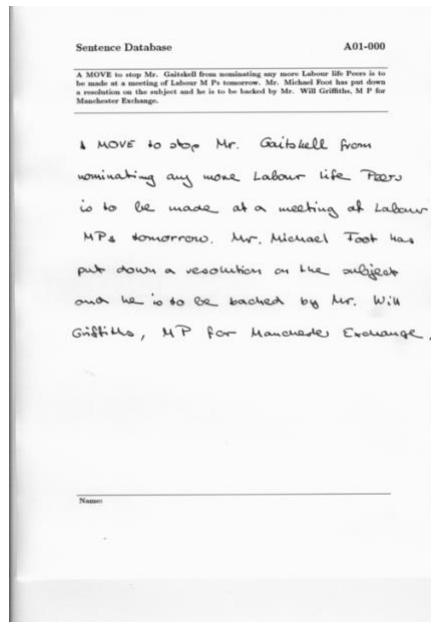


Figure 21 - formulaire contient textes segmentés

### 3.3.2 Prétraitement

Le but du prétraitement est de rendre les données brutes utilisables par les ordinateurs. Le niveau de bruit sur une image doit être optimisé et les zones en dehors du texte supprimées. Le prétraitement est particulièrement important pour reconnaître les documents manuscrits qui sont plus sensibles au bruit et permet d'obtenir une image de caractère propre pour obtenir de meilleurs résultats de reconnaissance d'image.

Cette partie contient 6 étapes :

#### 3.3.2.1 Nettoyer les données

Ignorer toutes les images mal segmentées en utilisant le fichier words.txt qui contient toutes les informations sur l'ensemble des images de la base de données IAM



```

#-- words.txt -----#
#
# iam database word information
#
# format: a01-000u-00-00 ok 154 1 408 768 27 51 AT A
#
#      a01-000u-00-00 -> word id for line 00 in form a01-000u
#      ok              -> result of word segmentation
#                      ok: word was correctly
#                      er: segmentation of word can be bad
#
#      154             -> graylevel to binarize the line containing this word
#      1               -> number of components for this word
#      408 768 27 51   -> bounding box around this word in x,y,w,h format
#      AT              -> the grammatical tag for this word, see the
#                      file tagset.txt for an explanation
#      A               -> the transcription for this word
#
a01-000u-00-00 ok 154 408 768 27 51 AT A
a01-000u-00-01 ok 154 507 766 213 48 NN MOVE

```

Figure 22 - Fichier Words.txt

### 3.3.2.2 Fractionner les données

Cette technique permet de garantir la précision de la création de modèles de données et en divisant l'ensemble de données en trois sous-ensembles :

- Ensemble d'entraînement
- Ensemble de validation
- Ensemble de test

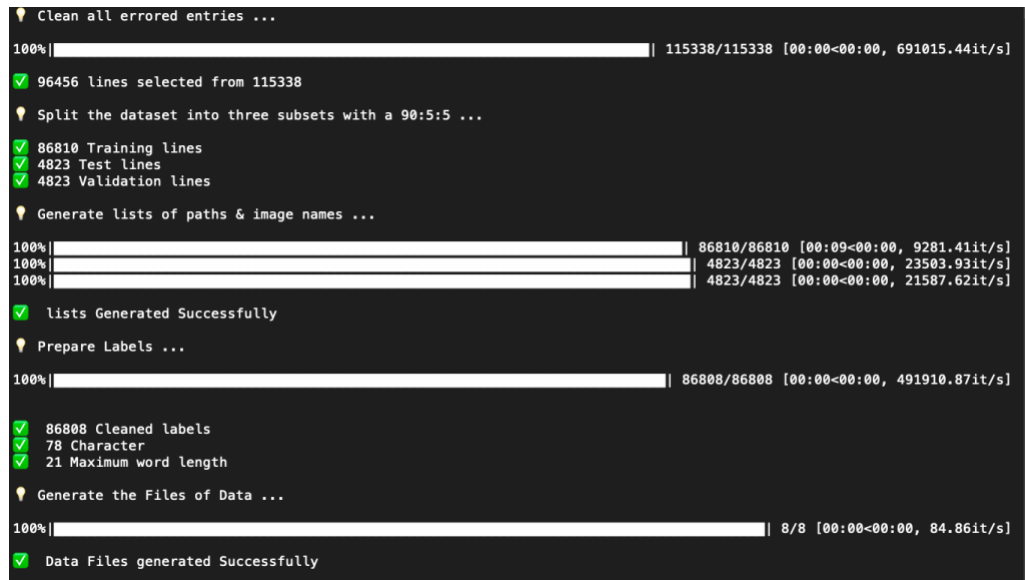
### 3.3.2.3 Préparer les données

Cette étape consiste principalement de générer à partir du fichier words.txt :

- Les Labels et les chemins d'accès directe aux images de la base de données IAM.
- Trouvez la longueur maximale et la taille du vocabulaire dans les données.

### 3.3.2.4 Sauvegarder les données

Créer des fichiers textuels qui contiennent toutes les données générées dans l'étape précédant dans un dossier sous le nom 'Data'.



```

Clean all errored entries ...
100% | 115338/115338 [00:00<00:00, 691015.44it/s]
✓ 96456 lines selected from 115338
Clean all errored entries ...
Split the dataset into three subsets with a 90:5:5 ...
✓ 86810 Training lines
✓ 4823 Test lines
✓ 4823 Validation lines
Clean all errored entries ...
Generate lists of paths & image names ...
100% | 86810/86810 [00:09<00:00, 9281.41it/s]
100% | 4823/4823 [00:00<00:00, 23503.93it/s]
100% | 4823/4823 [00:00<00:00, 21587.62it/s]
✓ Lists Generated Successfully
Clean all errored entries ...
Prepare Labels ...
100% | 86808/86808 [00:00<00:00, 491910.87it/s]
✓ 86808 Cleaned labels
✓ 78 Character
✓ 21 Maximum word length
Clean all errored entries ...
Generate the Files of Data ...
100% | 8/8 [00:00<00:00, 84.86it/s]
✓ Data Files generated Successfully
  
```

Figure 23 - Affichage du Terminal pendant le prétraitement

### 3.3.2.5 Construire le vocabulaire du caractère

Keras fournit différentes couches de prétraitement pour traiter différentes modalités de données. Notre cas implique le prétraitement des étiquettes au niveau du caractère. Cela signifie que s'il y a deux étiquettes, par ex. "chat" et "chien", alors notre vocabulaire de caractères devrait être {a, c, d, g, o, t}. Nous utilisons la couche StringLookup à cette fin.

### 3.3.2.6 Redimensionner les images sans distorsion

Nous devons effectuer notre redimensionnement de telle sorte que les critères suivants soient remplis :

- Le rapport hauteur/largeur est conservé.
- Le contenu des images n'est pas affecté.

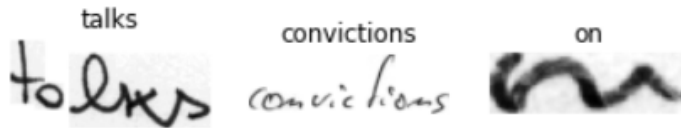


Figure 24 - échantillons avant le redimensionnement



Figure 25 - échantillons après le redimensionnement

### 3.3.3 Construire le modèle

#### 3.3.3.1 Entrée

Une image noire et blanche de 3 dimensions avec une taille de  $32 \times 128 \times 1$  est introduite dans les couches CNN .

### 3.3.3.2 Conv-1 & Conv-2

La première couche convolutive est constituée de 32 noyaux de taille  $3 \times 3$  et la deuxième couche de 64 noyaux de la même taille, Chaque couche se compose de deux opérations. Premièrement, l'opération de convolution, qui applique le noyau de filtre dans les deux couches. Ensuite, la fonction RELU non linéaire est appliquée.

### 3.3.3.3 MaxPool-1 & MaxPool-2

La couche MaxPool-1 et MaxPool-2 suivent Conv-1 et Conv-2 qui consistent en une taille de pool de  $3 \times 3$ .

### 3.3.3.4 Dense-1

La couche dense garantit que chaque neurone reçoit une entrée de tous les neurones de la couche précédente. Dense-1 est connectée avec la sortie de la deuxième couche de convolution et est constituée de 64 neurones, une fonction d'activation RELU et couche dropout avec 0.2.

### 3.3.3.5 BLSTM-1 & BLSTM-2

La première séquence de caractéristiques contient 128 caractéristiques par pas de temps et la deuxième 64 caractéristiques, avec dropout de 0.25. Le RNN propage les informations pertinentes à travers cette séquence. La séquence de sortie RNN est mappée sur la couche Dense-2.

### 3.3.3.6 Dense-2

L'ensemble de données IAM se compose de 79 caractères différents, deux autres caractères supplémentaires est nécessaire pour l'opération CTC (étiquette vierge CTC), il y a donc 80 entrées avec une fonction d'activation Softmax.

### 3.3.3.7 CTC

Notre modèle utilisera la perte CTC comme couche de point final. Lors de l'entraînement du modèle, le CTC reçoit la matrice de sortie et le label et calcule la valeur de perte. Lors de l'inférence, le CTC ne reçoit que la matrice et il la décode dans le texte final. Le texte de référence et le texte reconnu peuvent comporter au maximum 32 caractères.

| Model: "HTR"              |                      |         |                                    |
|---------------------------|----------------------|---------|------------------------------------|
| Layer (type)              | Output Shape         | Param # | Connected to                       |
| image (InputLayer)        | [(None, 128, 32, 1)] | 0       | []                                 |
| Conv-1 (Conv2D)           | (None, 128, 32, 32)  | 320     | ['image[0][0]']                    |
| MaxPool-1 (MaxPooling2D)  | (None, 64, 16, 32)   | 0       | ['Conv-1[0][0]']                   |
| Conv-2 (Conv2D)           | (None, 64, 16, 64)   | 18496   | ['MaxPool-1[0][0]']                |
| MaxPool-2 (MaxPooling2D)  | (None, 32, 8, 64)    | 0       | ['Conv-2[0][0]']                   |
| Reshape (Reshape)         | (None, 32, 512)      | 0       | ['MaxPool-2[0][0]']                |
| Dense-1 (Dense)           | (None, 32, 64)       | 32832   | ['Reshape[0][0]']                  |
| Dropout (Dropout)         | (None, 32, 64)       | 0       | ['Dense-1[0][0]']                  |
| Blstm-1 (Bidirectional)   | (None, 32, 256)      | 197632  | ['Dropout[0][0]']                  |
| Blstm-2 (Bidirectional)   | (None, 32, 128)      | 164352  | ['Blstm-1[0][0]']                  |
| label (InputLayer)        | [(None, None)]       | 0       | []                                 |
| Dense-2 (Dense)           | (None, 32, 81)       | 10449   | ['Blstm-2[0][0]']                  |
| CTC (CTCLayer)            | (None, 32, 81)       | 0       | ['label[0][0]',<br>Dense-2[0][0]'] |
| Total params: 424,081     |                      |         |                                    |
| Trainable params: 424,081 |                      |         |                                    |
| Non-trainable params: 0   |                      |         |                                    |

Figure 26 - Résumé du Modèle

### 3.3.4 Compiler le modèle

Le modèle Keras fournit une méthode, `compile()` pour compiler le modèle. Les attributs de la méthode `compile()` sont les suivants :

- Loss function
- Optimizer
- Metrics

L'optimisation est un processus important qui optimise les poids d'entrée en comparant la prédiction et la fonction de perte. Nous avons utilisé l'optimisateur Adam.

### 3.3.5 Entraîner le modèle

Les modèles sont entraînés par des tableaux NumPy à l'aide de `fit()`. L'objectif principal de cette fonction d'ajustement est utilisé pour évaluer le modèle en entraînement. Il a la syntaxe suivante :

**`model.fit(X, Y, Epochs)`**

- X, Y : C'est un tuple de données.
- Epochs : nombre de fois où le modèle doit être évalué pendant la formation.

Où :

- X = données d'entraînement.
- Y = données de validation.
- Epoch : 50.

### 3.3.6 Évaluer le modèle

L'évaluation est un processus au cours du développement du modèle pour vérifier si le modèle est le mieux adapté au problème donné et aux données correspondantes. Le modèle Keras fournit une fonction, `evaluate` qui effectue l'évaluation du modèle. Il a trois arguments principaux :

- Tester les données
- Étiquette de données de test
- Verbose - vrai ou faux

### 3.3.7 Prédiction du modèle

Enfin, nous prédisons en utilisant les données de test comme ci-dessous

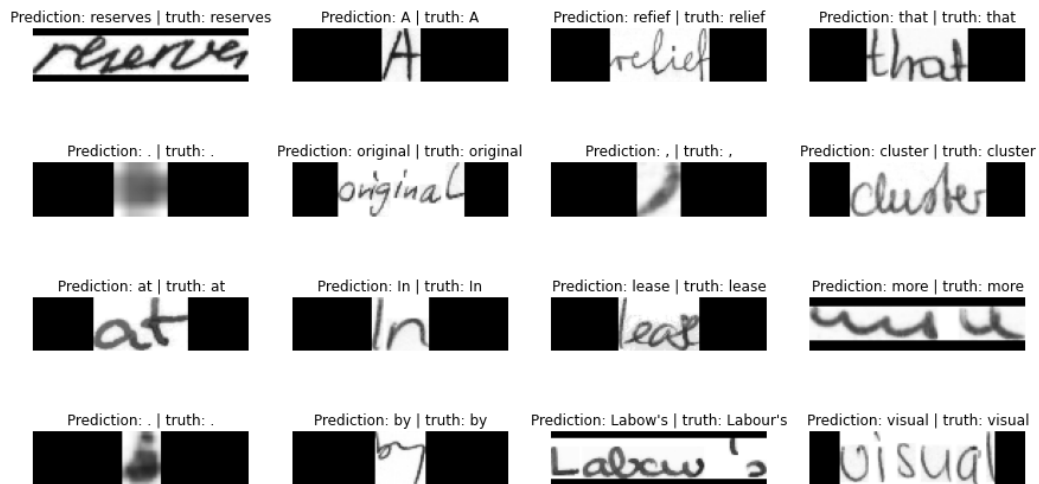


Figure 27 - Prédictions avec les données de test

Et de tester avec des nouvelles images :

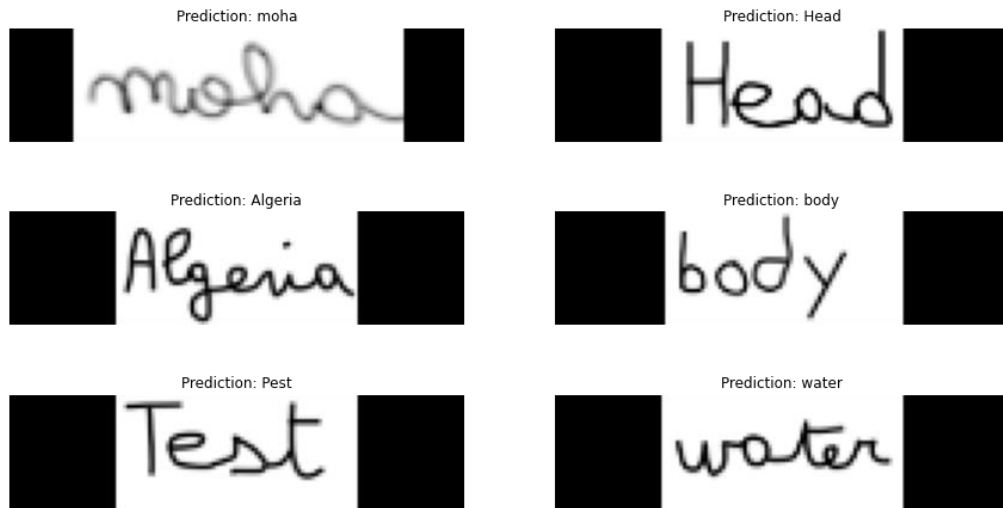


Figure 28 - Prédiction avec nouvelles images

### 3.4 Conclusion

BLSTM surpasse les autres modèles lorsque nous voulons que notre modèle apprenne des dépendances à long terme. La capacité de BLSTM à oublier, mémoriser et mettre à jour les informations lui donne une longueur d'avance sur les RNN.



# Conclusion Générale

La création d'un système de reconnaissance d'écriture manuscrite est une tâche complexe qui requiert beaucoup d'efforts et de ressources et qui combine plusieurs techniques. Il existe plusieurs méthodes avec différents avantages et inconvénients, ou leur performances dépendent entièrement des tâches et des ressources données.

Les études faites dans ce rapport nous ont permis d'avoir une opinion détaillée sur les différentes architectures CNN & RNN et pouvoir choisir l'approche la mieux adaptée au système que nous voulons créer qui est l'architecture BLSTM (Bidirectional Long Short Term Memory).

En effet, les BLSTM ont prouvé être un véritable atout dans le processus de création d'un système de reconnaissance et encore plus la reconnaissance d'écriture manuscrite où les résultats obtenus ont été satisfaisants par rapport à la ressource matérielle donnée dans un temps d'exécution concurrentiel. Sa disponibilité et sa popularité montrent l'efficacité de ce dernier, ce qui a encouragé notre choix .

Pour accomplir à bien notre travail, nous avons pris du temps pour trouver les meilleurs outils et environnements basés sur les critères suivants : Open Source, Facile à utiliser, Disponibilité, Documentation et Recherche fluide etc.

les outils trouvés les mieux adaptés à notre manière de travailler sont :

- Python : créer des programmes rapidement , disponibilité des bibliothèques et les APIs et une très grande communauté active.
- Tensorflow : une meilleure performance une bonne présentation graphique et un soutien communautaire.

- Keras : API simple d'utiliser ,facile a apprendre et est très facile d'ajouter de nouvelles fonctionnalités.
- Google colab : plateforme qui nous permettre d'utiliser un environnement Jupyter Notebook configuration et qui fournis des ressources tel que CPU ,GPU totalement gratuite pour avoir plus de performance

Finalemnt, toutes ces recherches vont nous servir à réaliser notre projet de fin d'étude, la réalisation d'un système de reconnaissance de l'écriture manuscrite à l'aide de différentes technologies modernes. La plus importante est les réseaux de neurones récurrents plus exactement les BLSTM qui va nous permettre de transformer une théorie à un système accompli, c'est-à-dire capable de faire la reconnaissance d'écriture manuscrite de manière fiable et exacte dans un lapse de temps raisonnable.

# Bibliographie

- [1] «Reconnaissance de l'écriture manuscrite,» 20 Novembre 2020. [En ligne]. Available: [https://fr.wikipedia.org/wiki/Reconnaissance\\_de\\_l%27%C3%A9criture\\_manuscrite](https://fr.wikipedia.org/wiki/Reconnaissance_de_l%27%C3%A9criture_manuscrite).
- [2] 2018. [En ligne]. Available: <https://www.juripredis.com/fr/blog/id-19-demystifier-le-machine-learning-partie-2-les-reseaux-de-neurones-artificiels>.
- [3] P. baheti, «The Essential Guide to Neural Network Architectures,» 17 October 2021. [En ligne]. Available: <https://www.v7labs.com/blog/neural-network-architectures-guide>.
- [4] R. M. M. S. Yaroslav Shkarupa, «Offline Handwriting Recognition Using LSTM Recurrent Neural Networks,» Novembre 2016. [En ligne]. Available: [https://www.researchgate.net/publication/311672974\\_Offline\\_Handwriting\\_Recognition\\_Using\\_LSTM\\_Recurrent\\_Neural\\_Networks](https://www.researchgate.net/publication/311672974_Offline_Handwriting_Recognition_Using_LSTM_Recurrent_Neural_Networks).
- [5] «wikipedia,» 1 December 2021. [En ligne]. Available: [https://en.wikipedia.org/wiki/Bidirectional\\_recurrent\\_neural\\_networks](https://en.wikipedia.org/wiki/Bidirectional_recurrent_neural_networks).
- [6] «Reseaux de neurones récurrents pour le traitement automatique de la parole,» Gregory Gelly, Université Paris Saclay, 2017.
- [7] «Translation Modeling with Bidirectional Recurrent Neural Networks,» Martin Sundermeyer, Tamer Alkhouli, Joern Wuebker, and Hermann Ney, Orsay, France, 2014.
- [8] «Diagnosis Prediction in Healthcare via Attention-based Bidirectional Recurrent Neural Networks,» Fenglong Ma ,Quanzeng You.
- [9] J. Bradbury, « New Neural Network Building Block Allows Faster and More Accurate Text Understanding,» einstein.ai, 2016. [En ligne]. Available:

<https://blog.einstein.ai/new-neural-network-building-block-allows-faster-and-more-accurate-text-understanding/>.

- [10] C. P. H. T. J. P. ., R. V. ., S. P. K. M. a. H. G. Dulari Bhatt, «Review CNN Variants for Computer Vision: History, Architecture, Application, Challenges and Future Scope,» <https://www.mdpi.com/journal/electronics>, 2021.
- [11] «Convolutional neural network,» wikipedia, 8 Novembre 2021. [En ligne]. Available: [https://en.wikipedia.org/wiki/Convolutional\\_neural\\_network#History](https://en.wikipedia.org/wiki/Convolutional_neural_network#History).
- [12] «Long short-term memory,» Wikipedia, 29 Novembre 2021. [En ligne]. Available: [https://en.wikipedia.org/wiki/Long\\_short-term\\_memory](https://en.wikipedia.org/wiki/Long_short-term_memory).
- [13] Ö. YILDIRIM, «A novel wavelet sequence based on deep bidirectional LSTM network model for ECG signal classification,» 3 2018. [En ligne]. Available: [https://www.researchgate.net/publication/324056729\\_A\\_novel\\_wavelet\\_sequence\\_based\\_on\\_deep\\_bidirectional\\_LSTM\\_network\\_model\\_for\\_ECG\\_signal\\_classification](https://www.researchgate.net/publication/324056729_A_novel_wavelet_sequence_based_on_deep_bidirectional_LSTM_network_model_for_ECG_signal_classification).
- [14] A. Saeed, «Hybrid Bidirectional LSTM model for Short-Term Wind Speed Interval Prediction,» Septembre 2020. [En ligne]. Available: [https://www.researchgate.net/publication/344481568\\_Hybrid\\_Bidirectional\\_LSTM\\_model\\_for\\_Short-Term\\_Wind\\_Speed\\_Interval\\_Prediction](https://www.researchgate.net/publication/344481568_Hybrid_Bidirectional_LSTM_model_for_Short-Term_Wind_Speed_Interval_Prediction).
- [15] S. Z. Y. S. S. Z. Junshu Jiang, «GL-BLSTM: a novel structure of bidirectional long-short term memory for disulfide bonding state prediction,» 2018. [En ligne]. Available: [arxiv.org](https://arxiv.org).
- [16] «Landslide Deformation Prediction Based on a GNSS Time Series Analysis and Recurrent Neural Network Model».
- [17] «A Comparison of LSTMs and Attention Mechanisms for Forecasting Financial Time Series».

- [18] P. D. a. H. N. P. Voigtlaender, «Handwriting recognition with large multidimensional long short-term memory recurrent neural networks,» ICFHR, 2016.
- [19] K. Spirina, «OCR Algorithm: Improve and Automate Business Processes,» InData Labs, 4 5 2019. [En ligne]. Available: <https://indatalabs.com/blog/ocr-automate-business-processes>.
- [20] U.-V. Marti, « IAM (IAM Handwriting),» paperswithcode, [En ligne]. Available: <https://paperswithcode.com/dataset/iam>.
- [21] A. Ul-Hasan, «Can we build language-independent OCR using LSTM networks?,» 26 02 2014. [En ligne].
- [22] D. Mwit, «Using a Keras Long Short-Term Memory (LSTM) Model to Predict Stock Prices,» kdnuggets, 2018. [En ligne]. Available: <https://www.kdnuggets.com/2018/11/keras-long-short-term-memory-lstm-model-predict-stock-prices.html>.
- [23] F. C. Scott Zhu, «Working with RNNs,» keras, 2019. [En ligne]. Available: [https://keras.io/guides/working\\_with\\_rnn/](https://keras.io/guides/working_with_rnn/).
- [24] «wikipedia,» [En ligne]. Available: [https://en.wikipedia.org/wiki/Convolutional\\_neural\\_network](https://en.wikipedia.org/wiki/Convolutional_neural_network).
- [25] « Réseau de neurones convolutifs,» dataanalyticspost, [En ligne]. Available: <https://dataanalyticspost.com/Lexique/reseau-de-neurones-convolutifs/>.