République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université Mouloud Mammeri de Tizi Ouzou
Faculté de Génie Èlectrique et d'Informatique
Département d'Informatique

Mémoire de fin d'étude

En vue de l'obtention du diplôme Master
Domaine : Mathématique et **Informatique**
Filière : ***Informatique***

Spécialité :

Réseaux, Mobilité et Systèmes Embarqués

Thème

Implémentation d'un modèle de Deep Learning basé sur un réseau de neurones sur la carte STM32

Présenté par :

➢ *HADJ MOHAND* Amel
➢ *ABDERRAHMANI* Rania

Proposé et Encadré par :

M^r *M.* DAOUI

Soutenu le 15/11/2021 devant les membres du jury :

Président : M^r *XXX* Xxxxxx
Examinateurs : M^{me} *XXXXX* Xxxxxx
M^{elle} *XXXX* Xxxx

Année universitaire : 2020/2021.

Remerciements

Nous tenons à remercier en premier lieu le bon Dieu, de nous avoir donné la force, la patience, le courage et la volonté afin d'accomplir ce modeste travail qui présente le fruit de plusieurs années de sacrifices.

Nous remercions nos parents qui ont toujours été là pour nous. Nous remercions nos sœurs et frères pour leurs encouragements. Ainsi que toutes nos familles.

Nous addressons toute notre reconnaissance et sincères remerciement à notre promoteur Mr DAOUI Mehammed, qui a accepté de nous encadrer et pour l'aide continuelle qu'il a fourni et les connaissances qu'il a su nous transmettre. Nous le remercions pour sa patience, sa disponibilité, également pour la qualité de ses conseils et son encouragement tout au long de ce travail.

Nous tenons à remercier également tous les enseignants du département informatique, en particulier les enseignants de la spécialité RMSE, qui nous ont fourni les outils nécessaires à la réussite de nos études universitaires.

Nous remercions par ailleurs vivement les membres du jury de nous avoir fait l'honneur d'examiner notre travail.

Enfin, nous tenons à remercier tous les amis et camarades qui nous ont apporté leur soutien moral et intellectuel tout au long de notre démarche.

Merci à toutes et à tous.

Dédicace

Je dédie ce modeste travail accompagné d'un profond amour :

À mes très chers parents source de vie, pour l'amour qu'ils m'ont toujours donné, leurs encouragements et toute l'aide qu'ils m'ont apportée durant mes études. Aucun mot ne pourrait exprimer mon respect.

À mes chers sœurs et frères, pour tous les sacrifices qu'ils n'ont cessé de m'apporter tout au long de mes années d'études. Que dieu leur apporte le bonheur, les aide à réaliser tous leurs vœux et leur offre un avenir plein de succès.

À ma chère grand-mère, que Dieu lui accorde santé, bonheur, prospérité et longue vie.

À mes Oncles, mes tantes, mes cousins et mes cousines, nulle dédicace ne pourrait exprimer ma profonde affection et mon immense gratitude pour tous les encouragements et soutiens.

À tous mes amis.

À ma chère binôme Amel.

Rania

Dédicace

Je tien à dédier ce modeste travail

À mes très chers parents,

Aucun hommage ne saurait exprimer mon amour éternel, ma reconnaissance et ma considération pour les sacrifices que vous avez consentis pour mon éducation et mon bien être. Vous avez toujours été mon école de patience, de confiance et surtout d'espoir et d'amour. Vous êtes et vous resterez ma référence, la lumière qui illumine mon chemin. Je prie Dieu le tout puissant, de vous accorder une bonne santé, une longue vie, beaucoup de bonheur, et de vous récompenser de toutes les peines et sacrifices données auxquels je ne rendrai jamais assez.

À ma chère sœur Doudouch,

Aucune dédicace ne peut exprimer mon amour et ma gratitude de t'avoir comme sœur. Je ne pourrais jamais imaginer la vie sans toi, je n'oublierais jamais ton encouragement et ton soutien le long de mes études, Merci d'être là pour moi et d'être la grande sœur que tu es. Je te promets de toujours être là pour toi en retour. Je te souhaite beaucoup de succès, de prospérité et une vie pleine de joie et de bonheur.

À mes adorables frères,

Anouar et Fares qui m'ont épaulé durant tout au long de mes études, qui ont été toujours là pour moi. Sans eux je ne serais jamais arrivée là ou je suis aujourd'hui.

À la mémoire de ma grand-mère,

« Yemma Taoues » qui m'a toujours soutenu, poussé et motivé dans mes études. J'espère qu'elle apprécie cet humble geste comme preuve de reconnaissance de la part de sa petite fille qui a toujours prié pour son âme. Puisse dieu, le tout puissant, l'avoir en sa sainte miséricorde. Que ton âme repose en paix.

À ma grande famille,

Mes grands-pères « Jeddi Nouar » et « Jeddi Omar », ma grand-mère maternelle « Yemma l3ali », mes oncles, mes tentes en particulier ma chère tente Saida, et mes cousins(es). Je vous dédie ce travail pour vos attentions particulières, vos prières et votre amour inconditionnel.

À Mes Amis(es), *Mes Camarades de la promotion RMSE « Amel, Lydia, Amira, Karim, Madjid, Moh, Lamine,» et ma binôme Rania.*

À mon très cher ami et frère « Kader ».

À tous ceux qui m'aiment. À tous ceux que j'aime.

Amel

TABLE DES MATIÈRES

LISTE DES TABLEAUX

TABLE DES FIGURES

LISTE DES ABRÉVIATIONS

IA : Intelligence Artificielle.
ML : Machine Learning.
DL : Deep Learning.
NLP : Natural Language Processing.
GPU : Graphics Processing Unit.
CPU : Central Processing Unit.
MCU : µC : Microcontrôleur.
IDE : Integrated Development Environment.
RNA : Réseau de Neurone Artificiel.
CNN : Convolutional Neural Network.
MNIST : Modified ou Mixed National Institute of Standards and Technology.
OCR : Optical Character Recognition.
GPIO : General Puropse Input Output.
UART : Universal Asynchronous Receiver Transmitter.
USART : Universal Synchronous Asynchronous Receiver Transmitter.

INTRODUCTION GÉNÉRALE

Depuis quelques années, l'Intelligence Artificielle (IA) fait l'objet d'une médiatisation et d'une attention sans précédent. Elle devient le phénomène du dernier siècle grâce à d'importantes avancées technologiques qui ont permis d'accroître de façon considérable les performances des ordinateurs dans de nombreux domaines. Ces avancées ont ouvert de vastes perspectives en termes d'innovation technologique et d'automatisation sous différentes formes (applications, robots, etc.). Notons aussi que de plus en plus de secteurs sont concernés (industrie, santé, agriculture, finance, banque, assurance, transport, éducation, etc.).

L'intérêt principal de l'intelligence artificielle est d'imiter les aspects de l'intelligence humaine (raisonnement, apprentissage et résolution de problèmes) et les implémenter dans une machine. Les principaux problèmes rencontrés aujourd'hui découlent du flux important de données qui augmente de manière exponentielle, ce qui rend leur traitement et leur exploitation très difficiles. C'est à partir de là qu'apparait le concept de Deep Learning basé sur les réseaux de neurones artificiels, permettant la manipulation de large quantité de données pour des fins de prédiction, détection et de reconnaissance.

Actuellement, la majorité des fonctions de l'intelligence artificielle sont implémentées au niveau du Cloud, autrement dit le stockage en ligne. Il permet de fournir différents services à la demande au moyen d'Internet. Toutefois, cette situation engendre des problèmes tels que la latence, les risques de sécurité et la dépendance à la disponibilité d'une connexion Internet. Grace aux avancées technologiques dans le domaine des systèmes embarqués, les réseaux de neurones peuvent maintenant tenir dans des systèmes embarqués qui deviennent de plus en plus puissants et à faible consommation d'énergie.

Dans ce cadre, notre projet consiste à implémenter un modèle de Deep Learning basé sur un réseau de neurones artificiels sur une carte de développement STM32F429I-Discovery, dotée d'un microcontrôleur ARM Cortex-M4. Nous prendrons comme cas d'étude la reconnaissance de l'écriture manuscrite en particulier les chiffres manuscrits.

Pour mener à terme notre travail, nous avons adopté la structure suivante :

Le premier chapitre aborde les notions de l'IA et ses deux grandes branches : Machine Learning et Deep Learning, ainsi que leur fonctionnement et leurs domaines d'application. Le deuxième chapitre sera consacré à la présentation de la notion d'image, plus précisément l'image numérique et ses différentes caractéristiques. Ainsi, nous allons voir les différentes méthodes de classification des images, en s'intéressant à la méthode de réseaux de neurones artificiels en particulier convolutifs qui est l'objet de notre travail.

Les deux prochains chapitres seront consacrés à notre contribution. Le chapitre trois présente le cas d'étude de notre travail qui est la reconnaissance de chiffres manuscrits. Ainsi, nous allons présenter les étapes de la mise en œuvre de notre modèle de réseaux de neurones convolutifs. Le chapitre quatre sera consacré à la présentation de l'environnement matériel et logiciel de notre travail. Aussi, nous allons présenter la phase d'implémentation de notre modèle de réseaux de neurones pré-entrainé sur la carte STM32F429I-Discovery.

Nous terminons ce mémoire par une conclusion et des perspectives de ce travail.

CHAPITRE

1

GÉNÉRALITÉS SUR L'IA ET LE DEEP LEARNING

Introduction

L'intelligence artificielle est présente partout, nous nous en servons déjà dans notre vie de tous les jours mais sans s'en rendre compte. Elle a donné naissance à plusieurs techniques qui sont utilisées globalement dans presque tous les domaines. C'est un sujet qui prend place dans un contexte où le numérique est omniprésent dans notre quotidien et les nouvelles technologies engagent de grands changements dans nos modes de vie. Notamment les deux branches de l'IA : Machine Leraning et le Deep Learning attirent beaucoup l'attention, et pour cause, le niveau de performance atteint est tout simplement extraordinaire.

Dans ce chapitre, nous allons décrire les concepts de l'intelligence artificielle ainsi que son fonctionnement, et nous aborderons en détail l'apprentissage automatique en donnant ses différents types illustrés avec des exemples. Puis, nous définirons la notion de l'apprentissage profond en expliquant son fonctionnement et citant ses domaines d'application, pour arriver par la suite à distinguer entre ces deux terminologies. Vers la fin, nous allons voir à propos de l'implémentation du Deep Learning sur le cloud, ses avantages et ses limitations. Et là on arrivera à parler sur l'évolution des systèmes embarquées qui permet la mise en place d'un modèle du Deep Learning sur des microcontrôleurs.

1.1 Intelligence artificielle

1.1.1 Définition

Définir l'intelligence artificielle (IA) n'est pas chose facile, le champ est si vaste qu'il est impossible de la restreindre à un domaine de recherche spécifique. Néanmoins plusieurs définitions ont été attribué à cette notion :

Selon le Larousse, l'intelligence Artificielle est l'ensemble de théories et de techniques mises en œuvre en vue de réaliser des machines capables de simuler l'intelligence.

Marvin Lee Minsky l'un des créateurs de l'IA la définit comme : « La construction de programmes informatiques qui s'adonnent à des taches qui sont, accomplies de façon plus satisfaisantes par des êtres humains car elles demandent des processus mentaux de haut niveau tels que : l'apprentissage perceptuel, l'organisation de la mémoire et le raisonnement critique ».

Plus précisément, l'intelligence artificielle est un ensemble d'algorithmes qui traitent un ensemble d'informations ou de données, relatives à des taches, de manière semblable ou identique à celle qu'adopterait un être humain pour prendre une décision ou résoudre un problème.

Le point commun entre ces définitions se résume dans l'objectif majeur de l'IA qui présente son ambition d'imiter les processus cognitifs de l'être humain. Ces processus comprennent l'apprentissage (acquisition d'informations et de règles liées à leur utilisation), le raisonnement (application des règles pour parvenir à des conclusions approximatives ou précises) et l'auto-correction [1]. L'IA vise aussi un peu loin, cela afin de mettre au point des systèmes qui résolvent certains problèmes bien mieux que les humains, par tous les moyens disponibles.

1.1.2 Fonctionnement

Selon Harry Shum, Président Executif de Microsoft : « l'IA fonctionne seulement s'il y a présence d'une vaste quantité de data, d'une puissance informatique extraordinaire, notamment grâce au cloud, et des algorithmes révolutionnaires, basés sur le deep learning ».

Il existe de nombreuses divisions des types d'intelligence artificielle. Nous nous focalisons sur 2 types :

✓ **IA simple (ou faible)** : conçue pour réaliser des tâches prédéfinies et spécifiques. Par exemple les assistants virtuels par voix de nos smartphones.
✓ **IA complexe (ou fort)** : imite les capacités cognitives humaines. Ce type d'IA, plus avancé, implique la recherche de solutions à des tâches inconnues, sans avoir préalablement défini de solutions pour ces tâches.

L'intelligence artificielle est basée sur des données et des algorithmes qui fonctionnent à partir d'eux grâce au processus détaillé ci-dessous :

➢ Identifier l'importance du problème.
➢ Analyser les situations passées et étudier toutes les variables possibles liées au problème qu'on souhaite analyser.
➢ Grâce à un système de statistiques, prédire le résultat de ce problème, toujours à partir de données connues.
➢ Une fois que le système a toutes les données, il fournit la solution la plus réaliste au problème. Ainsi, l'IA apprend à résoudre automatiquement, dans le futur, un problème semblable.

D'une manière ou d'une autre, l'IA est présente dans presque tous les domaines : éducation, commerce, santé, finance, juridique, industriel, les jeux de réflexion, la recherche mathématique, les assistants personnels et la domotique, la reconnaissance faciale, la compréhension des langues, et la robotique.

Depuis quelques années, on associe presque toujours l'intelligence aux capacités d'apprentissage. C'est grâce à l'apprentissage qu'un système intelligent capable d'exécuter une tâche peut

améliorer ses performances avec l'expérience. C'est grâce à l'apprentissage qu'il pourra apprendre à exécuter de nouvelles tâches et acquérir de nouvelles compétences.

1.2 Apprentissage automatique

L'apprentissage automatique (Machine Learning en anglais) est un sous domaine de l'intelligence artificielle. Il fait référence au développement, à l'analyse et à l'implémentation de méthodes qui permettent à une machine d'évoluer grâce à un processus d'apprentissage, et aussi de remplir des tâches qu'il est difficile ou voire impossible de les remplir par des moyens algorithmiques classiques.

Arthur Samuel, un pionnier dans ce domaine a défini le ML comme : « le domaine d'étude qui donne aux ordinateurs la capacité d'apprendre sans être explicitement programmé ». Pour être plus clair, ce que fait le ML, c'est apprendre à résoudre un problème de manière automatique en utilisant les données.

1.2.1 Fonctionnement

L'apprentissage automatique se fait en deux étapes. La première est la phase d'apprentissage qui s'agit de l'extraction de l'information pertinente de données étudiées au fil d'un processus de mise-à-jour appelé entraînement. Une fois que le modèle est bien déterminé, vient la phase de déploiement où de nouvelles données sont introduites afin de réaliser la tâche souhaitée.

Les algorithmes de l'apprentissage automatique fonctionnent en construisant un modèle à partir d'entrées d'exemple afin de faire des prédictions ou des décisions basées sur les données, dont l'objectif est de minimiser ce qu'on appelle l'erreur, c'est-à-dire de se tromper le moins possible.

Le machine learning et l'IA sont souvent abordés ensemble, et les termes sont parfois utilisés de manière interchangeable, mais ils ne veulent pas dire la même chose. Une distinction importante est que, même si l'intégralité du machine learning repose sur l'intelligence artificielle, cette dernière ne se limite pas au machine learning.

1.2.2 Domaines d'application

De nos jours, cette technologie est présente dans divers domaines, Lorsque nous interagissons avec les banques, achetons en ligne ou utilisons les médias sociaux, des algorithmes de machine

learning entrent en jeu pour optimiser, fluidifier et sécuriser notre expérience. Elle est représentée notamment dans les robots marcheurs qui apprennent seuls. Le ML est aussi utilisé dans le domaine médical afin d'assister au mieux les spécialistes, et tant d'autres utilisation tels que :

- Traitement d'image et vision par ordinateur : la reconnaissance faciale, la détection d'objets.
- Biologie : la détection des tumeurs.
- Traitement du langage naturel : les applications de reconnaissance vocale.
- Finance : Détection de fraudes.
- Automobile : pour la maintenance prédictive.
- Agriculture : la prévision des demandes en eau.
- Réseaux : Prédiction du trafic (Volume de trafic attendu), classification du trafic (Dropbox, Facebook, LinkedIn, Skype, YouTube), prédiction des pannes, sécurité du réseau.[2]

1.2.3 Types d'apprentissage automatique

Généralement dans l'apprentissage machine, les tâches à résoudre sont classées en catégories. Ces catégories sont définies selon la façon dont l'apprentissage est reçu ou comment le retour d'informations est donné au système développé.

Il existe plusieurs méthodes d'apprentissage automatique. Si les classes sont prédéterminées et les exemples connus, le système apprend à classer selon un modèle de classement ; on parle alors d'apprentissage supervisé .Par contre, quand le système ne dispose que d'exemples, mais non d'étiquettes, et que le nombre de classes et leur nature n'ont pas été prédéterminés ; on parle d'apprentissage non supervisé. Dans ce cas on ne fournit pas à l'algorithme des données étiquetées afin de lui pouvoir permettre de définir une structure et de découvrir une logiques dans les données entrées. Citons aussi l'apprentissage par renforcement et l'apprentissage profond. Explorons donc ces méthodes plus en détail dans ce qui suit.

1.2.3.1 L'apprentissage supervisé :

L'apprentissage supervisé est une variété de machine Learning qui utilise un ensemble de données d'apprentissage étiquetées afin de créer des modèles d'intelligence artificielle. Le but de cette méthode est d'être capable d'apprendre en comparant sa sortie réelle avec les sorties enseignées pour trouver des erreurs et modifier le modèle en conséquence. Donc l'apprentissage supervisé utilise des modèles pour déterminer les valeurs d'étiquettes pour un ensemble de données non étiquetées. Le schéma de la figure 1.1 montre le fonctionnement de l'apprentissage supervisé.

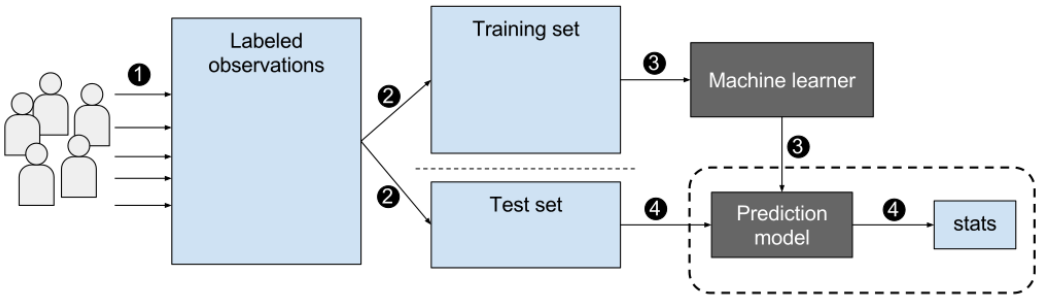


FIGURE 1.1 – Fonctionnement de l'apprentissage supervisé.

Pour pouvoir illustrer ce type d'apprentissage, on peut se baser sur l'exemple d'un enfant qui apprend à identifier des animaux en l'apprenant à partir d'un livre d'images. Dans le cadre de l'apprentissage supervisé, l'algorithme est entraîné par un ensemble de données qui est déjà étiqueté et qui a une sortie prédéterminée.

Cette méthode d'apprentissage permet de réaliser deux types de tâches :

✓ **Des tâches de classification** :

Ces tâches consistent à attribuer une classe à des objets. Par exemple, dans le milieu bancaire, on peut identifier si une transaction est frauduleuse ou non frauduleuse de manière automatique. On parle de détection d'anomalie. Dans l'industrie, on peut déterminer si oui ou non une machine est susceptible de tomber en panne. On associe une réponse prédéfinie (oui ou non, jaune, rouge, vert ou bleu) à un objet, avant de demander à l'algorithme de réaliser cette classification.

✓ **Des tâches de régression** :

Ici, on n'attribue pas une classe mais une valeur mathématique : un pourcentage ou une valeur absolue. Par exemple, une probabilité pour une machine de tomber en panne (15 %, 20 %, etc.) ou le prix de vente idéal d'un appartement en fonction de critères comme la surface, le quartier, etc.

1.2.3.2 L'apprentissage non supervisé :

Dans le cas de l'apprentissage non supervisé, on utilise un ensemble de données d'entrées non étiquetées, afin de laisser l'algorithme d'apprentissage trouver tout seul les points communs parmi cet ensemble de données. Les méthodes d'apprentissage automatique qui facilitent l'apprentissage non supervisé sont particulièrement utiles, vu que les données non étiquetées étant plus considérables que celles étiquetées. On peut considérer que l'objectif initial de l'apprentissage non supervisé est aussi simple que de détecter les modèles cachés dans un ensemble de données, mais

il peut aussi avoir un objectif d'apprentissage des caractéristiques, ce qui va rendre la machine intelligente capable de découvrir automatiquement les représentations nécessaires pour classer des données brutes.

Pour rester sur l'exemple précédent d'apprentissage des animaux pour un enfant, avec la méthode non supervisée, cet enfant apprend à identifier ces animaux en observant les couleurs, les dessins et les caractéristiques, plutôt que de mémoriser les noms à l'aide d'un livre ou d'un enseignant. L'enfant recherchera les points communs entre les images et sera capable de les séparer en groupes, en attribuant à chaque groupe sa propre nouvelle étiquette.

Pour y voir plus clair, voici des exemples de tâches réalisables grâce à cette méthode :

✓ **Des tâches de clustering :**

Ici, on demande à la machine de grouper des objets dans des ensembles de données les plus homogènes possible. Cette technique peut sembler proche de celle de la classification dans l'apprentissage supervisé. Mais à la différence de cette dernière, les classes ne sont pas préremplies par un humain, c'est la machine qui invente ses propres classes, à un niveau de finesse pas toujours évident pour un humain. C'est une technique très utile dans le marketing pour faire de la segmentation client.

Imaginons qu'on veuille segmenter des automobilistes, pour construire des offres de mobilité personnalisée pour les clients, grâce à des algorithmes d'apprentissage. Le clustering rapproche des individus aux habitudes très similaires dans un groupe et éloigne des individus très différents. On obtient alors des groupes homogènes aux caractéristiques propres. Par exemple, les individus qui utilisent leur voiture en semaine et passent du temps dans les bouchons, ceux qui ont plutôt tendance à conduire le week-end pour des trajets courts, etc. Cela permet au marketing de leur proposer des offres week-end, semaine, etc. Le clustering permet d'atteindre des niveaux de précision impossibles pour un humain et d'identifier des schémas qu'il n'aurait pas pu repérer.

✓ **Des tâches de filtrage collaboratif :**

Ici l'objectif est toujours de personnaliser une expérience client. C'est une technique utilisée par de très nombreuses plateformes, tel que Netflix,. Leurs algorithmes étudient ce que vous avez regardé, aimé, mais aussi ce que des profils similaires au vôtre ont apprécié, pour vous faire des recommandations automatiques. Le modèle s'appuie sur des facteurs implicites inconscients (ce que l'utilisateur a fait), plutôt que des facteurs explicites (des critères remplis par l'utilisateur). Ce genre d'outils est très utilisé dans le E-commerce, pour proposer des produits qui pourraient intéresser le client. Amazon en est devenu spécialiste.

❖ **Différence entre apprentissage supervisé et non supervisé :**

En parallèle de l'apprentissage supervisé, il est également possible d'effectuer un apprentissage non supervisé. Le principe reste similaire dans le fait de préserver l'autonomie de l'intelligence artificielle dans ses fonctions. Cependant, la méthode non supervisée n'utilise pas d'exemples ni de modèles d'étiquetage. En l'absence de ces données de base, le système doit alors s'appuyer sur des critères communs. Selon la cible, il peut s'agir d'une forme ou d'un usage.

1.2.3.3 L'apprentissage par renforcement :

L'apprentissage par renforcement fait référence à une classe de problèmes d'apprentissage automatique. Il consiste à apprendre à partir d'expériences successives, ce qu'il convient de faire de façon à trouver les meilleures solutions. Autrement dit, les machines intelligentes essaient plusieurs situations afin de pouvoir déterminer les actions les plus avantageuses, et ne se contentent pas de recevoir des instructions sur les actions à appliquer, ce qui distingue cette méthode des autres techniques d'apprentissage. L'apprentissage par renforcement est un modèle d'apprentissage comportemental. L'algorithme dans ce cas reçoit les informations en analysant des données, pour pouvoir orienter l'utilisateur vers les meilleurs résultats. Dans ce type d'apprentissage, le système n'est pas entraîné à partir d'un ensemble de données mais il apprend par essais et erreurs, ce qui le diffère des autres types d'apprentissage supervisé.

Prenons un drone autonome qui doit livrer un colis d'un entrepôt de livraison vers une maison. Dans ce cas, l'acteur est le drone. Il peut réaliser différentes actions : avancer, reculer, descendre, monter, accélérer, freiner, etc. Chacune de ses actions modifie son état et l'état de l'environnement. Son but est de se rendre sans encombre, en 30 minutes, à l'adresse indiquée et revenir. Il se lance alors, fais des premiers choix, joue sur différentes variables. Il évalue ses performances et comprend petit à petit ce qui fonctionne le mieux. Au bout de nombreuses tentatives, il finit par remplir sa mission de manière efficace.

C'est également cette méthode qui sera exploitée pour construire les algorithmes des voitures autonomes. L'entraînement d'une voiture autonome est un processus extrêmement complexe à cause de nombreux obstacles possibles. Si toutes les voitures étaient autonomes, les essais et les erreurs seraient plus faciles à surmonter, mais dans le monde réel, les facteurs humains sont souvent imprévisibles. [3]

1.2.3.4 L'apprentissage profond :

L'apprentissage profond est l'une des principales technologies du Machine Learning et d'intelligence artificielle. Nous allons découvrir en quoi consiste cette technologie, son fonctionnement, et ses différents secteurs d'application dans les sous titres suivants où tout sera détaillé.

1.3 Apprentissage Profond

L'apprentissage profond ou Deep Learning (DL) en anglais, est un type d'intelligence artificielle dérivé du machine learning où la machine est capable d'apprendre et de s'améliorer de manière autonome. Contrairement à la programmation où elle se contente d'exécuter à la lettre des règles prédéterminées.

L'apprentissage en profondeur utilise une succession de couches d'unités de traitement non linéaire pour pouvoir extraire ou transformer les caractéristiques des données. La sortie d'une couche sert d'entrée de la couche suivante. Les algorithmes de l'apprentissage profond peuvent être supervisés et servir à classer les données, ou non supervisés et aider à effectuer une analyse du modèle. L'algorithme de Deep Learning absorbe des quantités de données énormes par rapport aux autres algorithmes d'apprentissages machine utilisés et développés actuellement. Il a été capable de battre les humains dans certaines tâches cognitives.

Par exemple, la reconnaissance faciale par ordinateur et la reconnaissance vocale ont connu des progrès significatifs et cela grâce aux approches d'apprentissage approfondies.

1.3.1 Fonctionnement

Le Deep Learning s'appuie sur un réseau de neurones artificiels s'inspirant du fonctionnement des neurones biologiques du cerveau humain. Cette structure est disposée en plusieurs couches interconnectées entre elles.

La première couche correspond aux neurones d'entrée et la dernière transmet les résultats de sortie. Entre ces deux se trouvent plusieurs couches intermédiaires par lesquelles l'information est traitée. Cette architecture est propre au deep learning et permet que chaque couche analyse de manière plus précise les données d'entrée.

Ainsi, plus le réseau de neurones artificiels est profond et donc contient plusieurs couches, plus le système peut effectuer des tâches complexes. Il est capable de déterminer par lui-même une représentation de ce qu'il reçoit, que ce soit une image ou un texte. Par exemple, à partir de portraits humains, le programme va d'abord distinguer le visage des cheveux, puis reconnaître le

nez, la bouche, les yeux, etc.

À chaque information intégrée, les connexions entre neurones s'étendent et se modifient. C'est pour cela qu'un système avec un IA à apprentissage profond a la capacité d'apprendre de nouvelles choses en autonomie. Il améliore également de lui-même ses prévisions et ses prises de décision, sans qu'aucune intervention humaine ne soit requise. Il a donc pour particularité d'apprendre de ses propres erreurs. [4]

1.3.2 Domaines d'applications du Deep Learning

En plein essor depuis une dizaine d'années, le deep learning s'avère utile dans divers secteurs. Il a permis d'obtenir des résultats impressionnants dans des domaines aussi nombreux. Passons en revue quelques applications :

- Les IA à Deep Learning sont très efficaces pour les analyses d'images. Elles sont par exemple employées dans l'imagerie médicale pour détecter des maladies ou dans le secteur automobile dans le cas des voitures autonomes. Mais aussi pour les reconnaissances faciales comme sur les smartphones ou sur Facebook.
- De plus, le Deep Learning est également un atout dans la création de contenu. En effet, un ordinateur peut être capable de rédiger de manière autonome des textes ou d'effectuer des traductions. La seule condition est l'accès à une quantité de données suffisante de formation. Cela fait partie du NLP (Natural Language Processing) une branche de l'IA, qui traite automatiquement le langage humain.
- C'est également le cas des assistants vocaux, tels que Siri, Alexa ou Google Home. Ceux-ci se fondent sur la technologie du deep learning pour développer leur compréhension du langage et leur vocabulaire. Tout comme les chatbots qui permettent de répondre de plus en plus précisément aux diverses demandes des clients.
- Par ailleurs, l'apprentissage profond trouve toute sa place dans le marketing. Il facilite l'élaboration de campagnes publicitaires et d'e-mails ultra personnalisés. Il peut aussi servir à optimiser le score des leads, à classer et à faire remonter les problèmes des clients.
- D'autre part, il est utilisé en sécurité informatique pour identifier les dangers documentés et les risques inconnus. Il est en effet, capable de détecter des anomalies et de renforcer les mesures de sécurité.
- Également, l'apprentissage profond est très présent dans le domaine industriel. Que ce soit en matière de robotique ou dans les solutions de maintenance.

De toute évidence, ce ne sont qu'une petite partie des vastes applications auxquelles l'apprentissage en profondeur peut être appliqué. On pourrait citer encore beaucoup d'exemple plus originaux les uns que les autres, mais ce qu'il faut retenir c'est que le Deep Learning permet de faire apprendre à un ordinateur une tache précise en observant un grand nombre d'exemples.

1.4 Deep Learning vs Machine Learning

Le Deep Learning constituant un sous-domaine du Machine Learning, il convient de connaître les différences entre ces deux programmes d'intelligence artificielle (Figure 1.2).

Le Machine Learning est la technologie la plus ancienne et la plus simple. Elle s'appuie sur un algorithme qui adapte lui-même le système à partir des retours faits par l'humain. La mise en place de cette technologie implique l'existence de données organisées. Le système est ensuite alimenté par des données structurées et catégorisées lui permettant de comprendre comment classer de nouvelles données similaires. En fonction de ce classement, le système exécute ensuite les actions programmées. Il sait par exemple identifier si une photo montre un chien ou un chat et classer le document dans le dossier correspondant.

Le Deep Learning n'a pas besoin de données structurées. Le système fonctionne à partir de plusieurs couches de réseaux neuronaux, qui combinent différents algorithmes en s'inspirant du cerveau humain. Cette approche est particulièrement adaptée pour les tâches complexes, lorsque tous les aspects des objets à traiter ne peuvent pas être catégorisés en amont.

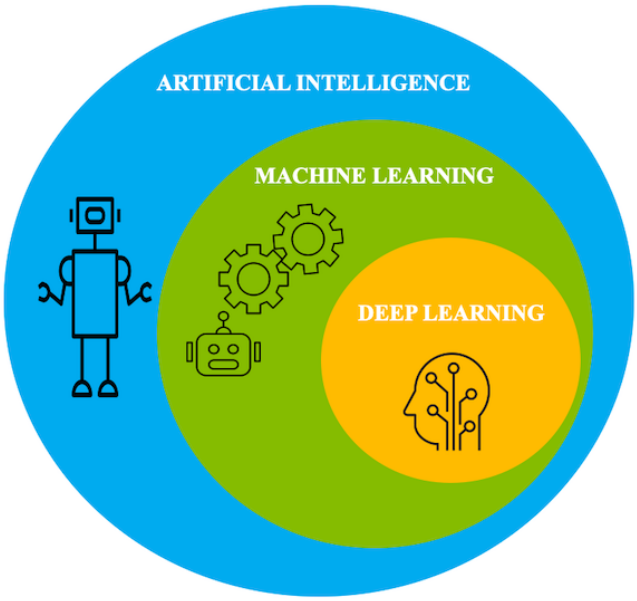


FIGURE 1.2 – Intelligence Artificielle, Machine Learning et Deep Learning.

Avec le Deep Learning, le système identifie lui-même les caractéristiques discriminantes des données, sans avoir besoin d'une catégorisation préalable. Le système n'a pas besoin d'être entraîné par un développeur. Il évalue lui-même le besoin de modifier le classement ou de créer des catégories inédites en fonction des nouvelles données. Tandis que le Machine Learning fonctionne à partir d'une base de données contrôlable, le Deep Learning a besoin d'un volume de données

bien plus considérable pour donner des résultats fiables.

Un IA à apprentissage profond possède également la capacité à apprendre en toute autonomie, contrairement à un IA à apprentissage automatique, qui nécessite l'intervention humaine.

Par ailleurs, la technologie nécessaire pour le Deep Learning est plus sophistiquée. Elle exige plus de ressources IT et s'avère nettement plus coûteuse que le Machine Learning. Elle n'est donc pas intéressante, du moins à l'heure actuelle, pour une utilisation de masse par les entreprises.

❖ **Synthèse des différences entre Machine Learning et Deep Learning [5] :**

	Machine Learning	**Deep Learning**
Organisation de données	Données structurées	Données non structurées
Base de données	Contrôlable	>1 million d'enregistrements
Entrainement	par l'humain nécessaire	Système d'apprentissage autonome
Algorithme	Algorithme modifiable	Réseau neuronal d'algorithmes
Champ d'application	Actions simples de routine	Tâches complexes

TABLE 1.1 – Différence entre Machine Learning et Deep Learning.

1.5 L'lA et le Cloud Computing

De manière générale, le Cloud Computing reste un ensemble de solutions qui permettent de fournir différents services à la demande au moyen d'Internet. Par ce système, toutes nos données sont enregistrées sur un serveur distant au lieu qu'elles soient sur le disque dur de notre ordinateur.

En utilisant le Cloud Computing, les utilisateurs et les entreprises n'ont pas à gérer eux-mêmes les serveurs physiques ni à exécuter des applications logicielles sur leurs propres machines. Le Cloud permet aux utilisateurs d'accéder aux mêmes fichiers et aux mêmes applications à partir de presque n'importe quel appareil. Car les processus informatiques et le stockage ont lieu sur des serveurs dans un datacenter et non localement sur la machine utilisateur. C'est pourquoi un utilisateur dont le téléphone est défaillant peut se connecter à son compte Instagram à partir d'un nouveau téléphone et retrouver son compte actif en place, avec toutes ses photos, vidéos et l'historique de ses conversations. Il en va de même avec les fournisseurs de messagerie cloud

comme Gmail ou Microsoft Office 365 et les fournisseurs de stockage cloud comme Dropbox ou Google Drive.

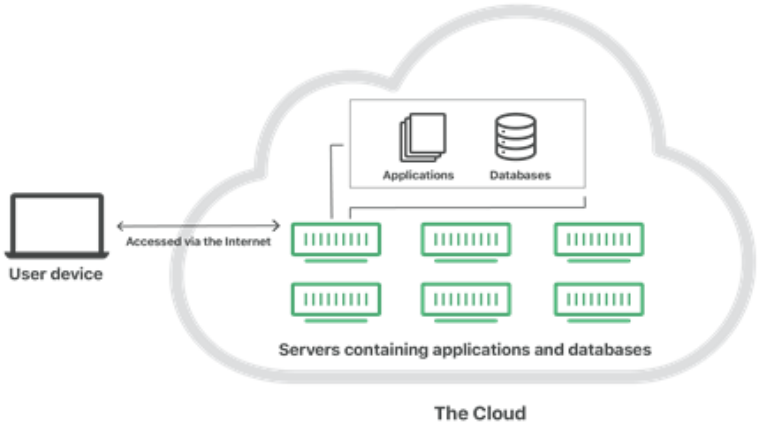


FIGURE 1.3 – le Cloud Computing

L'utilisation du Cloud Computing pour l'apprentissage profond permet d'intégrer et de gérer facilement d'importants ensembles de données destinés à entraîner les algorithmes. Ils offrent également des ressources de calcul permettant l'exécution rapide des algorithmes. Les modèles d'apprentissage profond peuvent alors effectuer des mises à l'échelle efficaces et à moindre coût en utilisant la puissance de traitement des GPU. L'apprentissage profond sur le cloud optimise la distribution des réseaux afin de concevoir, de développer et d'entraîner plus rapidement des applications d'apprentissage profond.

1.5.1 Les Avantages de l'implémentation de l'apprentissage profond sur le cloud

✧ ***Rapidité :***

Les algorithmes d'apprentissage profond sont conçus à des fins d'apprentissage rapide. En utilisant des clusters de GPU et de CPU pour effectuer des opérations matricielles complexes pour des tâches de calcul intensif, les utilisateurs peuvent accélérer l'entraînement des modèles d'apprentissage profond. Ces modèles peuvent être déployés afin de traiter d'importants volumes de données et d'obtenir des résultats de plus en plus pertinents.

✧ ***Evolutivité :***

Les réseaux de neurones de l'apprentissage profond s'adaptent parfaitement à l'exploitation de plusieurs processeurs et à la distribution harmonieuse et efficace des charges de travail entre les différents types et quantités de processeurs. Grâce au large éventail de ressources à la demande disponible sur le cloud, nous pouvons déployer des ressources pratiquement illimitées afin d'amorcer des modèles d'apprentissage profond de toutes tailles.

✧ ***Flexibilité :***

Les frameworks d'apprentissage profond tels que Apache MXNet, TensorFlow, le Cognitive Toolkit de Microsoft, Caffe, Caffe2, Theano, Torch et Keras peuvent être exécutés sur le cloud. Ce qui nous permet d'utiliser l'ensemble de bibliothèques d'algorithmes d'apprentissage profond qui correspond le plus à notre cas d'utilisation, qu'il concerne des appareils Web, mobiles ou connectés. [6]

1.5.2 Les inconvénients du Cloud Computing

✧ ***La fiabilité et la localisation du cloud***

Lorsque le cloud n'est pas hébergé dans les meilleures conditions, la sécurité de nos données n'est pas garantie. Il nous faudra donc bien nous renseigner sur les mesures de chiffrement de nos données ainsi que sur les mesures de sécurité garanties par notre prestataire cloud.

✧ ***La nécessité d'avoir une connexion internet :***

Pour accéder au cloud, nous devons disposer d'une connexion internet pour accéder à nos données et applications. Si ce n'est pas le cas, nous ne serons pas en mesure d'accéder à nos données.

1.6 L'IA et les systèmes embarqués

Un système embarqué est un système électronique et informatique autonome. Son rôle principal est de s'occuper d'une tâche précise. Ils intègrent des dispositifs industriels, des calculateurs miniatures, des interfaces opérateurs, des modules de communication radio et Ethernet, et des ordinateurs portables. Par ailleurs, ce terme désigne aussi bien le Hardware (matériel) que le Software (logiciel). Grâce aux tendances technologiques, le stockage en ligne autrement dit le « Cloud » permet à l'utilisateur du compte d'externaliser ou d'enregistrer de gros volumes de données sur des serveurs distants. Par ailleurs, les microcontrôleurs ne consomment pas beaucoup d'énergie lors de l'utilisation.

L'apprentissage profond et l'apprentissage automatique ont toujours été associés à de gros ordinateurs dotés de processeurs et de GPU rapides, d'une grande taille de RAM ou d'algorithmes en cours d'exécution sur le cloud. Cependant, imaginer effectuer du Machine Learning sur un microcontrôleur alimenté par une seule pile est impossible, mais avec la technologie d'aujourd'hui, l'impossible est désormais possible avec les nouveaux microcontrôleurs.

En ce moment, la question qui se pose est pourquoi utiliser des microcontrôleurs pour l'apprentissage automatique et l'IA ? Et comment peut-il se faire ?

Malgré que les microcontrôleurs disposent de peu d'espace mémoire et de puissance du calcul, ils présentent divers avantages qui l'emportent sur cet inconvénient à savoir :

✓ ***Faible consommation d'énergie :***

En raison de leur petite taille, au détriment de la puissance de traitement, de la mémoire et du stockage, les microcontrôleurs consomment très peu d'énergie et sont efficaces. Cependant, pour alimenter le GPU et les ordinateurs pour l'apprentissage automatique beaucoup de puissance est nécessaire, ce qui entraîne des limitations et des contraintes.

✓ ***Coût :***

Normalement, pour l'apprentissage automatique, Nous devons dépenser quelques milliers dollars pour créer une station de travail d'apprentissage automatique hautes performances. Cependant, avec les microcontrôleurs, nous pouvons facilement en obtenir un à 200 dollars et moins, qui sont également fiables.

✓ ***Flexibilité :***

Les microcontrôleurs sont très courants. Ils sont pratiquement partout autour de nous, comme nos appareils électroménagers, nos jouets, nos voitures, etc. Les possibilités sont infinies lorsque nous apportons l'apprentissage automatique aux microcontrôleurs. Ainsi, nous pouvons ajouter de l'IA à divers appareils sans dépendre de la connectivité réseau qui est limitée par la bande passante, la puissance et la latence élevée.

✓ ***Intimité :***

Pour l'apprentissage automatique, nous devrons enchaîner toutes nos données brutes sur le cloud qui pourraient contenir des informations confidentielles ou privées. Cependant, pour les microcontrôleurs, les utilisateurs n'ont pas à s'inquiéter de ce problème car aucune donnée ne devra quitter l'appareil.

Le déploiement de modèles d'apprentissage automatique (ML) sur des microcontrôleurs est l'un des développements les plus passionnants de ces dernières années, permettant à de petits appareils alimentés par batterie de détecter des mouvements complexes, de reconnaître des sons, de trouver des anomalies dans les données des capteurs et aussi de faire la reconnaissance de formes qui est l'objet de notre travail. Dans ce cadre, nous allons implémenter un modèle de Deep Learning sur le ST Microelectronics STM32F429I-DISC1 qui est une carte de découverte pour

les microcontrôleurs STM32F429. Il inclut tous les éléments nécessaires permettant aux utilisateurs expérimentés et débutants de démarrer le développement d'applications avec des interfaces graphiques utilisateur. Cette carte de développement en particulier est également équipée d'un écran LCD TFT 2.4". Grâce à un nouvel ensemble de solutions d'intelligence artificielle (IA) de ST, nous avons désormais la possibilité de cartographier et d'exécuter des réseaux de neurones artificiels préformés et pré-entrainés sur le vaste portefeuille de microcontrôleurs STM32 .

Le STM32Cube.AI est un pack d'extension de l'outil de configuration et de génération de code STM32CubeMX largement utilisé permettant l'IA sur les microcontrôleurs basés sur STM32 Arm Cortex M.

❖ **La cartographie du réseau de neurones artificiels simplifiée avec le STM32Cube.AI :**

- Interopérable avec les outils de formation d'apprentissage profond populaires.
- Compatible avec de nombreux IDE et compilateurs.
- Offre la possibilité d'utilisation du système FreeRTOS (Temps réel) et des capteurs.
- Permet à plusieurs réseaux de neurones artificiels d'être exécutés sur un seul MCU STM32.
- Prise en charge complète des microcontrôleurs STM32 ultra basse consommation.

Conclusion

Ce chapitre nous a permis d'avoir une vue globale sur l'intelligence artificielle, en se basant sur les principes de base du Machine Learning et du Deep Learning et en faisant la différence entre eux. Nous avons aussi cité les divers domaines d'applications de ces derniers. Nous avons clôturé ce chapitre par le principe d'implémentation du Deep Learning sur le Cloud avec ses avantages et inconvénients et nous avons proposé comme solution l'utilisation du microcontrôleur STM32F4 vu les performances des systèmes embarquées dans le domaine de l'IA.

Dans le chapitre suivant, nous allons voir les méthodes de classifications des images numériques en se basant sur les réseaux de neurones artificiels en particulier convolutifs.

CHAPITRE

2

CNN ET CLASSIFICATION D'IMAGES

Introduction

Pour l'humain, voir des objets et les reconnaitre est tout ce qui est de plus naturel, l'apprentissage s'étant fait précédemment. Mais pour la machine, le processus n'est pas aussi simple. Pour ce faire on applique un algorithme de classification des images qui consiste à attribuer automatiquement une classe à une image. Cela peut s'appliquer sur des objets, animaux, formes, caractères, empreintes et même des visages.

Dans ce chapitre nous allons voir comment utiliser un réseau de neurones artificiels pour la classification des images. Mais avant ça, nous allons d'abord donner un aperçu sur les images numériques et leurs caractéristiques, puis nous aborderons les différentes méthodes de classifications de ces images en se basant sur la méthode des réseaux de neurones artificiels, où nous allons voir l'architecture des RNA, leur fonctionnement et leurs différents types. Par la suite, nous verrons en détail les réseaux de neurones convolutifs, vu qu'il est le type utilisé pour la classification des images.

2.1 Traitement d'Image

Le traitement d'images est une discipline de l'informatique et des mathématiques appliquées qui étudie les images numériques et leurs transformations, dans le but d'améliorer leurs qualités ou d'en extraire de l'information. Citons par exemple les opérations de rehaussement de contraste, élimination du bruit et correction d'un flou.

La compréhension du traitement d'image commence par la compréhension de ce qu'est une image.

2.1.1 Définition de l'image

L'image est une représentation d'une personne ou d'un objet par la peinture, la sculpture, le dessin, la photographie, le film, etc. C'est aussi un ensemble structuré d'informations qui, après l'affichage sur l'écran, ont une signification pour l'œil humain.

Du point de vue mathématique une image est généralement représentée par une fonction bidimensionnelle f(x,y) de brillance analogique continue, définie dans un domaine borné, tel que x et y sont les coordonnées spatiales d'un point de l'image et f est une fonction d'intensité lumineuse et de couleur [7]. Sous cet aspect, l'image est inexploitable par la machine, ce qui nécessite sa numérisation.

2.1.2 Définition de l'image numérique

Le terme d'image numérique désigne, dans son sens le plus général, toute image qui a été acquise, traitée et sauvegardée sous une forme codée représentable par des nombres (valeurs numériques).

La numérisation est le processus qui permet de passer de l'état d'image physique (image optique par exemple) qui est caractérisée par l'aspect continu du signal qu'elle représente (une infinité de valeurs de l'intensité lumineuse par exemple), à l'état d'image numérique qui est caractérisée par l'aspect discret (l'intensité lumineuse ne peut prendre que des valeurs quantifiées en un nombre fini de points distincts). La figure 2.1 montre le processus de numérisation d'une image.

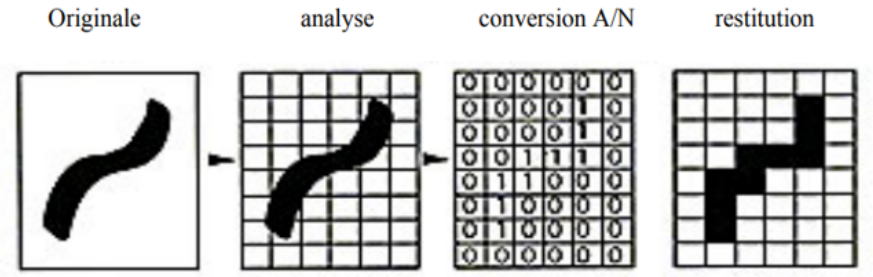


FIGURE 2.1 – Numérisation d'une image.

Mathématiquement, La numérisation d'une image est la conversion de celle-ci en matrice bi-dimensionnelle de valeurs numériques I (n, m) où : (n, m) sont représentées par des coordonnées cartésiennes d'un point de l'image, I (n, m) est le niveau de gris ou de couleur en ce point.

Pour chaque pixel, on code en binaire chacune des 3 valeurs des composantes R, V, B de la couleur du pixel. Le code binaire de l'image est obtenu en indiquant successivement pour chaque pixel le code binaire des 3 composantes. Si on code chaque composante sur 8 bits, chaque pixel sera donc représenté par 24 bits.

La figure 2.2 représente un morceau d'une image couleur codé sur 3 octets.

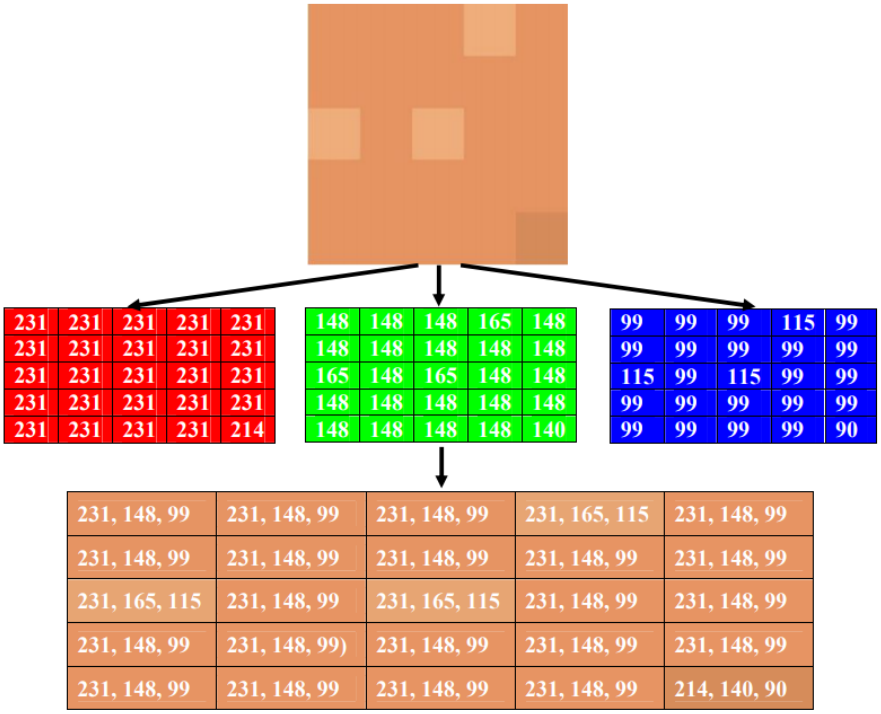


FIGURE 2.2 – Codage sur 3 octets d'un morceau d'image couleur.

C'est cette forme numérique qui permet une exploitation ultérieure par des outils logiciels sur ordinateur.

2.1.3 Le codage des couleurs

Les images binaires :

Les images binaires sont des images de profondeur K=1 bit, donc un pixel peut prendre l'une des valeurs : 0 pour le noir ou 1 pour le blanc. C'est typiquement le type d'images que l'on utilise pour scanner du texte quand celui-ci est composé d'une seule couleur. La figure 2.3 montre un exemple d'une image binaire.

FIGURE 2.3 – Représentation d'une image binaire.

Les images en niveaux de gris :

En général, les images en niveaux de gris sont des images de profondeur k=8 bits, donc chaque pixel peut prendre l'une des valeurs de l'intervalle [0-255], où la valeur 0 représente la brillance minimale (le noir) et 255 la brillance maximale (le blanc). La figure 2.4 représente un exemple d'image en niveaux de gris.

Ce type d'image est fréquemment utilisé pour reproduire des photos en noir et blanc ou du texte. Dans plusieurs applications professionnelles de photographie et d'impression ainsi qu'en médecine et astronomie, 8 bits par pixel n'est pas suffisant, pour cela il existe d'autre types d'images en niveaux de gris de profondeur K=12, K=14 ou K=16 bits.

FIGURE 2.4 – Représentation d'une image en niveaux de gris.

Les images couleurs :

L'espace couleur est basé sur la synthèse additive des couleurs. C'est-à-dire que le mélange entre différentes couleurs (trois, quatre...) donne une couleur. La plupart des images couleurs sont basées sur trois couleurs primaires : Rouge, Vert et Bleu (RVB) ou (RGB en anglais), et utilisent

typiquement 8 bits pour chaque composante de couleur.Donc chaque pixel nécessite 3×8=24 bits pour coder les trois composantes, et chaque composante de couleur peut prendre l'une des valeurs de l'intervalle [0-255]. La figure 2.5 montre un exemple d'une image couleur en prenant en compte de système RGB.

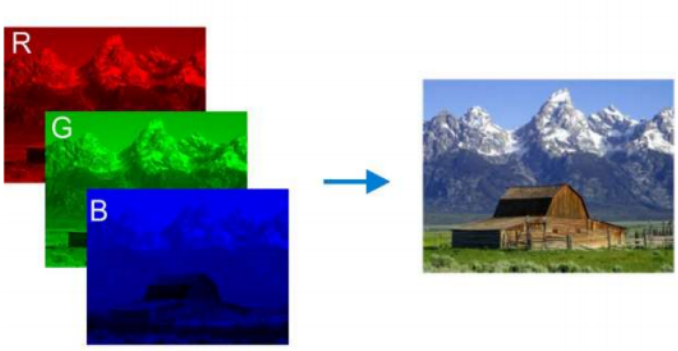


FIGURE 2.5 – Représentation d'une image couleur.

2.1.4 Caractéristique de l'image numérique

L'image est constituée d'un ensemble structuré d'informations. Parmi ces informations nous pouvons citer :

Le pixel : représente le plus petit élément constitutif d'une image. Le mot pixel provient d'une abréviation de l'expression britannique (Picture Element). Il matérialise un point donné (x, y) du plan de l'image. Sa valeur numérique représente une intensité lumineuse, et l'ensemble de ces pixels est contenu dans un tableau à deux dimensions constituant l'image.

Dimension : La dimension est la taille de l'image. Elle se présente sous forme d'une matrice dont les éléments sont des valeurs numériques représentatives des intensités lumineuses (pixels). Le nombre de lignes de cette matrice multipliée par le nombre de colonnes nous donne le nombre total de pixels dans une image.

Résolution : la résolution est la clarté ou la finesse de détails atteinte par un moniteur ou une imprimante dans la production d'images. Sur les moniteurs d'ordinateur, la résolution est exprimée en nombre de pixels par unité de mesure qui est en général le pouce (1 pouce = 2,54cm). On parle alors de pixel/pouce PPP) ou DPI (dots per inch) en anglais, parfois en point par cm. On utilise aussi le mot résolution pour désigner le nombre total de pixels horizontaux et verticaux sur un moniteur. Plus ce nombre est grand, plus la résolution est meilleure.

La luminance : est le degré de luminosité des points de l'image. Elle est définie aussi comme étant le quotient de l'intensité lumineuse d'une surface par l'air apparente de celle-ci. Une bonne

luminance est caractérisée par :

- Des images lumineuses (brillantes).
- Un bon contraste : il faut éviter les images où la gamme de contraste tend vers le blanc ou le noir ; ces images entraînent des pertes de détails dans les zones sombres ou lumineuses.
- L'absence de parasite.

Contraste : C'est l'opposition marquée entre deux régions d'une image, plus précisément entre les régions sombres et les régions claires de cette image. Le contraste est défini en fonction des luminances de deux zones d'images.

Une image contrastée présente une bonne dynamique de la distribution des valeurs de gris sur tout l'intervalle des valeurs possibles, avec des blancs bien clairs et des noirs profonds. Par contre une image peu contrastée a une faible dynamique, la plupart des pixels ayant des valeurs de gris très proches. [8]

Si L1 et L2 sont les degrés de luminosité respectivement de deux zones voisines A1 et A2 d'une image, le contraste est défini par le rapport :

$$C = \frac{L1-L2}{L1+L2}$$

Voisinage : Le plan de l'image est divisé en termes de formes rectangulaires ou hexagonales permettant ainsi l'exploitation de la notion de voisinage. Le voisinage d'un pixel est formé par l'ensemble des pixels qui se situent autour de ce même pixel. On définit aussi l'assiette comme étant l'ensemble de pixels définissant le voisinage pris en compte autour d'un pixel.

On distingue deux types de voisinage :

- **Voisinage à 4 :** On ne prend en considération que les pixels qui ont un coté commun avec le pixel considéré. (Figure 2.6)

- **Voisinage à 8 :** On prend en compte tous les pixels qui ont au moins un point en liaison avec le pixel considéré. (Figure 2.7)

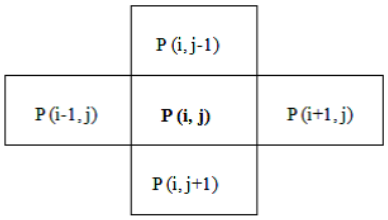


FIGURE 2.6 – Voisinage à 4.

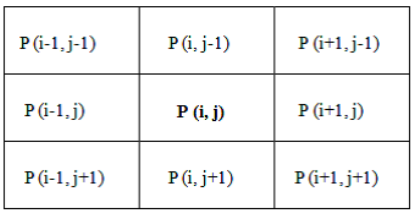


FIGURE 2.7 – Voisinage à 8.

Contours et textures : Les contours représentent la frontière entre les objets de l'image, ou la limite entre deux pixels dont les niveaux de gris représentent une différence significative. Les textures décrivent la structure de ceux-ci. L'extraction de contour consiste à identifier dans l'image les points qui séparent deux textures différentes. [10]

Le bruit : Un bruit (parasite) dans une image est considéré comme de brusques variations de l'intensité d'un pixel par rapport à ses voisins. Il peut provenir de différentes causes :

- Capacité des capteurs ou une mauvaise utilisation de ces derniers.
- Problème de prise de vue, mise au point.
- Défaut de système de numérisation.

Histogramme de l'image : Un histogramme est un graphique statistique permettant de représenter la distribution des intensités des pixels d'une image. C'est-à-dire le nombre de pixels pour une intensité lumineuse. Par convention un histogramme représente le niveau d'intensité en abscisse en allant du plus foncé (à gauche) au plus clair (à droite).

La figure 2.8 présente un histogramme d'une image Avec : H(x) est le nombre de pixels dont le niveau de gris est égal à x. [10]

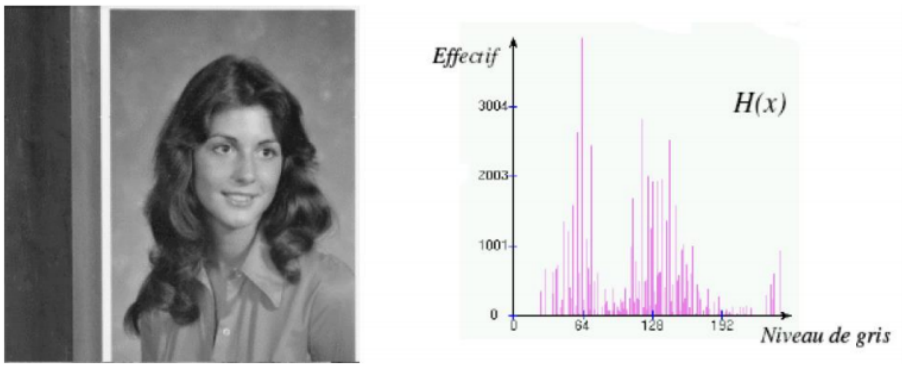


FIGURE 2.8 – Représentation d'un histogramme d'une image.

2.1.5 Quelques techniques fondamentales en traitement d'images

Une panoplie de traitements peut être appliquée à l'image numérique. Dans ce qui suit, nous donnons quelques exemples de ces traitements.

2.1.5.1 Acquisition de l'image :

L'acquisition d'images est une mesure spatiale d'une interaction entre une onde et de la matière. L'onde est émise par une source et reçu par un capteur. La matière occupe de l'espace et possède une masse. Elle a pour objet de passer de la scène physique à une forme numérique observée.

2.1.5.2 La segmentation :

C'est une procédure permettant de partitionner l'image en ses constituants ou objets. La segmentation automatique est la tâche la plus difficile en traitement d'images. Plus la segmentation est meilleur plus l'étape de reconnaissance d'objets est réussite.

2.1.5.3 La reconnaissance :

Est le traitement qui affecte une étiquette (exemple : route, voiture) à un objet en se basant sur ses descripteurs. La base de connaissance contient la connaissance du domaine du problème en cours du traitement. Dans son aspect le plus simple, elle peut consister en coordonnées de l'objet à traiter, ceci permet de réduire l'espace de recherche. Comme elle peut être complexe contenant toutes les défaillances que peut présenter un produit manufacturé.

2.1.5.4 La classification :

La classification d'images est un problème fondamental en vision par ordinateur, qui a de nombreuses applications concrètes. Le but est de construire un système capable d'assigner correctement une catégorie à n'importe quelle image en entrée. Nous allons voir plus en détail les différentes méthodes de classification existantes dans les sous titres suivants.

2.2 Méthodes de classification des images numériques

La classification d'image est le fait de classifier des images selon une description donnée au préalable (Titre, tag, descriptions), ou selon les formes existantes dans l'image par exemple, si l'image contient une certaine forme ou un certain objet, ou bien selon des scènes (villes, montagnes, campagne, etc).

De nombreuses méthodes classiques ont été développées. Elles peuvent être réparties en deux grandes catégories : les méthodes supervisés (classement) et non supervisé (clustering).

2.2.1 Méthodes de classification supervisée

La classification supervisée consiste à définir des règles permettant de classer des données dans des classes et catégories déjà prédéfinies à partir de variables qualitatives ou quantitatives carac-

térisant ces objets. On dispose au départ d'un échantillon dit d'apprentissage dont le classement est connu. Cet échantillon est utilisé pour l'apprentissage des règles de classification.

Principe :

Le principe général de cette méthode consiste à classifier un ensemble de données constitué de n objets, sachant que ces n objets sont étiquetés. Ce système de classification va être conçu en se basant sur un ensemble d'apprentissage dont on connait la classe de tout exemple. C'est-à-dire on cherche à prédire si un élément « xi » de la base de données, décrit par un ensemble de descripteur appartient ou non à une classe « cj » parmi N classes.

Donc l'objectif est de chercher à prédire la classe de chaque nouvelle donnée en utilisant les données de la base d'apprentissage.

2.2.2 Méthodes de classification non supervisée

Cette méthode en revanche, ne requière pas un apprentissage au préalable. C'est-à-dire ne nécessite aucune tâche préalable d'étiquetage manuel.

Dans ce cas, c'est au programme d'extraire des caractéristiques spécifiques dans des images et les regrouper selon la similarité, on parle alors de Clustering, qui consiste à construire une collection d'objets similaire au sein d'un même groupe appelé cluster sans l'aide d'étiquetage.

Principe :

Contrairement à la classification supervisée on ne possède pas des connaissances à priori sur les classes prédéfinies des éléments. La division des objets dans les différents groupes (clusters) est réalisée en se basant sur le calcul de similarité entre les éléments. L'objectif des méthodes du clustering est de grouper des éléments proches dans un même groupe de manière à ce que deux données d'un même groupe soient le plus similaires possible et que deux éléments de deux groupes différents soient le plus dissemblables possible.

2.3 Quelques algorithmes de classification

2.3.1 Algorithme des k-means

K-means est un algorithme non supervisé de clustering non hiérarchique. Il permet de regrouper en clusters distincts les observations du data set. Ainsi, les données similaires se retrouveront dans un même cluster. Par ailleurs, une observation ne peut se retrouver que dans un cluster à la fois (exclusivité d'appartenance). Une même observation ne pourra pas donc appartenir à deux clusters différents. Le regroupement se fait en minimisant la somme des distances entre chaque

objet et le centre de gravité du groupe ou du cluster.

k-means est un algorithme itératif qui minimise la somme des distances entre chaque individu et les centres des clusters appelés centroïde. Le choix initial des centroïdes conditionne le résultat final.

L'algorithme comporte trois étapes :

1. **Initialisation :** une fois le nombre de groupes, k choisi, k centroïdes sont établis dans l'espace de données, par exemple en les choisissant aléatoirement.
2. **Affectation des objets aux centroïdes :** chaque objet des données est affecté à son centroïde le plus proche.
3. **Mise à jour des centroïdes :** La position du centroïde de chaque groupe est mise à jour en prenant comme nouveau centroïde la position moyenne des objets appartenant audit groupe.

Répétez les étapes 2 et 3 jusqu'à ce que les centroïdes ne bougent pas ou se déplacent en dessous d'une distance seuil à chaque étape. La figure 2.9 représente un exemple de classification par l'algorithme des K-means.

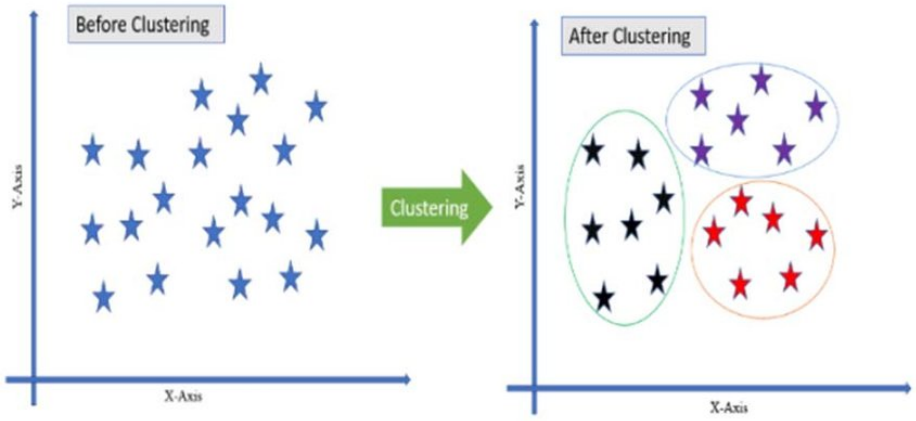


FIGURE 2.9 – Classification par l'algorithme des K-means.

2.3.2 K plus proches voisins (KNN)

L'algorithme des K plus proches voisins est un algorithme d'apprentissage supervisé. Son principe est de classer les exemples dont la classe est inconnue, en se basant sur leur similarité avec un ensemble d'échantillons de référence. La similarité est déterminée par un calcul de distance. La distance euclidienne décrite dans l'équation suivante est la plus utilisée :

$$D\left(X,u\right)=\sqrt{\sum_{i=1}^{n}\left(x_i-u_i\right)^2}$$

Où :
D : est la distance euclidienne.
$X = (x_1, x_2, \ldots, x_n)$: est l'élément à classifier.
u : est un élément de la base de référence.
n : étant la dimension du vecteur de caractéristiques.

2.3.3 Les Arbres de décision

L'arbre de décision est un outil de classification et prédiction. Sa popularité repose sur sa simplicité. Il emploie une représentation hiérarchique de la structure des données sous forme des séquences de décisions, en vue de la prédiction d'un résultat ou d'une classe.

C'est une représentation graphique de la procédure de classification, où chaque observation, qui doit être attribuée à une classe, est décrite par un ensemble de variables, qui sont testées dans les nœuds de l'arbre. Les tests s'effectuent dans les nœuds internes et les décisions sont prises dans les nœuds feuilles. Pour classifier un nouvel objet, on suit le chemin partant de la racine à une feuille en effectuant les différents tests d'attributs à chaque nœud.

La figure 2.10 représente un arbre qui peut prendre une décision pour un joueur d'aller jouer au tennis ou pas en fonction de ciel, de l'humidité et du vent.

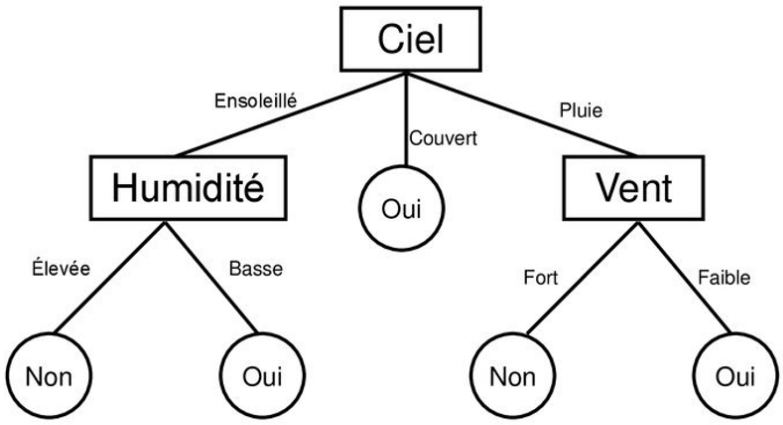


FIGURE 2.10 – Exemple d'un arbre de décision.

2.3.4 Les réseaux de neurones

Les réseaux de neurones, communément appelés des réseaux de neurones artificiels (RNA) sont des imitations simples des fonctions d'un neurone dans le cerveau humain pour résoudre des problématiques d'apprentissage de la machine (Machine Learning). Les RNA sont un système d'apprentissage supervisé constitué d'un grand nombre d'éléments simples, appelés neurones ou

perceptrons. Chaque neurone peut prendre des décisions simples et remonte ses décisions vers d'autres neurones, organisés en couches interconnectées.

Au cours de notre travail, afin de faire une classification de notre ensemble de données nous avons opté pour la méthode des réseaux de neurones artificiels. Nous allons voir plus en détail sur les RNA dans ce qui suit.

2.4 Les réseaux de neurones artificiels

Les réseaux de neurones artificiels est un ensemble de processeurs élémentaires, fortement connectés, fonctionnant en parallèle. Chaque processeur (neurone) calcule une sortie à la base des entrées qu'il reçoit. Mathématiquement, un RNA est un graphe orienté, dans lequel chaque nœud (neurone) réalise des calculs élémentaires. Il est exprimé généralement par une fonction sigmoïd :

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

La conception des réseaux de neurones artificiels s'appuie sur la structure des neurones biologiques du cerveau humain où les synapses assurent les connexions avec les autres neurones, les dendrites sont les entrées, les axones sont les sorties, et enfin le noyau qui active les sorties selon les stimulations en entrées.

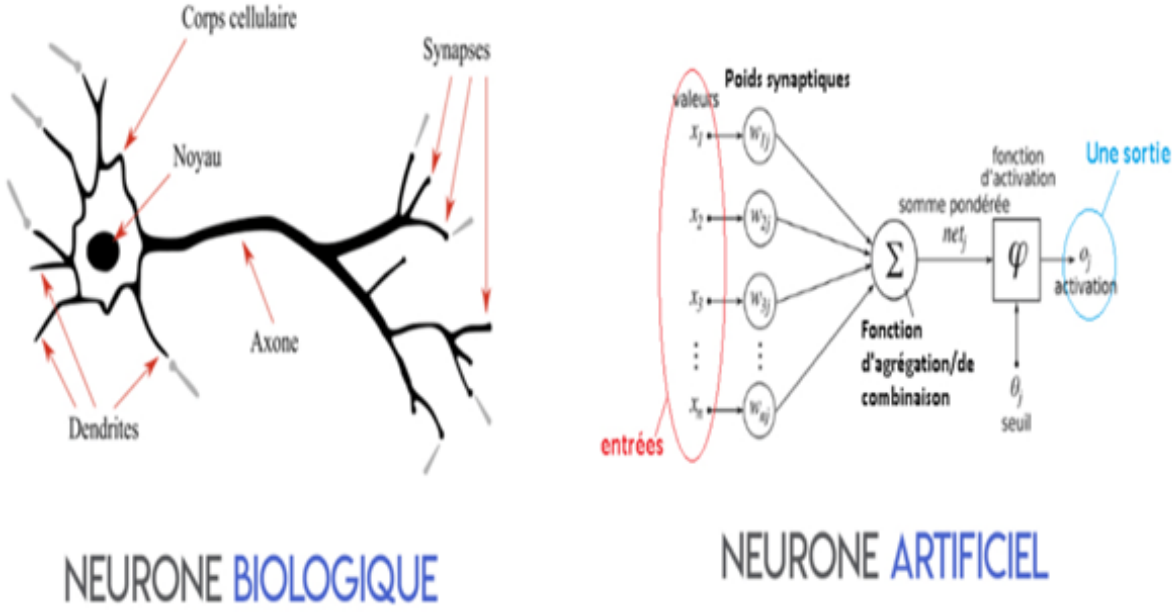


FIGURE 2.11 – Neurone biologique et neurone artificiel.

D'après la figure 2.11, on déduit qu'un neurone formel (artificiel) n'est qu'une représentation mathématique d'un neurone biologique. Ainsi, les entrées et la sortie d'un neurone artificiel,

correspondent aux dendrites et axone dans le neurone biologique. La figure 2.12 présente les différentes correspondances entre ces deux derniers.

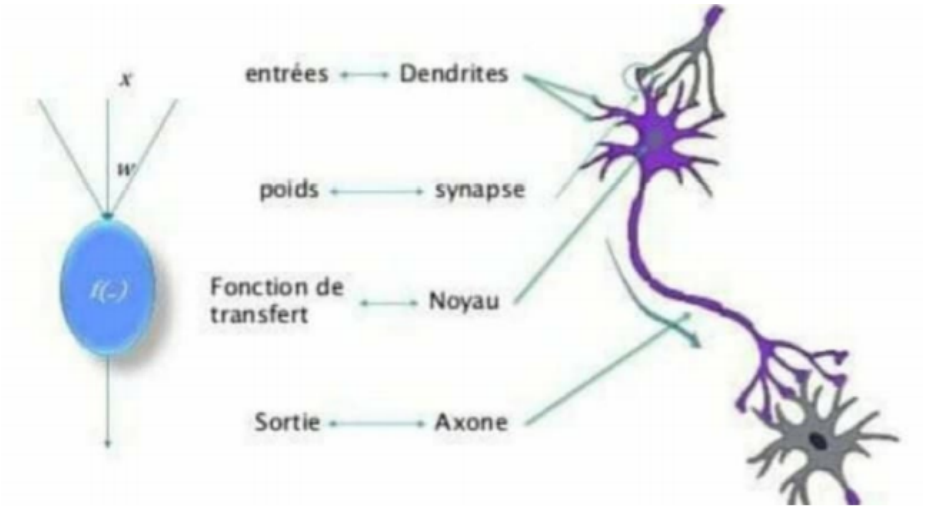


FIGURE 2.12 – Correspondance entre neurone biologique et neurone formel.

❖ **Synthèse des Correspondances entre neurone biologique et neurone formel :**

NEURONE BIOLOGIQUE	**NEURONE ARTIFICIEL**
SYNAPES	Poids de connexions
AXONES	Signal sortie
DENDRITES	Signal d'entrée
NOYAU	Fonction d'activation

TABLE 2.1 – La correspondance entre neurone biologique et neurone formel.

Les RNA sont en passe de devenir une composante incontournable de l'aide à la décision. L'analyse neuronale arrive aujourd'hui à un degré de maturité lui permettant de traiter un bon nombre de problèmes. De ce fait, ils sont largement appliqués pour résoudre des problèmes caractérisés par une relation entrée-sortie de la donnée d'information, tels que :

- La reconnaissance d'images, de vidéos et de formes.
- Les classifications de textes ou d'images.
- Identification et détection d'objets.
- Prédiction de données.
- Filtrage d'un ensemble de données.

2.4.1 Les principales composantes d'un réseau de neurones

Le réseau de neurones est constitué des composants principaux suivants :

➢ **Neurones : ensemble de fonctions**

Ils prennent une donnée d'entrée et produisent une donnée de sortie. Un certain nombre de neurones sont groupés en couches (ou layers). Tous les neurones du même groupe remplissent un type de fonction similaire.

Les neurones d'entrée reçoivent des données d'entrée, les traitent et les transmettent aux neurones de la couche suivante. Les neurones cachés prennent les données de sortie des précédents neurones en entrée, calculent de nouvelles données de sortie et les transmettent à des couches successives.

Dans un réseau de neurones à plus de 3 couches, les neurones de la dernière couche cachée (hidden layer) transmettent les données de sortie en entrée des neurones de la couche de sortie (output layer). À partir de là les neurones de la couche de sortie produisent les données de sortie finales.

➢ **Couches : groupement de neurones :**

Les couches (ou layers) contiennent des neurones et aident à faire circuler l'information. Il existe au moins deux couches dans un réseau de neurones : la couche d'entrée (input layer) et la couche de sortie (output layer). Les couches, autres que les couches d'entrée et de sortie, sont appelées les couches cachées (ou hidden layers).

➢ **Poids et biais : valeurs numériques**

Les poids et biais sont des variables du modèle qui sont mises à jour pour améliorer la précision du réseau. Un poids est appliqué à l'entrée de chacun des neurones pour calculer une donnée de sortie. Les RNA mettent à jour ce poids de manière continue. Il existe donc une boucle de rétro-action mise en œuvre dans la plupart des réseaux de neurones.

Les biais sont également des valeurs numériques qui sont ajoutées une fois que les poids sont appliqués aux valeurs d'entrée. Les poids et les biais sont donc en quelque sorte des valeurs d'auto-apprentissage de nos RNA.

2.4.2 Structure d'un RNA

Un réseau de neurones est composé en général d'une succession de couches dont chacune prend ses entrées sur les sorties de la précédente. Chaque couche (i) est composée de N_i neurones, prenant leurs entrées sur les N_{i-1} neurones de la couche précédente. À chaque neurone est associé un poids (W) correspondant à la force de connexion. Les N_{i-1} neurones sont multipliés par ce poids, puis additionnés par les neurones de niveau i, ce qui est équivalent à multiplier le vecteur d'entrée par une matrice de transformation. Disposer les différentes couches d'un RN l'une derrière l'autre reviendrait à mettre en cascade plusieurs matrices de transformation. La figure 2.13 présente une vue simplifiée de la structure de réseau de neurone.

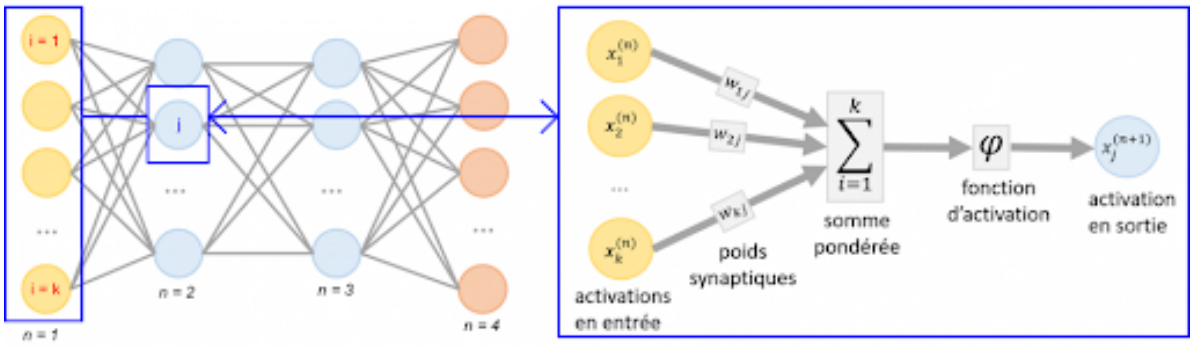


FIGURE 2.13 – Une vue simplifiée de la structure du réseau de neurone.

2.4.3 Fonctionnement d'un RNA

Un réseau neuronal se compose de nombreuses unités de traitement homogènes (les neurones), fortement interconnectées par des liens d'intensité variable. L'activité de l'unité unique est simple et la puissance du modèle réside dans la configuration des connexions (topologie et intensité). À partir des unités d'entrée, auxquelles les données sont envoyées pour résoudre un problème, le calcul se propage en parallèle dans le réseau jusqu'aux unités de sortie, qui fournissent le résultat. Un réseau de neurones n'est pas programmé pour exécuter une certaine activité unique, mais entraîné (en utilisant un algorithme d'apprentissage automatique) au moyen d'une série d'exemples de la réalité à modéliser.

Au sein d'un réseau de neurones artificiels, le traitement de l'information suit toujours la même séquence : les informations sont transmises sous la forme de signaux aux neurones de la couche d'entrée, où elles sont traitées. À chaque neurone est attribué un « poids » particulier, et associé à la fonction dite de transfert. Ce poids permet de déterminer quelles informations peuvent entrer dans le système.

À l'étape suivante, une fonction dite d'activation est associée à une valeur seuil calcule et pondère la valeur de sortie du neurone. En fonction de cette valeur, un nombre plus ou moins grand de neurones sont connectés et activés. Cette connexion et cette pondération dessinent un algorithme qui fait correspondre un résultat à chaque entrée. Chaque nouvelle itération permet d'ajuster la pondération et donc l'algorithme de façon à ce que le réseau donne à chaque fois un résultat plus précis et fiable.

Notamment Le fonctionnement du neurone peut être résumé par la formule mathématique suivante : [11]

$$Sortie = A + (b + \sum_{i=1}^{n} x_i \times w_i)$$

Où :

• A : est une fonction d'activation, qui prend en entré le résultat de la somme des entrées pondérées du perceptron et du biais, et renvois en sortie une valeur dite d'activation.

• b : Il s'agit d'un biais, c'est-à-dire, une constante de correction utilisée afin d'affiner les résultats en sortie du neurone.

• x_i : (i allant de 1 à n) correspond à l'ensemble des entrées (inputs) reçues par le perceptron, celles-ci, pouvant être le résultat en sortie d'autres neurones, ou des données brutes reçues en entrée.

• w_i : (i allant de 1 à n) est l'ensemble des poids associés aux entrées, où chaque entrée (input) se voit associée un poids unique.

L'objectif général d'un RNA est de trouver la configuration des poids de connexion entre neurones pour qu'il associe à chaque configuration d'entrée, une réponse adéquate. L'utilisation d'un RNA se fait en deux temps. Tout d'abord une phase d'apprentissage qui est chargée d'établir des valeurs pour chacune des connexions du réseau. Puis, une phase d'utilisation proprement dite, où l'on présente au réseau une entrée et où il nous indique en retour sa sortie calculée.

➢ **Les fonctions d'activation :**

Une fonction d'activation (ou fonction de transfert) est une équation mathématique qui détermine la sortie de chaque élément (perceptron ou neurone) dans le réseau neuronal. Elle sert à convertir le résultat de la somme pondérée des entrées d'un neurone en une valeur de sortie et associer un état au neurone (actif ou non actif), selon les entrées qu'elle reçoit.

Le biais b joue un rôle de seuil, quand le résultat de la somme pondérée dépasse ce seuil, l'argument de la fonction de transfert devient positif ou nul, dans le cas contraire il est considéré

négatif. Si le résultat de la somme pondérée est :

- En dessous du seuil : le neurone est considéré comme non-actif.
- Aux alentours du seuil : le neurone est considéré en phase de transition.
- Au-dessus du seuil : le neurone est considéré comme actif.

Dans ce dernier cas la fonction d'activation est activée et une valeur de sortie est calculée. Cette valeur de sortie est ensuite transmise aux couches suivantes ou précédentes (en fonction de la complexité du réseau), ce qui peut aider les réseaux de neurones à modifier le poids de leurs neurones.

Il y a plusieurs types de fonctions de transfert qui peuvent être utilisé dans les RNA. Les fonctions d'activation souvent utilisées sont :[11]

❖ **La fonction Sigmoïde (logistique) :** Définie par la formule suivante :

$$\forall x \in R \; \exists y \in [0,1] \quad tel\ que \quad y = S(x) = \frac{1}{1+\exp(-x)}$$

Elle produit une courbe en forme de S (Figure 2.14). Bien que de nature non linéaire, il ne tient toutefois pas compte des légères variations des entrées, ce qui entraîne des résultats similaires.

❖ **La fonction tangente hyperbolique (tanh) :** Définie par :

$$\forall x \in R \; \exists y \in [-1,1] \quad tel\ que \quad y = \tanh(x) = \frac{1-\exp(-2x)}{1+\exp(-2x)}$$

Il s'agit d'une fonction supérieure comparée à Sigmoid. Cependant, elle rend moins bien compte des relations et elle est plus lente à converger (Figure 2.14).

❖ **La fonction d'unités linéaire rectifiées (ReLu) :** Définie par :

$$\forall x \in R \; \exists y \in R \quad tel\ que \quad y = ReLu(x) = Max(0,x)$$

En d'autres termes : Si x $<$ 0 alors ReLu(x) = 0 Et Si x $>$ 0 alors ReLu(x) = x

Cette fonction converge plus rapidement(Figure 2.14), optimise et produit la valeur souhaitée plus rapidement. C'est la fonction d'activation la plus populaire utilisée dans les couches cachées.

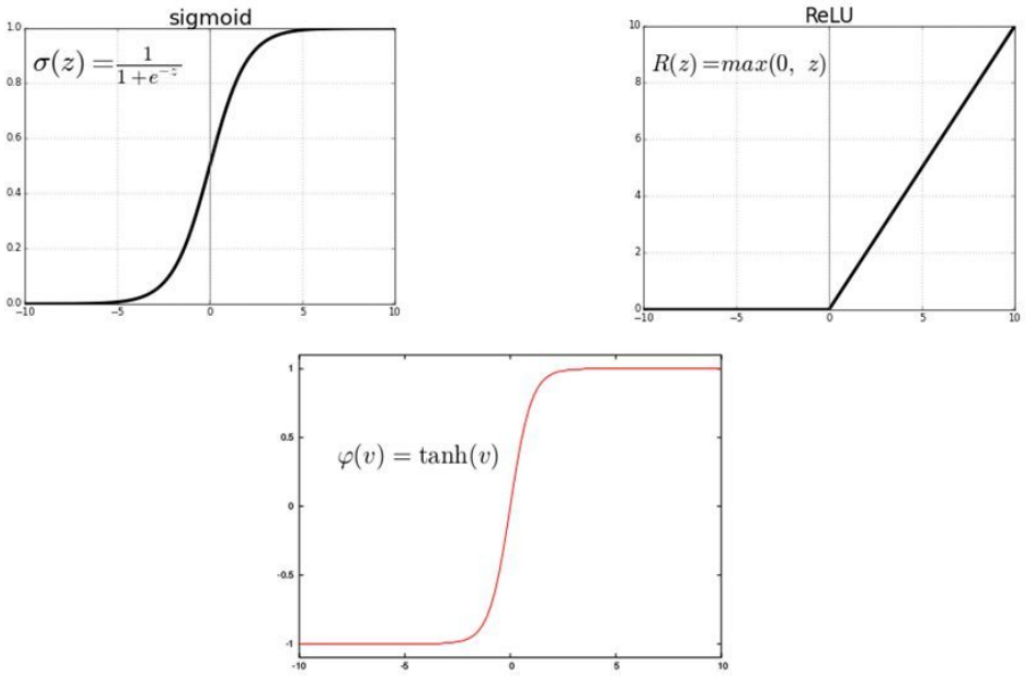


FIGURE 2.14 – Les fonctions d'activation.

➢ **La fonction de coût :**

Les fonctions de coût, permettent d'évaluer l'écart entre les résultats attendus pour une entrée proposées au réseau de neurones, et la prédiction de ce dernier. Il est à noter qu'il en existe un certain nombre de variantes, qui selon l'architecture du réseau ou encore le type de problématique abordée peut plus ou moins convenir. Le réseau va s'adapter jusqu'à ce que l'erreur entre la valeur prédite et la valeur attendue, soit négligeable.

➢ **La Descente de Gradient :**

La Descente de Gradient, (ou Gradient Descent en anglais) est l'un des algorithmes les plus importants de tout le Machine Learning et de tout le Deep Learning. Il s'agit d'un algorithme d'optimisation extrêmement puissant qui permet d'entraîner les modèles de régression linéaire, régression logistique ou encore les réseaux de neurones.

C'est un algorithme d'optimisation, qui permet de minimiser la valeur de sortie d'une fonction pondérée, en permettant d'altérer pas à pas les valeurs des variables de cette fonction, pour aboutir à un minimum local ou global, et cela, en minimisant l'erreur. C'est à dire, notre réseau de neurones commence à se rapprocher petit à petit de la valeur attendue en sortie.

➢ **Forward propagation et Back propagation :**

Ils sont à l'origine de la faculté d'apprentissage des réseaux de neurones. Ils interviennent lors de la phase d'entraînement de notre modèle. Afin d'aboutir à des résultats qui correspondent aux attentes spécifiées, les deux méthodes se comportent de façons différentes.

La forward propagation, c'est le fait de circuler les données de la première couche jusqu'à la dernière, afin de produire une sortie. La back propagation intervient couche par couche, en commençant par la couche de sortie, puis, remonter jusqu'à la toute première, en indiquant aux RNA les ajustements à apporter aux valeurs de l'ensemble de ses poids et des biais.

2.4.4 Les différents types de réseaux de neurones artificiels

Un réseau de neurone peut prendre des structures différentes en fonction de la méthode d'apprentissage utilisée et de la complexité de l'objectif recherché. Les RNA les plus populaires sont :

2.4.4.1 Réseaux de neurones à propagation avant :

Un réseau de neurones artificiels à propagation avant ne peut transmettre l'information que dans un sens unique de traitement (Figure 2.15). Cela signifie tout simplement que la donnée traverse le réseau d'entrée à la sortie sans retour en arrière de l'information.

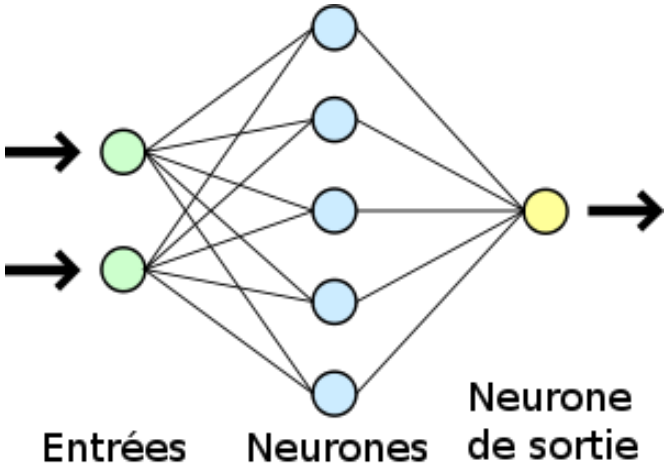


FIGURE 2.15 – Réseaux de neurones à propagation avant.

Typiquement, dans la famille des réseaux à propagation avant, on distingue les réseaux monocouches (perceptron simple) et les réseaux multicouches (perceptron multicouche).

Le perceptron simple est dit simple parce qu'il ne dispose que de deux couches : la couche en entrée et la couche en sortie. Le traitement de la donnée dans ce réseau se fait entre la couche

d'entrée et la couche de sortie qui sont toutes reliées entre elles. Le réseau intégral ne dispose ainsi que d'une matrice de poids.

Dans le perceptron multicouche (Figure 2.16), les neurones sont arrangés par couche. Chaque neurone d'une couche est connecté à tous les neurones de la couche suivante. Les connexions de celle-ci, se font qu'avec les neurones des couches avales, il n'y a pas de connexion entre les neurones d'une même couche.

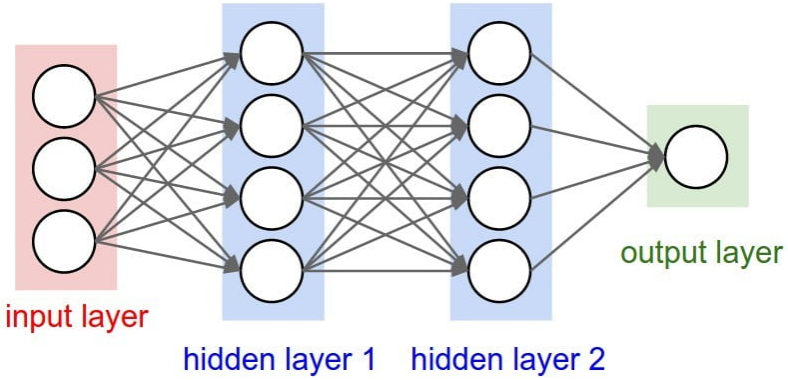


FIGURE 2.16 – Réseau multicouches.

2.4.4.2 Réseaux de neurones récurrents

Les Réseaux de Neurones récurrents traitent l'information en cycle. Ces cycles permettent au réseau de traiter l'information plusieurs fois en la renvoyant à chaque fois au sein du réseau. Du coup, on fait passer l'information dans des boucles de rétroaction, et la faire revenir vers une couche précédente, ce qui permet au système de se constituer une mémoire.

Un réseau de neurones récurrents se compose de trois couches, avec des connexions qui retournent de la couche intermédiaire à la couche d'entrée et de la sortie à l'entrée de la couche intermédiaire comme le montre la figure 2.17.

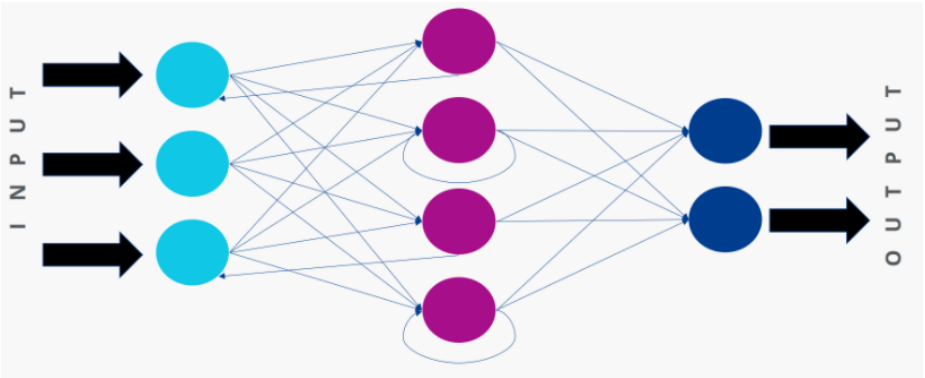


FIGURE 2.17 – Réseau de neurones récurrents.

La force des Réseaux de neurones récurrents réside dans leur capacité de prendre en compte des informations contextuelles suite à la récurrence du traitement de la même information. Cette dynamique auto-entretient le réseau. Les RNC sont utilisés par exemple en matière de reconnaissance vocale, de traduction et aussi de reconnaissance de l'écriture manuscrite.

2.4.4.3 Réseaux de neurones convolutifs :

Les réseaux de neurones convolutifs ou CNN pour Convolutional Neural Network en anglais, sont des réseaux de neurones multicouches spécialisés dans la classification des images et la reconnaissance de formes. C'est le type de RNA que nous allons utiliser dans le cadre de notre projet afin de concevoir un modèle capable de classifier des images pour la reconnaissance de chiffres manuscrits.

Nous allons voir plus en détail sur l'architecture et le fonctionnement des CNN dans ce qui suit.

2.5 Les réseaux de neurones convolutifs

Les réseaux de neurones convolutifs (CNN) ont une méthodologie similaire à celle des méthodes traditionnelles d'apprentissage supervisé. Ils reçoivent des images en entrée, détectent les caractéristiques (features) de chacune d'entre elles et entraînent un classifieur dessus. Ce qui fait leur force, est qu'ils apprennent les features de chaque image par eux même [12]. De plus, l'architecture spécifique du réseau permet d'extraire des features de différentes complexités, des plus simples au plus sophistiquées. Ces réseaux reposent sur des filtres de convolution (matrices numériques). Les filtres sont appliqués aux entrées avant que celles-ci ne soient transmises aux neurones.

2.5.1 Fonctionnement des CNN

Les CNN comportent deux parties bien distinctes. En entrée, une image est fournie sous la forme d'une matrice de pixels. Elle a deux dimensions pour une image en niveaux de gris. La couleur est représentée par une troisième dimension, de profondeur 3 pour représenter les couleurs fondamentales [Rouge, Vert, Bleu]. La première partie d'un CNN est la partie convolutive à proprement parler. Elle fonctionne comme un extracteur de caractéristiques des images. Une image est passée à travers une succession de filtres, ou noyaux de convolution, créant de nouvelles images appelées cartes de convolutions. Certains filtres intermédiaires réduisent la résolution de l'image par une opération de maximum local (fonction de max pooling).

Au final, les cartes de convolutions sont mises à plat et concaténées en un vecteur de caractéristiques, appelé code CNN. Ce code CNN en sortie de la partie convolutive est ensuite branché

en entrée d'une deuxième partie, constituée de couches entièrement connectées (perceptron multicouche). Le rôle de cette partie est de combiner les caractéristiques du code CNN pour classer les pixels d'image ou les images elles-mêmes. La sortie est une dernière couche comportant un neurone par catégorie. Les valeurs numériques obtenues sont généralement normalisées entre 0 et 1, de somme 1 pour produire une distribution de probabilité sur les catégories (voir figure 2.18).

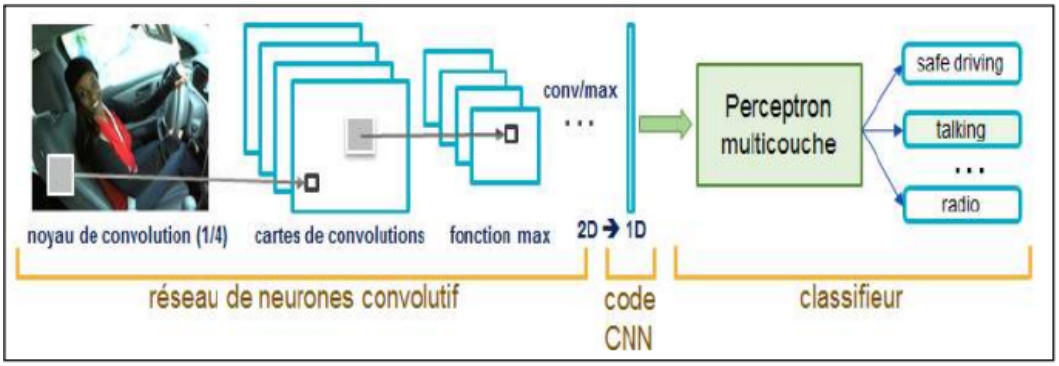


FIGURE 2.18 – Architecture standard d'un réseau de neurone convolutionnel.

2.5.2 Architecture globale d'un CNN

Comme le montre la figure 2.19, un CNN est composé de quatre types de couches : la couche de convolution, la couche de pooling, la couche de correction ReLU et la couche fully connected.

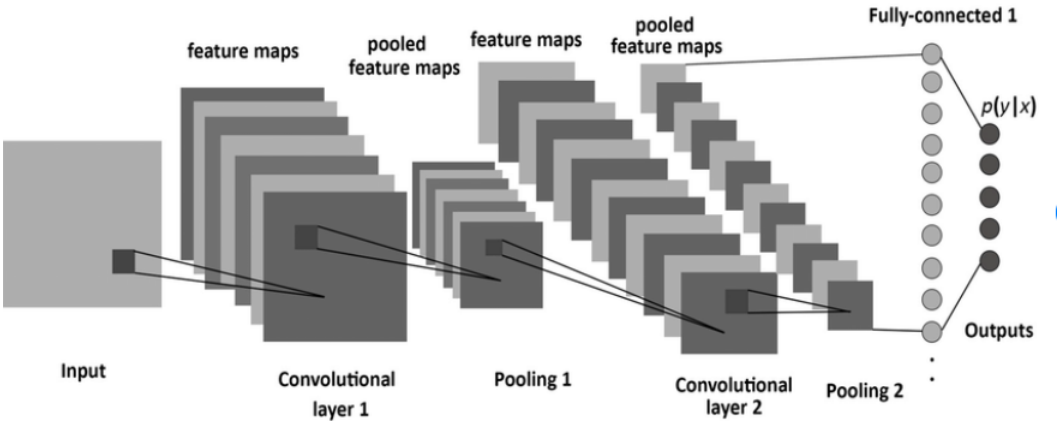


FIGURE 2.19 – Architecture d'un CNN.

2.5.2.1 La couche de convolution :

La couche de convolution est la composante clé des réseaux de neurones convolutifs, et constitue toujours au moins leur première couche. Son but est d'extraire les caractéristiques dans les images reçues en entrée et produit le résultat sous forme d'une carte de caractéristiques.

Pour cela, on réalise un filtrage par convolution. Le principe est de faire glisser une fenêtre représentant la caractéristique (feature) sur l'image, et de calculer le produit de convolution entre la caractéristique et chaque portion de l'image balayée. Une caractéristique est alors vue comme un filtre, les deux termes sont équivalents dans ce contexte. La couche de convolution reçoit donc en entrée plusieurs images, et calcule la convolution de chacune d'entre elles avec chaque filtre. Les filtres correspondent exactement aux caractéristiques que l'on souhaite retrouver dans les images. On obtient pour chaque paire (image, filtre) une carte d'activation, ou cartes de caractéristiques (features map), qui nous indique où se situent les caractéristiques dans l'image. Plus la valeur est élevée, plus l'endroit correspondant dans l'image ressemble à la caractéristique. La figure 2.20 montre un exemple d'application de l'opération de convolution sur une image de 5*5 pixels.

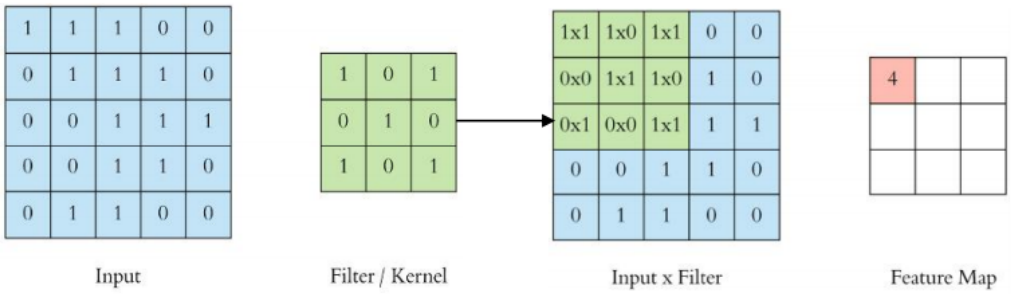


FIGURE 2.20 – Opération d'une convolution sur une image de 5*5 pixels.

2.5.2.2 La couche de pooling :

Cette couche abaisse les données d'image extraites par les couches convolutionnelles pour réduire la dimensionnalité de la carte des caractéristiques afin de diminuer le temps de traitement. Une couche pooling permet de réduire la taille de la matrice d'entrée, d'augmenter la vitesse tout en gardant les informations les plus importantes. Elle est souvent placée entre deux couches de convolution. Elle reçoit en entrée plusieurs feature maps, et applique à chacune d'entre elles l'opération de pooling.

Il existe deux types de pooling : [13]

• Le Maximum Pooling : divise la taille de l'entrée par 2 en prenant le maximum des carrés de taille 2 x 2 pixels, comme le montre la figure 2.21.

• L'Average Pooling : divise la taille de l'entrée par 2 en calculant la moyenne des carrés de taille 2 x 2 pixels.

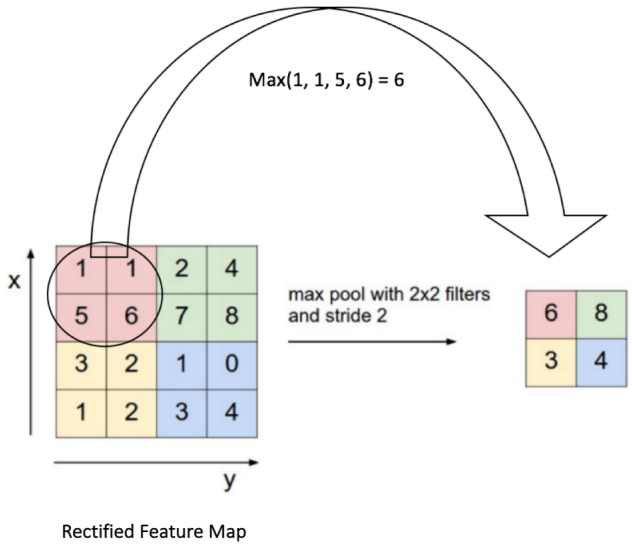


FIGURE 2.21 – Exemple d'application de l'opération Maximum Pooling.

Les couches de Maximum Pooling sont essentielles après des couches de convolutions, afin de pouvoir reconnaître des objets plus grands, et donc plus complexes. Souvent le maximum pooling est le plus utilisé.

2.5.2.3 La couche Relu :

ReLU (Rectified Linear Units) est une fonction réelle non linéaire définie par : ReLU(x) = max (0, x). Elle permet d'augmenter les propriétés non linéaires de la fonction de décision et de l'ensemble du réseau sans affecter les récepteurs de la couche de convolution. Cette fonction force les neurones à retourner des valeurs positives, elle remplace donc toutes les valeurs négatives reçues en entrées par des zéros. La figure 2.22 montre un exemple d'application de la fonction ReLU à une matrice.

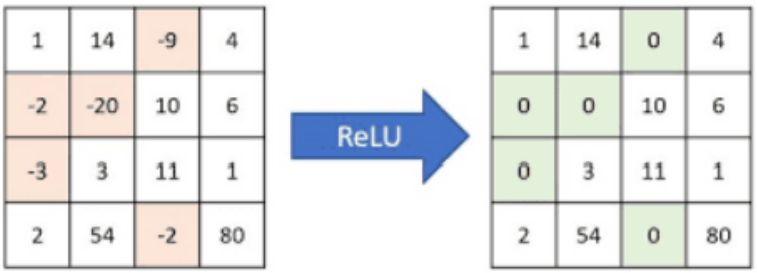


FIGURE 2.22 – fonctionnement de la fonction Relu.

2.5.2.4 La couche fully conneceted :

Elle constitue la dernière couche de tous les types de réseau de neurones, convolutif ou non. Elle n'est donc pas caractéristique d'un CNN. Ce type de couche reçoit un vecteur en entrée

et produit un nouveau vecteur en sortie. Pour cela, elle applique une combinaison linéaire puis éventuellement une fonction d'activation aux valeurs reçues en entrée.

La dernière couche fully-connected permet de classifier l'image en entrée du réseau. Elle renvoie un vecteur de taille N, où N est le nombre de classes dans notre problème de classification d'images. Chaque élément du vecteur indique la probabilité pour l'image en entrée d'appartenir à une classe.

La figure 2.23 représente un exemple de réseau de neurones convolutif avec ses différentes couches, permettant la classification des images des animaux.

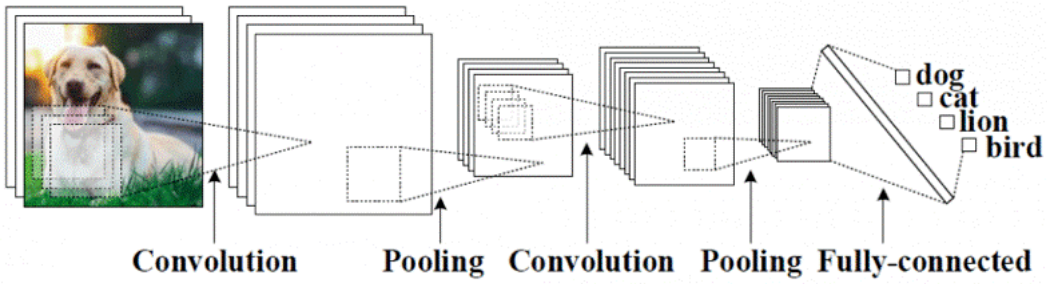


FIGURE 2.23 – Exemple d'un réseau de neurones convolutif.

Conclusion

Au cours de ce chapitre, nous avons présenté le concept de la classification des images en utilisant les réseaux de neurones artificiels. En particulier, nous avons pu voir un aperçu sur les images numériques et leurs caractéristiques, les différentes méthodes de classification des images en se basant sur la méthode des réseaux de neurones artificiels, leurs composants, leur fonctionnement et leurs différents types. Nous nous sommes basées sur les CNN du moment qu'il est le type le plus utilisé pour la classification des images.

Dans le chapitre suivant, nous allons voir la partie conception et implémentation de notre modèle de réseau de neurones pour la reconnaissance de chiffres manuscrits.

CHAPITRE

3

CONCEPTION

Introduction

La reconnaissance des caractères manuscrits apparait comme un sujet de recherche toujours vivace et suffisamment documenté pour le prendre comme une base de travail dans un système embarqué. De nombreux cas d'utilisation peuvent être cités. A titre d'exemple, certains véhicules modernes sont dotés d'écrans tactiles et les conducteurs introduisent des commandes de façon manuscrites, leurs permettant de ne pas regarder l'écran et de rester concentrés sur la route.

Dans le cadre de notre travail nous allons réaliser un système de reconnaissance des chiffres manuscrits par le biais d'un réseau de neurones convolutifs implémenté sur la carte de développement STM32F429I-DISCOVERY.

Dans ce chapitre nous nous concentrons sur la partie conception de notre modèle CNN. En commençant par définir la reconnaissance de caractères, les différentes phases d'un système de reconnaissance d'écriture, puis nous allons décrire la base de données MNIST que nous avons utilisé pour l'apprentissage de notre modèle, et nous allons voir en détail les différentes étapes de la création de ce modèle.

3.1 Définition de la reconnaissance de caractères

La reconnaissance optique de caractères sous le nom d'O.C.R (Optical Character Recognition) est une opération informatique rapide permettant de réaliser la transformation d'un texte écrit à la main en un texte sous forme d'un fichier informatique en représentation symbolique (par exemple pour les écritures latines, le codage opéré est le code ASCII (American Standard code for information interchange), tandis que pour l'arabe on utilise généralement le code ASMO (Arabic Standard Metrology Organization). [14]

La reconnaissance de caractères manuscrits est la transcription des données manuscrites au format numérique pour obtenir sa signification. C'est une tâche difficile due à la grande variabilité des styles d'écriture du moment que chaque personne possède une écriture qui lui est propre. Elle est plus difficile au niveau de la reconnaissance du texte où elle passe par plusieurs processus qui sont : la segmentation du texte en phrases, les phrases en mots et les mots en lettres pour leur reconnaissance.

Un système de reconnaissance de caractères nécessite trois étapes principales notamment : Le prétraitement dont l'objectif est l'extraction des caractères bruts sans bruit ni texture en procédant par des techniques de Binarisation, de la réduction de bruit, de la normalisation, et de la squelettisation. L'extraction de caractéristiques est une étape cruciale dont l'objectif est d'extraire les vecteurs d'attributs constituant les informations pertinentes permettant la distinction

des caractères. La classification est l'étape finale dont l'objectif est la décision. Elle se fait par des techniques de classification ou d'apprentissage.

L'intérêt remarquable de ce domaine de recherche a été démontré par plusieurs applications réelles dans plusieurs domaines telles que :

➢ **Domaine bancaire :** l'authentification des chèques par les banques, l'identification du montant sur chèques bancaires.

➢ **Assistance à l'éducation :** la reconnaissance et la traduction de textes manuscrits et l'apprentissage de l'écriture et de la lecture.

➢ **Lecture des adresses postales :** la reconnaissance automatique et traitement des adresses postales, ceci a permis d'atteindre le développement des machines de tri automatique de courrier.

➢ **La police et la sécurité :** la reconnaissance des numéros minéralogiques pour le contrôle routier.

3.2 Les phases d'un système de reconnaissance d'écriture

Un système de reconnaissance fait appel généralement aux étapes suivantes : acquisition, prétraitement, segmentation, extraction des caractéristiques et classification, suivie d'une phase de post-traitement. [15]

La figure 3.1 montre un schéma représentant les phases d'un système de reconnaissance de caractères.

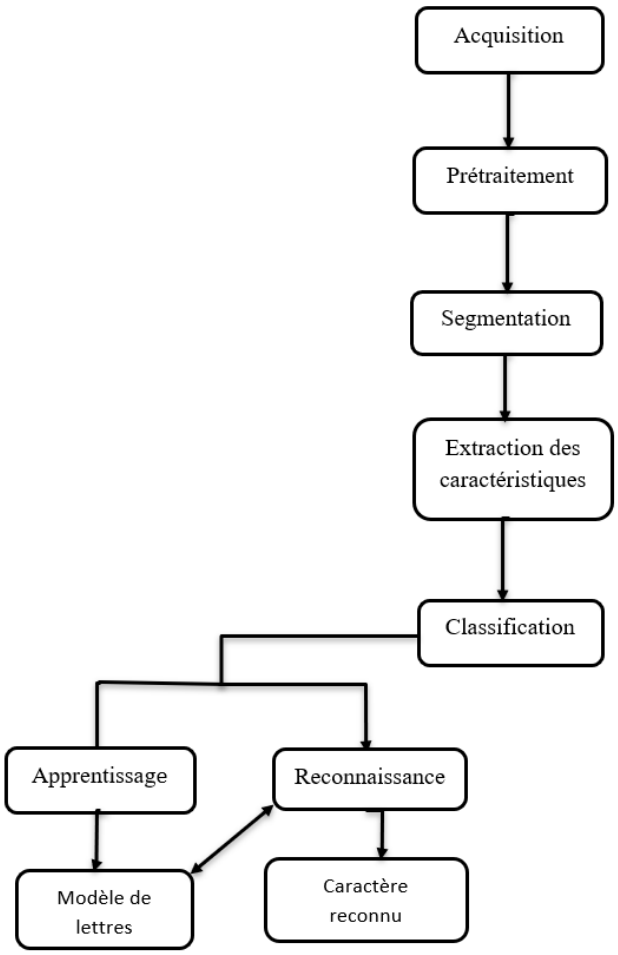


FIGURE 3.1 – Schéma général d'un système de reconnaissance de caractères.

3.2.1 Acquisition

L'acquisition permet la conversion du document papier sous la forme d'une image numérique (bitmap). Cette étape est importante car elle se préoccupe de la préparation des documents à saisir, du choix et du paramétrage du matériel de saisie (scanner), ainsi que du format de stockage des images.

L'acquisition hors ligne et l'acquisition en ligne sont deux modes différents d'un système de reconnaissance des mots. Chacune ayant ses propres outils d'acquisition et ses algorithmes de reconnaissance correspondants.

• L'acquisition hors ligne :

Il s'agit de reconnaître des textes manuscrits à partir de documents écrits au préalable. L'image du texte écrit est numérisée à l'aide d'un scanner, les informations recueillies se présentent sous la forme d'une image discrète constituée d'un ensemble de pixels.

• L'acquisition en ligne :

Il s'agit de reconnaître l'écriture au fur et à mesure de son tracé. Le texte est saisi avec un stylo sur une tablette à numériser. Les informations recueillies sont constituées par une suite ordonnée de points (définis par leurs coordonnées) échantillonnés à cadence fixe.

3.2.2 Prétraitement

Cette étape prépare l'image d'entrée pour l'étape de reconnaissance. Il s'agit essentiellement de réduire le bruit superposé aux données et essayer de ne garder que l'information significative de la forme représentée. Le bruit peut être dû aux conditions d'acquisition (éclairage, disposition incorrecte du document, ...) ou encore à la qualité du document d'origine.

Parmi les opérations de prétraitement généralement utilisées on peut citer : l'extraction des composantes connexes, le redressement de l'écriture, le lissage, la binarisation, la normalisation et la squelettisation.

3.2.3 Segmentation

Dans cette phase les différentes parties logiques d'une image sont extraites. A partir d'une image acquise, il y'a d'abord la séparation des blocs de texte et des blocs graphiques, puis à partir d'un bloc de texte on procède à l'extraction des lignes. Ensuite, à partir de ces lignes on extrait les mots puis les caractères.

3.2.4 Extraction de caractéristiques

C'est l'une des étapes les plus délicates et les plus importantes en OCR. La reconnaissance d'un caractère passe d'abord par l'analyse de sa forme et l'extraction de ses traits caractéristiques (primitives) qui seront exploités pour son identification.

3.2.5 Classification

La phase de classification consiste à interpréter les données obtenues et à décider à quelle classe appartient le caractère analysé. Cette phase regroupe les deux taches d'apprentissage et de décision.

Le résultat de l'apprentissage est soit la réorganisation ou le renforcement des modèles existants en tenant compte de l'apport de la nouvelle forme, soit la création d'un nouveau modèle de l'apprentissage. Le résultat de la décision est un avis sur l'appartenance ou non de la forme aux modèles de l'apprentissage.

3.2.6 Post-traitement

Cette étape aide à réduire considérablement des erreurs. Elle permet la validation des décisions de l'analyse sur la base de connaissances. Cependant, ce n'est pas une étape complètement séparée des autres étapes.

3.3 La reconnaissance de chiffres manuscrits

La reconnaissance automatique de chiffres manuscrits est la capacité des ordinateurs à reconnaître les chiffres manuscrits humains. Elle a pour objectif de convertir des images de chiffres qui sont compréhensibles par l'homme, en un code interprétable par un ordinateur. C'est une tâche difficile pour la machine car les chiffres manuscrits diffèrent par leur taille, leur largeur et leur orientation, et l'écriture de chaque personne comporte des variations distinctes. Différentes cultures écrivent même les chiffres différemment. Ajoutant à cela le problème de similitude entre certains chiffres par exemple 3 et 8, 0 et 8, 5 et 6, etc.

L'objectif de ce travail est d'implémenter un système de reconnaissance de chiffres manuscrits sur un système embarqué à faible coût, à l'aide de l'ensemble de données MNIST (Modified ou Mixed National Institute of Standards and Technology). Cet ensemble servira de données d'entrainement pour notre modèle de réseau de neurones convolutifs. Le CNN sera implémenté par la suite sur la carte de développement STM32F429I-DISCOVERY.

3.4 Description de la base de données MNIST

Nous avons implémenté notre système de reconnaissance de chiffres manuscrits avec le célèbre jeu de données MNIST issu d'un institut célèbre dans l'élaboration de normes et technologies (NIST). C'est un jeu de données très utilisé en apprentissage automatique. Depuis sa publication en 1999, cet ensemble de données classique d'images manuscrites a servi de base à l'analyse comparative des algorithmes de classification. À mesure que de nouvelles techniques d'apprentissage automatique émergent, MNIST reste une ressource fiable pour les chercheurs et les apprenants. [16]

La base de données MNIST contient une collection de 70 000 chiffres manuscrits divisés en un ensemble d'apprentissage de 60 000 images et un ensemble de test de 10 000 images. [16]

FIGURE 3.2 – Une partie de la base de données MNIST.

La figure 3.2 présente un échantillon de cet ensemble de données. Dans la figure, chaque chiffre a sa représentation sous forme d'une image ainsi que son étiquette correspondante. Par exemple, le premier chiffre en bas à gauche a une étiquette égale à 9. Par ailleurs, la représentation des chiffres est normalisée à travers tout le jeu de données MNIST. Ainsi, chaque chiffre est codé dans un format 28 pixels x 28 pixels, chaque pixel pouvant prendre une valeur comprise entre 0 et 255. Cette plage de valeurs représente les niveaux de gris. Autrement dit, chaque représentation d'une image est une matrice de dimension 28 x 28 pixels.

3.5 Conception du système

Dans le cadre de notre travail, nous déployons un réseau de neurones sur un système embarqué à faible consommation. La réalisation de notre système est faite en deux phases :

3.5.1 Phase de création du modèle de réseau de neurones

Cette phase consiste en la création d'un réseau neuronal convolutif permettant la reconnaissance de chiffres manuscrits à base de l'ensemble de données MNIST. Cela est fait en utilisant Tensorflow et Keras qui sont des bibliothèques implémentant des méthodes de deep learning et permettant l'interaction avec les algorithmes de réseaux de neurones profonds.

3.5.2 Phase d'implémentation sur l'µC

Cette phase consiste à implémenter le CNN pré-entrainé (modèle résultant de la phase précédente) sur la carte STM32F429I-Discovery. Cela est fait grâce à l'outil logiciel STM32CUBE.IA qui est entièrement intégré dans l'écosystème de développement logiciel STM32 en tant qu'extension de l'outil STM32CubeMX. Il permet une conversion rapide et automatique du CNN en

code optimisé (minimisant la complexité et les besoins en mémoire) pouvant s'exécuter sur le microcontrôleur de cette carte.

Nous parlerons plus en détails de ces différents outils et bibliothèques dans le chapitre suivant.

Le schéma de la figure 3.3 présente une architecture globale de notre système :

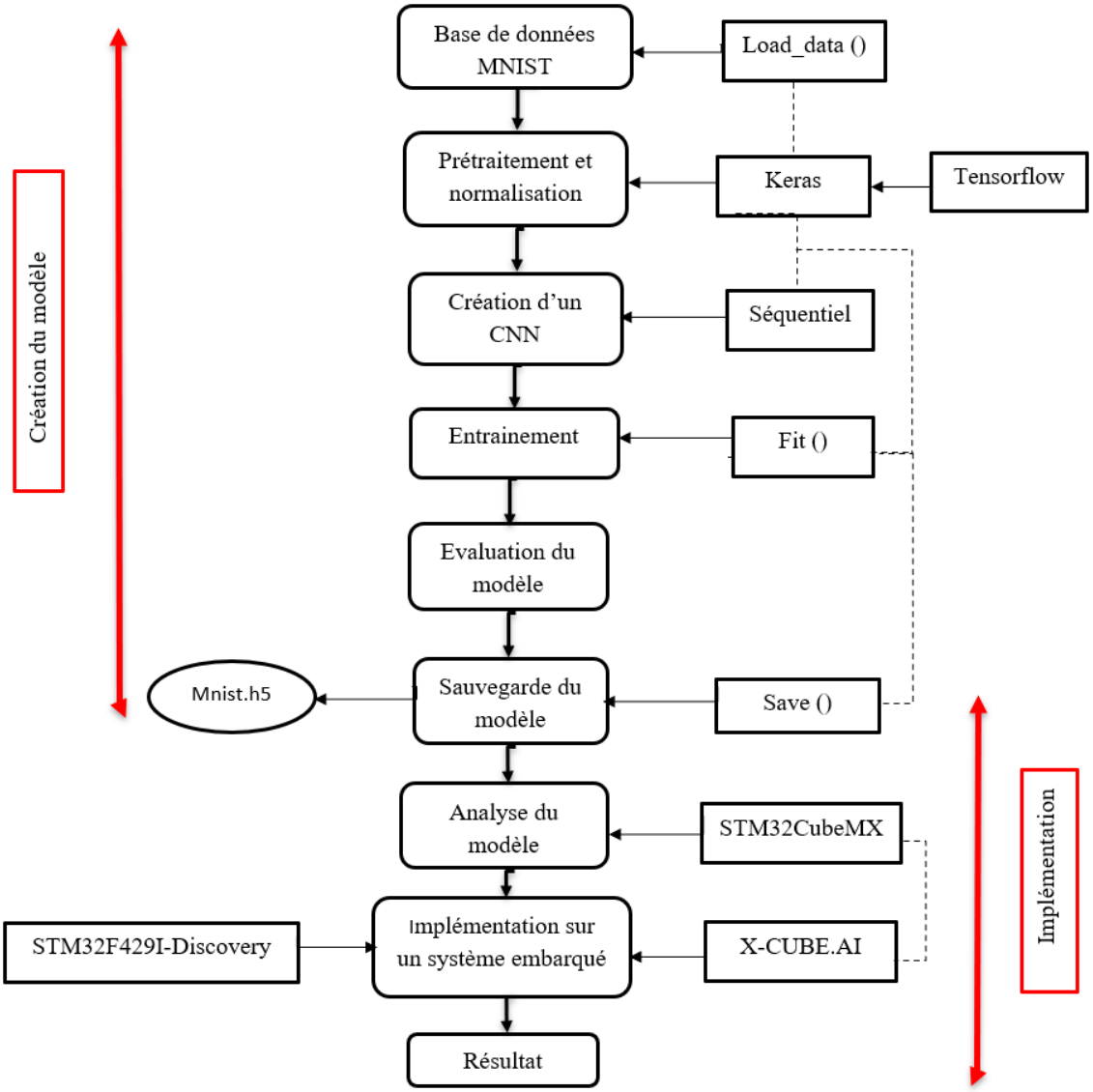


FIGURE 3.3 – Architecture globale du système.

Dans ce qui suit nous nous basons sur la phase de la création du notre modèle, nous allons voir en détail les étapes de la mise en œuvre de notre CNN partant de l'importation de la base de données MNIST jusqu'à l'étape d'évaluation du modèle.

3.6 Les étapes de la mise en œuvre de CNN

3.6.1 Importation de la base de données MNIST

La fonction *mnist.load_data*() de Keras nous renvoie les données d'apprentissage, ses étiquettes ainsi que les données de test et ses étiquettes de la base de données MNIST. Les images de cet ensemble de données sont enregistrées dans les fichiers de données csv (*mnist_train.csv* et *mnist_test.csv*). Chaque ligne de ces fichiers est constituée d'une image, soit 785 nombres compris entre 0 et 255. Le premier chiffre de chaque ligne est l'étiquette, c'est-à-dire le chiffre qui est représenté dans l'image. Les 784 nombres suivants sont les pixels de l'image 28 x 28.

La figure 3.4 montre une partie de la base de données MNIST sous forme de fichier csv (nous représentons 10 lignes) :

	7	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	...	0.658	0.659	0.660	0.661	0.662	0.663	0.664	0.665	0.666	0.667
0	2	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
1	1	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
3	4	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
4	1	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
5	4	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
6	9	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
7	5	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
8	9	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0

10 rows × 785 columns

FIGURE 3.4 – Représentation de la base de données MNIST sous format csv.

Nous présentons dans la figure 3.5 14 images du dataset prises de manière aléatoire, dont chaque image est étiquetée par le chiffre qu'elle représente.

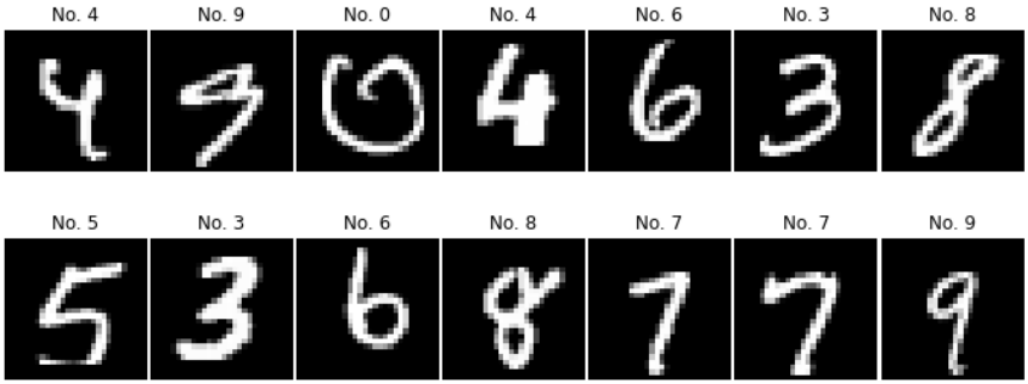


FIGURE 3.5 – Représentation de la base de données MNIST.

3.6.2 Pré-traitement et normalisation de données :

Nous rappelons que toutes les images de notre dataset sont représentées sous la forme d'une matrice 28 × 28 contenant des valeurs de pixels en niveaux de gris (0-255), et qu'il est divisé en 60 000 images d'entrainement (*train_data*) et 10 000 images de test (*test_data*). Par conséquent, la dimension des données de l'ensemble *train_data* est (60 000x28x28) et celle du *test_data* est (10 000x28x28). Ainsi, ces données ne peuvent pas être introduites directement dans notre modèle qui nécessitera une dimension de plus (256 niveaux de gris). Nous devons donc transformer données sous forme (60 000,28,28,1) pour l'ensemble *train_data* et (10 000,28,28,1) pour l'ensemble *test_data* comme indiqué dans la figure 3.6

```
train_data.shape
```

```
(60000, 28, 28, 1)
```

```
test_data.shape
```

```
(10000, 28, 28, 1)
```

FIGURE 3.6 – Représentation de la dimension de l'ensemble d'apprentissage et de test.

La normalisation est une technique souvent appliquée dans le cadre de la préparation des données pour le Machine Learning. Son objectif est de modifier les valeurs des colonnes numériques du jeu de données pour utiliser une échelle commune, sans que les différences de plages de valeurs ne soient faussées et sans perte d'informations.

Dans notre cas, les images de l'ensemble de données MNIST sont en niveaux de gris et les pixels sont compris entre 0 et 255 y compris les deux valeurs limites. Nous allons mapper ces valeurs dans un intervalle de [0, 1] en divisant chaque pixel par 255.

Comme notre dataset contient un ensemble de chiffres, on aura donc 10 étiquettes, ce qui est équivalant à 10 classes dont la dimension est de (60 000x10) pour l'ensemble d'apprentissage et (10 000x10) pour l'ensemble de test comme indiqué dans la figure 3.7.

```
train_labels_cat.shape
(60000, 10)

test_labels_cat.shape
(10000, 10)
```

FIGURE 3.7 – Représentation de la dimension des catégories de l'ensemble d'apprentissage et de test.

Avant de passer à la création de notre modèle nous avons partitionné les 60 0000 images de l'ensemble d'apprentissage en deux ensembles. Un ensemble servant à l'apprentissage proprement dit, formé de 90% d'images et un ensemble de test de validation formé de de 10%. On obtient ainsi les sous-ensembles indiqués dans la figure 3.8.

```
train_data2.shape
(54000, 28, 28, 1)

train_labels_cat2.shape
(54000, 10)
```

```
val_data.shape
(6000, 28, 28, 1)

val_labels_cat.shape
(6000, 10)
```

FIGURE 3.8 – Représentation de la dimension de l'ensemble d'entrainement et de validation.

La figure 3.9 représente un schéma global de la structure de données MNIST.

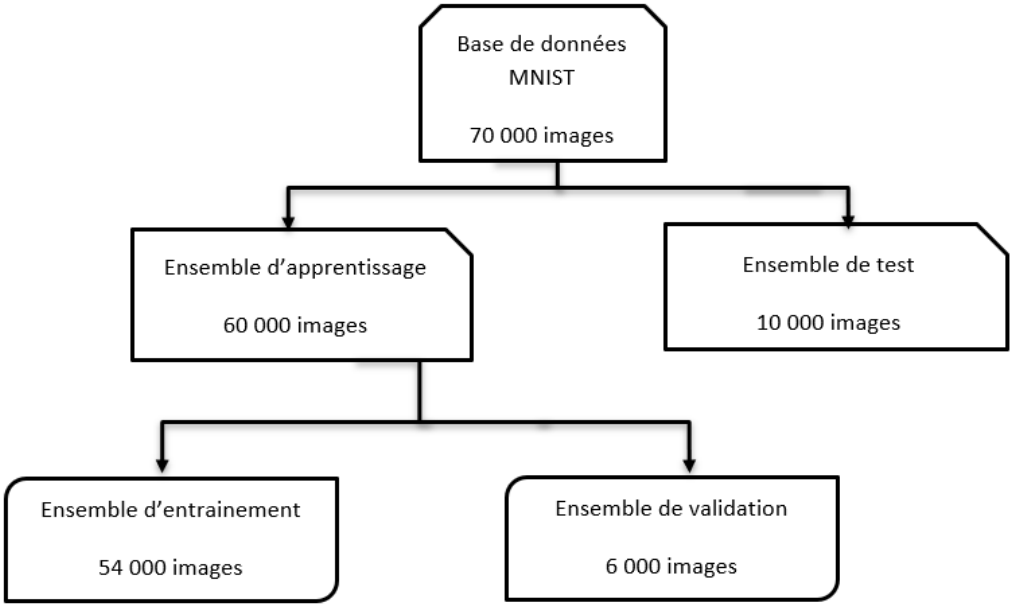


FIGURE 3.9 – Représentation globale de la structure de données MNIST.

À partir de lâ, nous sommes prêts à construire notre modèle.

3.6.3 Création du modèle

Nous allons maintenant créer notre modèle CNN. Ce dernier est composé de couches convolutives. Cela fonctionne mieux pour les données qui sont représentées sous forme de structures de grille. C'est la raison pour laquelle le CNN fonctionne bien pour les problèmes de classification d'images. Le type de modèle que nous avons utilisé est Séquentiel. Il est le moyen le plus simple de créer un modèle dans Keras. Il nous permet de construire un modèle couche par couche.

Notre CNN est composé au total de 9 couches à savoir :(voir la figure 3.10)

✓ **Une couche d'entrée :**

C'est d'une couche de convolution (Conv2D) qui traite nos images d'entrée. Elle contient 32 nœuds. La taille du noyau est une matrice de filtre 3x3, et elle est suivie d'une couche MaxPooling2D pour réduire la taille des données.

✓ **Deux couches de convolution :**

Chacune d'elles est composée de 64 nœuds, et suivie d'une couche MaxPooling2D de 64 nœuds dont le rôle est de Sous-échantillonner l'entrée le long de ses dimensions spatiales en prenant la valeur maximale sur une fenêtre d'entrée ($pool\_size$) pour chaque canal d'entrée.

✓ **Une couche Flatten :**
Qui sert de connexion entre la convolution et les couches denses, elle permet de rendre toutes les données à une dimension.

✓ **Une couche Dense (entièrement connectée) :**
Comportant 128 nœuds qui se connectent à tous les nœuds de la couche suivante qui s'agit de la couche de sortie.

✓ **Une couche de sortie :**
Composée de 10 nœuds, un pour chaque résultat possible (0-9).

Les fonctions d'activation que nous avons utilisé dans notre CNN est la fonction Relu pour la couche d'entrée et les couches cachées. Tandis que pour la couche de sortie nous avons utilisé la fonction Softmax qui fait la somme de sortie jusqu'à 1, afin que la sortie puisse être interprétée comme des probabilités. Le modèle fera ensuite sa prédiction en fonction de l'option ayant la probabilité la plus élevée.

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 28, 28, 32)        320
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 14, 14, 32)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 14, 14, 64)        18496
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 7, 7, 64)          0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 7, 7, 64)          36928
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 3, 3, 64)          0
_________________________________________________________________
flatten (Flatten)            (None, 576)               0
_________________________________________________________________
dense (Dense)                (None, 128)               73856
_________________________________________________________________
dense_1 (Dense)              (None, 10)                1290
=================================================================
Total params: 130,890
Trainable params: 130,890
Non-trainable params: 0
_________________________________________________________________
None
```

FIGURE 3.10 – Architecture du modèle séquentiel.

Maintenant notre modèle est prêt à être entrainé.

3.6.4 Entrainement du modèle

Pour l'entrainement de notre modèle nous avons utilisé la fonction fit() de Keras avec les paramètres suivants : données d'entraînement(Training data) données de validation (Validation data), la taille du lot (batch size) et le nombre d'epochs, qui est le nombre de fois que le modèle parcourra les données .Plus nous exécutons d'épochs plus le modèle s'améliorera jusqu'à un certain point. Après ce point, le modèle cessera de s'améliorer. Pour notre modèle, nous avons défini le nombre d'épochs à 15.

```
Epoch 8/15
54000/54000 [==============================] - 69s 1ms/sample - loss: 0.0117 - accuracy: 0.9962 - val_loss: 0.0297 - val_accura
cy: 0.9922
Epoch 9/15
54000/54000 [==============================] - 73s 1ms/sample - loss: 0.0120 - accuracy: 0.9960 - val_loss: 0.0318 - val_accura
cy: 0.9922
Epoch 10/15
54000/54000 [==============================] - 75s 1ms/sample - loss: 0.0102 - accuracy: 0.9966 - val_loss: 0.0355 - val_accura
cy: 0.9922
Epoch 11/15
54000/54000 [==============================] - 71s 1ms/sample - loss: 0.0102 - accuracy: 0.9966 - val_loss: 0.0421 - val_accura
cy: 0.9905
Epoch 12/15
54000/54000 [==============================] - 69s 1ms/sample - loss: 0.0074 - accuracy: 0.9976 - val_loss: 0.0293 - val_accura
cy: 0.9920
Epoch 13/15
54000/54000 [==============================] - 81s 1ms/sample - loss: 0.0074 - accuracy: 0.9975 - val_loss: 0.0388 - val_accura
cy: 0.9908
Epoch 14/15
54000/54000 [==============================] - 72s 1ms/sample - loss: 0.0066 - accuracy: 0.9980 - val_loss: 0.0382 - val_accura
cy: 0.9927
Epoch 15/15
54000/54000 [==============================] - 77s 1ms/sample - loss: 0.0056 - accuracy: 0.9979 - val_loss: 0.0485 - val_accura
cy: 0.9905
```

FIGURE 3.11 – Entrainement du modèle.

Comme le montre la figure 3.11, après l'entrainement de notre modèle à 15 epochs, nous avons obtenu une précision d'entrainement de 99,79% et une précision de validation de 99,05%. La perte de formation et de validation est de 0,56% et 4.85% respectivement.

Grâce à l'utilisation de la bibliothèque matplotlib, il nous a été possible de suivre l'évolution de la précision et de la perte de notre modèle tout au long de son entraînement comme le montrent les deux figures 3.12 et 3.13 suivantes :

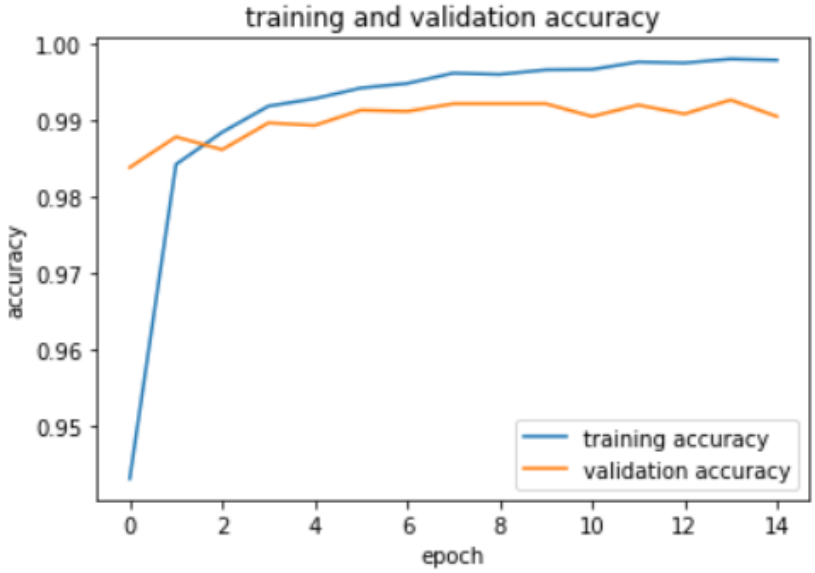


FIGURE 3.12 – La courbe d'évolution de la précision du CNN.

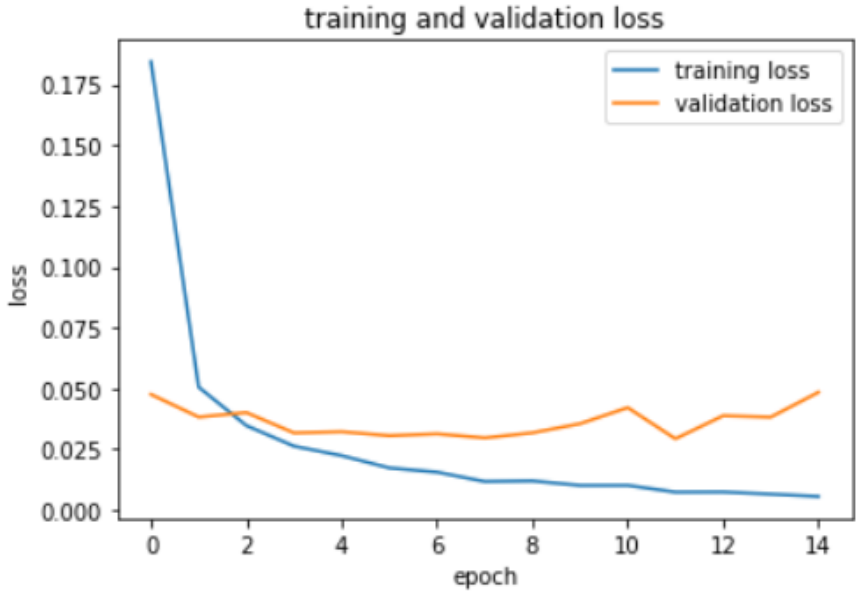


FIGURE 3.13 – La courbe d'évolution de l'erreur du CNN.

En se focalisant sur les deux courbes précédentes, nous pouvons observer l'évolution des deux mesures (la précision et l'erreur). En effet, la précision mesure la faculté du modèle à réaliser de bonnes prédictions, la fonction de coûts (erreur) permet d'évaluer l'écart entre les prédictions attendues et celles obtenues. Au fur et à mesure que notre modèle s'entraine en fonction de nombre d'époch déterminé, le nombre d'erreurs qu'il commet diminue et la précision augmente. Ainsi, on peut dire que notre modèle est opérationnel nous permettra de réaliser de bonnes prédictions.

Cependant, en faisant une évaluation sur l'ensemble de test, ces deux mesures (précision et erreur) prennent respectivement les valeurs 99.12% et 4%

```
10000/10000 [==============================] - 3s 313us/sample - loss: 0.0400 - accuracy: 0.9912
Test loss: 0.0400 and test accuracy: 0.9912
```

FIGURE 3.14 – La valeur de précision et de perte de l'ensemble du test.

À partir de là, on peut dire que l'ensemble de données MNIST est bien équilibré, ce qui nous a permet d'obtenir un modèle avec une précision d'environ 99%.

3.6.5 Evaluation du modèle

Maintenant, nous allons utiliser notre modèle pour faire des prédictions. Les données de notre ensemble test n'ont pas été impliquées dans l'apprentissage du modèle, il s'agit donc de nouvelles données pour notre modèle. Du coup, nous avons 10 000 images qui seront utilisées pour évaluer le bon fonctionnement de notre modèle.

Premièrement nous allons voir les prédictions réelles que notre modèle a fait pour les données de l'ensemble de test. Pour cela nous allons voir la prédiction des premiers 25 chiffres de cet ensemble, comme le montre la figure 3.15 :

```
predected values :
[7 2 1 0 4 1 4 9 5 9 0 6 9 0 1 5 9 7 3 4 9 6 6 5 4]
true values :
[7 2 1 0 4 1 4 9 5 9 0 6 9 0 1 5 9 7 3 4 9 6 6 5 4]
```

FIGURE 3.15 – la valeur de précision et de perte de l'ensemble test.

Pour être plus précis, nous allons montrer les prédictions pour quelques images de l'ensemble de test.

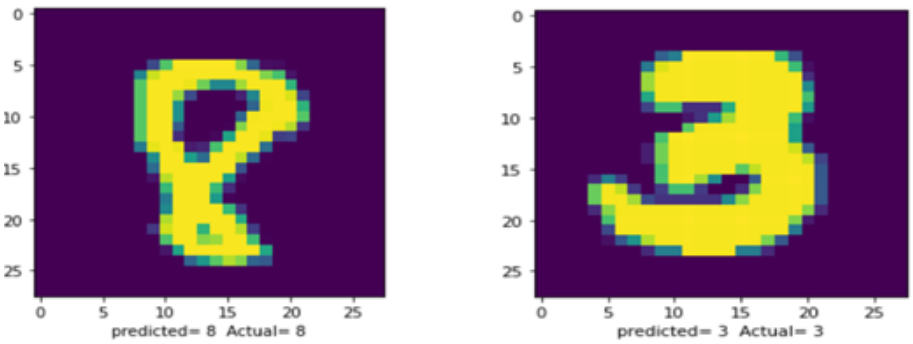


FIGURE 3.16 – Résultats de prédiction des images de l'ensemble test.

Voyons maintenat les prédictions que notre modèle fait pour des images des chiffres manuscrits que nous avons téléchargé du Google (Figure 3.17) :

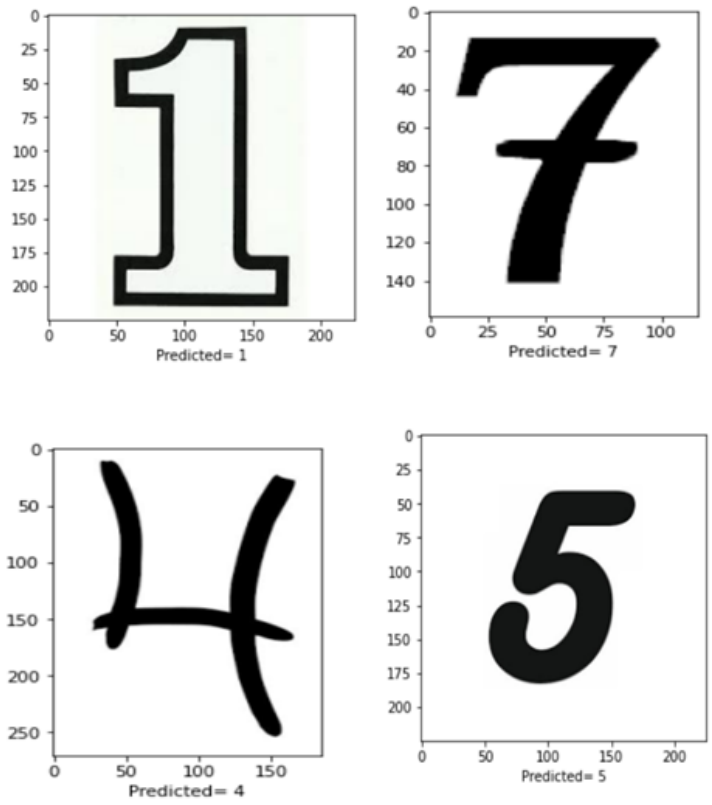


FIGURE 3.17 – Résultats de prédiction.

Apres avoir évalué notre modèle, nous avons constaté qu'il fait de bonnes prédictions. Maintenant nous le sauvegardons sous forme de fichier 'mnist.h5' pour pouvoir l'implémenter sur la carte de développement STM32F429I-Discovery.

Conclusion

Jusqu'à ici nous avons pu créer notre propre modèle de reconnaissance de chiffres manuscrits en utilisant le célèbre jeu de donnés MNIST avec lequel nous avons eu de bons résultats de prédiction. Maintenant nous pouvons passer à la deuxième phase de notre système.

Dans le prochain chapitre, nous présenterons la partie réalisation de notre travail qui consiste en l'implémentation de notre modèle CNN sur la carte STM32F429I-Discovery.Ainsi, nous allons présenter les différents outils matériels et logiciels que nous avons utilisé au cours de cette réalisation.

CHAPITRE

4

IMPLÉMENTATION ET RÉALISATION

Introduction

Dans ce chapitre nous présenterons l'environnement matériel et logiciel de notre travail que nous avons utilisé dans la phase d'implémentation. Ensuite, nous présenterons les différentes configurations nécessaires pour la carte STM32F429I-Discovery sur laquelle nous avons implémenté notre modèle de reconnaissance de chiffres manuscrits en utilisant l'environnement de développement STM32CubeMX et son package STM32CUBE.AI.

Les tests du système seront effectués directement sur la carte en utilisant l'écran tactile intégré á cette dernière.

4.1 Environnement de développement matériel

4.1.1 La carte STm32f429I-Discovery

STMicroelectronics (souvent appelée simplement ST) est un fabricant multinational franco-italien d'électronique et de semi-conducteurs. Il figure parmi les leaders mondiaux de la conception, de la production et de la commercialisation de semi-conducteurs. C'est une société de développement et de réalisation de solutions destinées à un grand nombre d'applications microélectroniques. Elle fournit une vaste gamme de cartes électroniques dont la famille microcontrôleur STM32FXXXX.

Dans le cadre de notre travail nous avons utilisé la carte STM32F429I-Discovery disponible dans notre département (Figure 4.1). Elle est développée autour du microprocesseur ARM Cortex-M4 étudié dans le cadre du module programmation d'interfaces. Elle exploite les capacités et les hautes performances des microcontrôleurs STM32F429 pour permettre aux utilisateurs de développer facilement des applications riches avec des interfaces utilisateur graphiques avancées.

Caractéristiques : [17]

La carte STM32F429I-DISCOVERY offre les caractéristiques suivantes :

• Microcontrôleur STM32F429ZIT6 doté de 2 Mo de mémoire Flash, 256 Ko de RAM dans un boîtier LQFP144.
• ST-LINK/V2 embarqué sur STM32F429I- DISC1.
• Fonctions USB :
  ✧ Port de débogage.
  ✧ Port COM virtuel.
  ✧ Stockage de masse.

• Alimentation de la carte : via le bus USB ou à partir d'une tension d'alimentation externe 3V ou 5V.

• LCD TFT QVGA 2.4 (240 x 320), 262K couleurs RGB.

• SDRAM 64 Mbit.

• L3GD20, capteur de mouvement ST-MEMS Gyroscope à sortie numérique 3 axes.

• Six LEDs :

✧ LD1 (rouge/vert) pour communication USB.

✧ LD2 (rouge) pour mise sous tension 3,3 V.

✧ Deux LEDs utilisateur : LD3 (vert), LD4 (rouge).

✧ Deux LEDs USB OTG : LD5 (vert) VBUS et LD6 (rouge) OC (surintensité).

✧ Deux boutons-poussoirs (utilisateur et reset).

• USB OTG avec connecteur micro-AB.

• En-têtes d'extension pour E/S LQFP144 pour une connexion rapide à la carte de prototypage.

• Logiciel comprenant une variété d'exemples, faisant partie du package STM32CubeF4 ou STSW-STM32138, pour l'utilisation des bibliothèques standard héritées.

FIGURE 4.1 – la carte STM32F429I-DISC1.

Comme notre modèle CNN vu dans le chapitre précèdent réalise la reconnaissance de chiffres manuscrits, nous avons donc utilisé l'afficheur tactile de la carte STM32F429I-DISC1 pour pouvoir écrire un chiffre à la main et le prédire.

La carte STM32F429I-DISC1 présente plusieurs fonctions d'acquisitions, de traitement et de communication. Ici, nous présenterons seulement les fonctionnalités en relation avec notre projet, qui sont :

4.1.1.1 Entrées Sorties à usage Général (GPIOs) :

Un microcontrôleur interagit avec les périphériques externes par le biais de plusieurs broches, ou pins en anglais. Ces broches sont regroupées généralement par paquet pour former ce que l'on appelle ports d'entrée/sortie (I/O ports).

Du point de vue logiciel, un port GPIO est un ensemble de registres dont :

✓ Quatre registres de configuration ($GPIOx\_MODER, GPIOx\_OTYPER, GPIOx\_OSPEEDR$ et $GPIOx\_PUPDR$).
✓ Deux registres de données d'entrée/sortie ou périphérique ($GPIOx\_IDR$ et $GPIOx\_ODR$).
✓ Un registre set/reset ($GPIOx\_BSRR$).
✓ Un registre de verrouillage ($GPIOx\_LCKR$).
✓ Deux registres de sélection de fonction alternative ($GPIOx\_AFRHetGPIOx\_AFRL$).

Pour configurer un périphérique relié à une broche il suffit de configurer les GPIOs Correspondants en utilisant le module HAL_GPIO.

Caractéristiques principales du GPIO :

• Jusqu'à 16 E/S sous contrôle.
• États de sortie : push-pull, open-drain, pull-up/down.
• États d'entrée : flottant, pull-up/down, analogique.
• Registres de sélection d'entrée/sortie de fonction alternative (au plus 16 fonctions par E/S).
• Basculement rapide capable de changer tous les deux cycles d'horloge.
• Attribution séparée des vitesses pour les E/S.
• Un multiplexage très flexible des broches permettant l'utilisation des broches d'E/S comme GPIO avec une simple configuration sans connaitre les registres, leurs mappages en mémoires ou la manière de configuration des périphériques.

4.1.1.2 Les UARTs :

L'UART pour Universal Asynchronous Receiver Transmitter est une interface classique de communication série en mode asynchrone. D'un côté, elle reçoit du processeur une donnée parallèle sur 8 bits, la sérialise et l'envoie bit par bit sur la ligne TxD. De l'autre côté Elle reçoit sur la ligne

RxD un ensemble de bits en série, reconstitue la donnée sur 8 bits et la fournit au processeur via le bus comme le montre le schéma de la figure 4.2 :

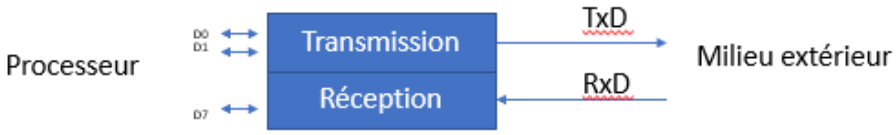


FIGURE 4.2 – la liaison UART.

La transmission est asynchrone, les horloges internes de l'émetteur et du récepteur doivent être préalablement réglées à la même cadence. On appel ce paramètre le Baud Rate. Les valeurs de BR les plus utilisées sont : 300 bps, 1 200 bps, 2 400 bps, 9 600 bps, 19 200 bps, 38 400 bps, 57 600 bps et 115 200 bps. [18]

La STM32F4 dispose de 6 interfaces USART, supportant la communication synchrone ou asynchrone. Une interface USART est connectée en externe à un autre appareil par trois broches. Toute communication bidirectionnelle USART nécessite un minimum de deux broches : Receive Data In (RX) et Transmit Data Out (TX). Grâce à ces broches, les données série sont transmises et reçues en mode USART comme des frames comprenant :

✓ Une ligne libre avant la transmission ou la réception (signal haut).
✓ Un bit start (signal bas).
✓ Un mot de données (8 ou 9 bits) le bit le moins significatif en premier.
✓ Un bit stop (0.5, 1, 1.5, 2) indiquant que la trame est terminée (signal bas).

Cette interface utilise :

✓ Un registre de débit en baud rate (registre $USART\_BRR$), avec une mantisse de 12 bits et 4 bits pour la partie fractionnaire.
✓ Un registre d'état ($USART\_SR$).
✓ Un registre de données ($USART\_DR$).
✓ Un registre Guardtime ($USART\_GTPR$) en cas de mode Smartcard.

Chaque interface USART offre plusieurs fonctions, à savoir :

✓ Fonctionnement en Full duplex.
✓ Génération de Baud rate avec grande précision.
✓ Contrôle de flux matériel (RTS/CTS).
✓ Taille de caractère et bit stop programmables.
✓ Différents indicateurs de transfert (Transmetteur vide, receveur plein, fin de transmission,etc).
✓ Différents indicateurs d'erreurs (parité, écrasement, détection d'interférences, etc).

✓ Support pour des protocoles de communication (IrDA « association de données infrarouges », SmartCard, etc).

✓ Fonctionnement en mode interruption.

4.1.1.3 Les horloges :

La carte STM32F4 dispose trois sources d'horloges pouvant alimenter l'horloge système du processeur dite SYSCLK :

➢ L'oscillateur interne HSI (High Speed Internal) basé sur un circuit RC.
➢ L'oscillateur externe HSE (High Speed External) basé sur un quartz externe.
➢ Le circuit overlock utilisant la technologie PLL (Phase Locked Loup).

Par défaut (après un reset), l'horloge du microcontrôleur SYSCLK est alimentée directement par l'oscillateur interne HSI, ensuite elle est divisée ou multipliée pour générer les différentes horloges alimentant la mémoire, les bus et les périphériques. L'utilisateur peut changer la source d'horloge et définir les diviseurs et multiplicateurs par programme. Ceci est assuré dans la STM32F4 par un périphérique spécial dit RCC (Reset and clock configuration).

Pour supporter des applications robustes, comme les applications temps réel, la carte peut utiliser deux autres sources d'horloge secondaires pour alimenter les périphériques les plus importants.

4.1.1.4 Les Timers :

Un timer est un compteur électronique avec une fréquence de comptage qui est une fraction de son horloge source. La vitesse de comptage peut être réduite à l'aide d'un préscaler(un diviseur) spécial pour chaque timer. Les timers de la STM32F4 sont regrouper en trois principales catégories [19]. Donnons un bref aperçu de chacune d'elles :

➢ Les timers de base :

TIM6 et TIM7 Ce sont des timers à 16 bits, Ils peuvent être utilisés comme générateurs de base de temps mais ils sont aussi spécifiquement utilisés pour piloter le convertisseur numérique-analogique (DAC). Ils représentent la forme la plus simple des timers dans les µCs STM32.

➢ Les timers à usage général :

Ce sont des timers à 16 ou 32 bits, Ils peuvent être utilisés à diverses fins, y compris la mesure des longueurs d'impulsion des signaux (capture d'entrée pour mesurer la fréquence d'un signal

externe) ou générer des formes d'onde de sortie (comparaison de sortie) par exemple pour la synchronisation, la génération de délai et la génération d'un signal PWM.

➢ **Les timers à contrôle avancé :**

TIM1 et TIM8 Ce sont des timers à 16 bits. En plus des fonctionnalités d'un timer à usage général, ils incluent plusieurs fonctionnalités liées à la commande de moteurs et aux applications de conversion de puissance numérique.

Les timers à contrôle avancé (TIM1 et TIM8) et à usage général (TIMx) sont complètement indépendants et ne partagent aucune ressource et ils peuvent être synchronisés ensemble.

4.1.1.5 Interface périphérique série SPI :

L'interface SPI fournit deux fonctions principales, prenant en charge soit le protocole SPI, soit le protocole audio I2S. Par défaut, c'est la fonction SPI qui est sélectionnée et il est possible de basculer vers la fonction I2S par logiciel. L'interface SPI permet une communication série, half / full-duplex, et synchrone avec des appareils externes.
Elle peut être configurée en tant que maître, dans ce cas elle fournit l'horloge de communication SCK au dispositif esclave externe.
L'interface SPI est connectée à des périphériques externes via quatre broches :

• **MISO (Master In / Slave Out) :** Cette broche peut être utilisée pour transmettre des données en mode esclave et recevoir des données en mode maître.
• **MOSI (Master Out / Slave In) :** Cette broche peut être utilisée pour transmettre des données dans le mode maître et recevoir des données en mode esclave.
• **SCK (Serial Clock) :** Sortie d'horloge série pour le maître SPI et entrée pour l'esclave SPI.
• **NSS (Slave Select) :** Il s'agit d'une broche optionnelle pour sélectionner un périphérique esclave. Cette broche agit comme un « chip select » pour laisser le maître SPI communiquer avec les esclaves individuellement et éviter les conflits sur les lignes de données.

Dans le cadre de notre travail, nous avons utilisé le protocole SPI (Serial Peripheral Interface) pour l'interaction avec l'afficheur tactile de la carte STM32F429I-Discovery. Ainsi, l'utilisateur pourra écrire un chiffre à la main sur cet afficheur.

4.1.1.6 Interface I2C :

L'interface I2C sert d'interfacer entre le microcontrôleur et le bus série I2C. Elle offre une capacité multi-maîtres et contrôle tous les bus I2C spécifiques du séquençage, arbitrage et timing.

Il prend en charge le mode standard SM (jusqu'à 100 kHz) et le mode rapide FM (jusqu'à 400 kHz). Il peut être utilisé à diverses fins, y compris la génération et la vérification de CRC, SMBus (bus de gestion du système) et PMBus (bus de gestion de l'alimentation).

En plus de recevoir et de transmettre des données, cette interface les convertit du format série en parallèle et vice versa. Les interruptions sont activées ou désactivées par le logiciel. Cette interface est connectée au bus I2C par une broche de données (SDA) et une broche d'horloge (SCL).

4.1.1.7 Périphérique FMC :

La STM32F4 utilise des périphériques FMC pour gérer la mémoire étendue. FMC est l'acronyme de Flexible Memory Controller, qui se traduit par Contrôleur de Mémoire Variable. Il peut être utilisé pour piloter des mémoires telles que SRAM, SDRAM, NOR FLASH et NAND FLSAH. Parmi les autres séries de contrôleurs STM32, le FSMC (Flexible Static Memory Controller) est traduit en contrôleur de mémoire variable statique, il ne peut pas donc interfacer des mémoires dynamiques comme la SDRAM.

Dans le cadre de notre travail, nous avons utilisé le périphérique FMC pour la gestion de la mémoire dynamique SDRAM. Ainsi que pour faire fonctionner l'afficheur graphique de la carte STM32F429I-Discovery, sur lequel nous allons afficher le chiffre manuscrit et le résultat de la prédiction.

4.1.2 Tranceiver USB to UART

Le convertisseur USB vers UART (Figure 4.3) est un circuit intégré qui fournit une connectivité USB aux appareils dotés d'une interface UART y compris la carte STM32F4. Il est utilisé pour envoyer ou recevoir des données série à partir d'un port USB en données série pouvant être reçues ou envoyées par une interface UART.

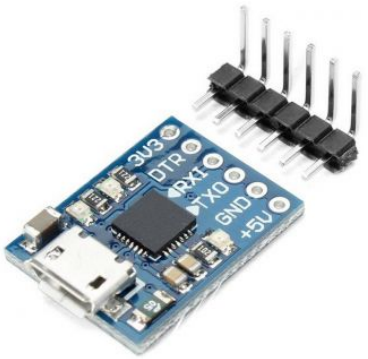


FIGURE 4.3 – Tranceiver USB to UART.

4.2 Environnement de développement logiciel

Le développement d'applications basé sur la carte STM32 ne peut se faire sans passer par les étapes de configuration et de programmation. Pour y parvenir, on utilise les deux outils STM32CubeMX et son IDE STM32CubeIDE, qui font partie de l'écosystème logiciel STM32Cube.

4.2.1 STM32CubeMX

C'est un outil graphique qui permet une configuration très simple des microcontrôleurs STM32 ainsi que la génération du code C d'initialisation correspondant pour le noyau ARM Cortex-M, à travers un processus pas à pas.

La première étape consiste à sélectionner soit un microcontrôleur STMicrolectronics STM32, soit un microprocesseur ou une plate-forme de développement qui correspond à l'ensemble des périphériques requis. Pour les microcontrôleurs, la deuxième étape consiste à configurer les GPIOs et l'horloge pour l'ensemble du système, et d'affecter de manière interactive les périphériques soit à l'ARM Cortex-M. Finalement, l'utilisateur lance la génération qui correspond aux choix de la configuration sélectionnée. Cette étape fournit le code C d'initialisation pour l'Arm Cortex-M, prêt à être utilisé dans plusieurs environnements de développement. [20]

Les piles de logiciels et de middlewares peuvent être étendues grâce aux packages d'extension STM32Cube améliorés. De plus, un utilitaire unique dans la livraison STM32CubeMX, STM32PackCreator, aide les développeurs à créer leurs propres packages d'extension STM32Cube améliorés.

Caractéristiques :

• Sélection intuitive du microcontrôleur STM32.
• Interface utilisateur graphique riche et facile à utiliser.
• Périphériques et modes fonctionnels middleware avec validation dynamique des contraintes de paramètres pour le cœur Arm Cortex-M.
• Arbre d'horloge avec validation dynamique de la configuration.
• Séquence d'alimentation avec résultats de consommation estimés.
• Génération de projet de code C d'initialisation, compatible avec MDK-ARM et STM32CubeIDE pour le noyau Arm Cortex-M.
• Intégration des packages d'extension STM32Cube dans le projet, et qui peuvent être améliorés grâce à STM32PackCreator.
• Disponibilité en tant que logiciel autonome fonctionnant sur les systèmes d'exploitation Windows, Linux et macOS.

4.2.2 STM32Cube IDE

STM32CubeIDE est une plate-forme de développement C/C++ avancée, offrant un ensemble de fonctionnalités permettant la configuration des périphériques, la génération, la compilation de code et le débogage pour les microcontrôleurs STM32.

Caractéristiques :

• Il est Basé sur le Framwork Eclipse avec prise en charge des modules complémentaires Eclipse, de la chaîne d'outils GNU C / C ++ pour Arm et du débogueur GDB.
• Intégration des services de STM32CubeMX.
• Fonctionnalités de débogage supplémentaires, notamment :
  ✧ Vues du cœur du processeur, du registre périphérique et de la mémoire.
  ✧ Analyse du système et traçage en temps réel (SWV).
  ✧ Outil d'analyse des défauts du processeur.

4.2.3 X-CUBE.AI

C'est un package d'extension d'intelligence artificielle pour STM32Cube qui fait partie de l'écosystème STM32Cube.AI. Il étend les capacités de STM32CubeMX avec la conversion automatique d'algorithmes d'intelligence artificielle pré-entraînés, y compris les modèles de réseau neuronal et d'apprentissage automatique classiques. Ce package offre également plusieurs moyens de valider les algorithmes d'IA et les modèles de réseau neuronal à la fois sur PC et STM32, ainsi que de mesurer les performances sur la carte STM32 sans code C écrit par l'utilisateur.

Caractéristiques :

• Génération d'une bibliothèque optimisée pour STM32 à partir de réseaux de neurones pré-entraînés et de modèles d'apprentissage automatique classiques.
• Prise en charge native de divers frameworks de Deep Learning tels que Keras et TensorFlow Lite, et prise en charge de tous les frameworks pouvant exporter au format standard ONNX tels que PyTorch, Microsoft Cognitive Toolkit, MATLAB, etc.
• prise en charge de la quantification 8 bits des réseaux Keras et des réseaux quantifiés TensorFlow Lite.
• Supporte l'utilisation de réseaux plus importants en stockant les poids dans la mémoire Flash externe et les tampons d'activation dans la RAM externe.
• Portabilité aisée sur différentes séries de microcontrôleurs STM32 grâce à l'intégration STM32Cube.
• Conditions de licence gratuites et conviviales.

4.2.4 Jupyter Notebook

Le bloc-notes Jupyter est un outil Web interactif (serveur-client) gratuit et open source (Figure 4.4). Il permet de modifier et d'exécuter des documents de notebook via un navigateur Web. Il est adapté à la capture de l'ensemble du processus de calcul : développement, documentation et exécution du code, ainsi que la communication des résultats. Il peut être exécuté sur un bureau local ne nécessitant aucun accès Internet ou peut être installé sur un serveur distant et accessible via Internet.

Le notebook Jupyter combine deux composants : Une application web permettant la création interactive de documents qui combinent du texte explicatif, des images et des formules mathématiques ; et des Documents Notebook représentants tout le contenu visible dans l'application Web. Dans le cadre de notre travail, nous avons utilisé ce Notebook au cours de la phase de la mise en œuvre de notre modèle CNN. Citant à titre d'exemples : l'affichage de la base de données MNIST, la création du CNN, ainsi que son entrainement et évaluation.

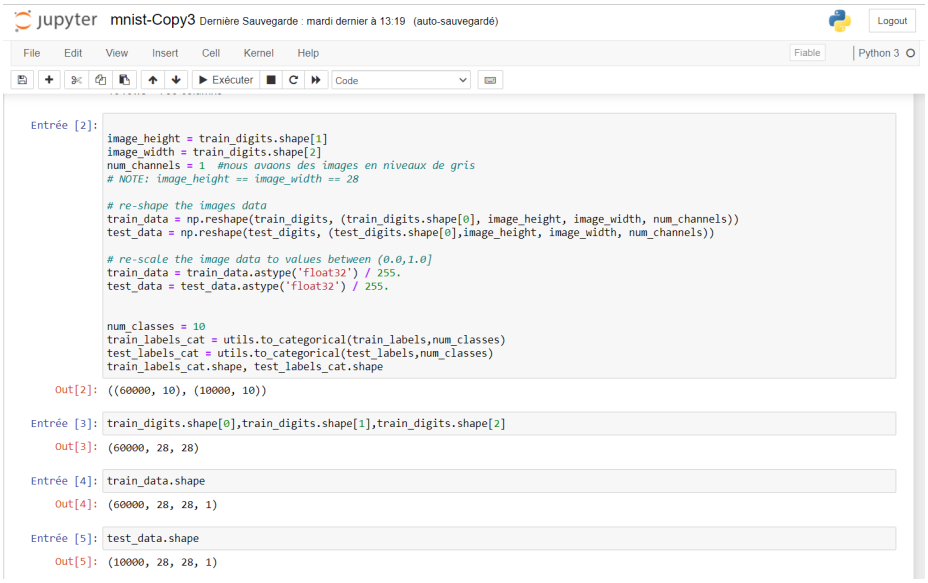


FIGURE 4.4 – Interface de Jupyter Notebook.

4.2.5 Python

Python est un langage de programmation de haut niveau, interprété, et orienté objet avec une sémantique dynamique. C'est un langage simple et permet une bonne réduction du cout de la maintenance des codes. Ainsi, il est le langage le plus utilisé dans le domaine du Machine Learning. Il est classé parmi les premiers langages de programmation grâce à la force de ses packages et librairies d'analyse bien testés. En effet, il dispose d'une multiplicité de bibliothèques pour les différents besoins du programmeur. Nous aborderons dans la suite de ce chapitre quelques bibliothèques que nous avons utilisé.

Caractéristiques :

✓ **Typage dynamique :** Pas besoin de définir le type des variables, des arguments ou des types de retour des fonctions, il est inféré à l'exécution.

✓ **Gestion automatique de la mémoire :** Pas besoin d'allouer ou de désallouer explicitement pour les variables et les tableaux de données.

✓ **Interprété :** Pas besoin de compiler le code, l'interpréteur Python lit et exécute directement le code python.

FIGURE 4.5 – Logo Python.

4.2.6 Tera term

Tera Term (figure 4.6) est un programme d'émulation de terminal (communications) open source, gratuit et mis en œuvre par logiciel. Il émule différents types de terminaux informatiques et prend en charge les connexions TCP/IP (telnet , SSH1 , SSH2) ainsi que les connexions de port série sur UART. Dans la cadre de notre travail, nous avons utilisé ce logiciel pour des fins de débogage et de la vérification de bon fonctionnement du notre code C.

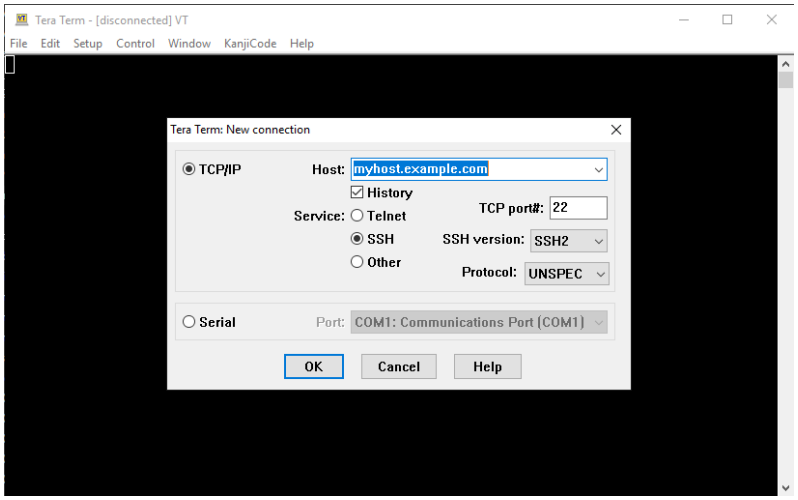


FIGURE 4.6 – Interface de Tera Term.

4.2.7 TensorFlow

TensorFlow est une plate-forme Open Source de bout en bout dédiée au machine learning, plus précisément, au calcul et à l'optimisation des opérations complexes, ce qui permet d'accélérer le processus d'apprentissage de nos modèles. Elle propose un écosystème complet et flexible d'outils, de bibliothèques et de ressources permettant de développer et d'exécuter des applications de Machine Learning et de Deep Learning.

Dans le cadre de notre travail nous avons utilisé la version TensorFlow 2.1.0 avec la bibliothèque keras 2.2.4 du moment que la version de keras supportée par STM32CubeMX est keras 2.3.1.

FIGURE 4.7 – Logo TensorFlow.

4.2.8 Les bibliothèques utilisées

➢ **NumPy :**

C'est une bibliothèque de calcul scientifique permettant d'effectuer des calculs numériques avec Python. Elle permet une exécution facile et efficace de calculs sur de grands tableaux et matrices.

➢ **Matplotlib :**

C'est une bibliothèque de traçage pour le langage de programmation Python et son extension mathématique numérique NumPy. Elle est destinée à tracer et visualiser des données sous formes graphiques. Nous avons utilisé cette bibliothèque pour tracer les deux courbes d'évolution de la précision et de la perte de notre Modèle CNN vu dans le chapitre précèdent.

➢ **Keras :**

C'est une API (Application Programmable Interface) open source, conçue pour l'apprentissage en profondeur, écrite en python. Elle permet d'interagir avec les algorithmes de réseaux de neurones profonds et d'apprentissage automatique. Elle est capable de s'exécuter sur la plate-forme d'apprentissage automatique Tensorflow, elle permet de simplifier l'utilisation des fonctionnalités offertes par cette dernière et d'accélérer le développement d'application de Deep Learning, en facilitant la création et la formation de nombreux types de réseaux de neurones. C'est-à-dire au

lieu d'effectuer des calculs manuels nous définissons simplement notre architecture réseau à l'aide d'une API conviviale, puis nous y alimentons des données d'entraînement.

4.3 Configuration du projet

Après avoir créé un modèle de reconnaissance de chiffres manuscrit vu dans le chapitre précédent, nous arrivons maintenant à l'implémentation de notre modèle sur la carte STM32F429I par le biais de l'environnement de développement STM32CubeMX.
Nous devons à présent effectuer les configurations nécessaires pour le bon fonctionnement du système. Mais avant cela, nous vérifions d'abord si la taille de notre modèle est compatible avec l'espace de stockage du µC. Pour ce faire, STM32CubeMX nous offre la possibilité d'analyser le modèle et de voir s'il est adapté à l'UC choisi. Quand c'est nécessaire, nous pouvons procéder à une compression du modèle pour minimiser sa taille et occuper moins d'espace mémoire.

La figure 4.21 montre le résultat de l'analyse de la taille du notre modèle :

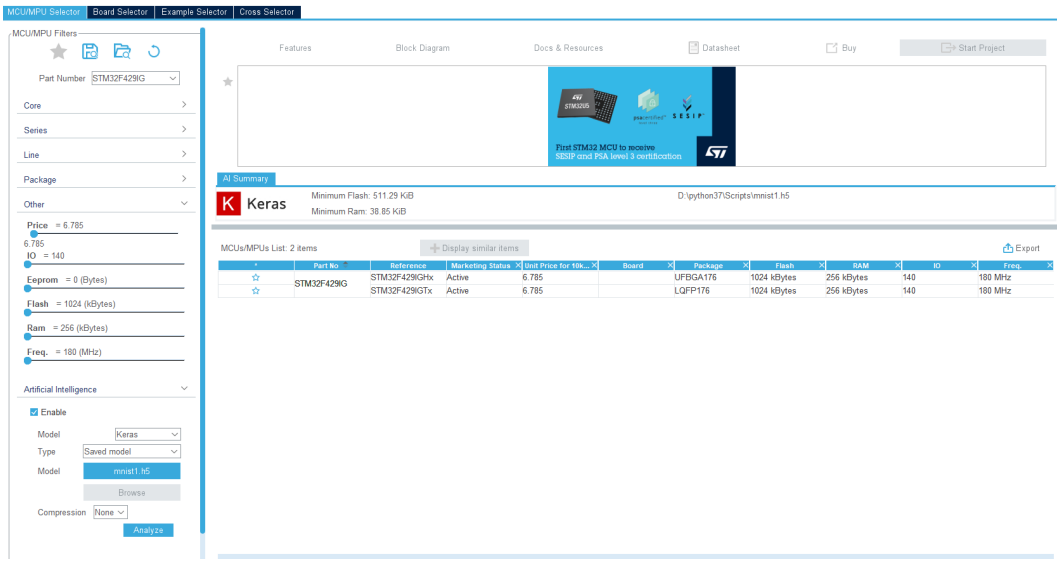


FIGURE 4.8 – Analyse de la taille du modèle.

4.3.1 Configuration d'horloge

Pour configurer les horloges, on sélectionne l'onglet **Clock Configuration** et on choisit l'horloge HSE réglée à 8MHz comme source d'horloge et on fixe l'horloge HCLK à 168MHz en gardant la valeur des présalaires par défaut. On aura automatiquement la valeur des horloges des bus des périphériques APB1 et APB2, comme montré dans la figure 4.9 :

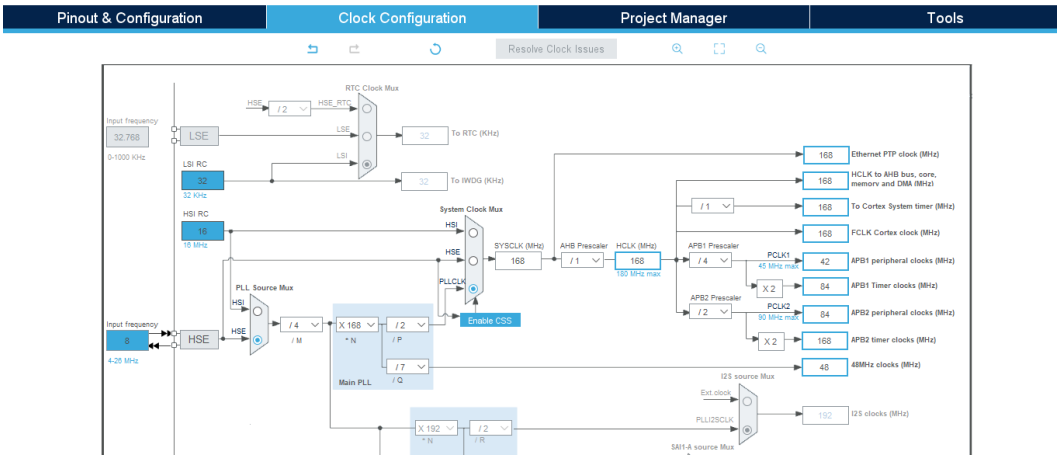


FIGURE 4.9 – Configuration des horloges.

4.3.2 Configuration de l'USART

Pour la configuration de l'usart (La figure 4.10), sous l'option **Catégories**, on sélectionne **USART1**, sous le menu **Connectivity** , on choisit le mode **Asynchronous** et on garde les valeurs des autres paramètres par défaut.

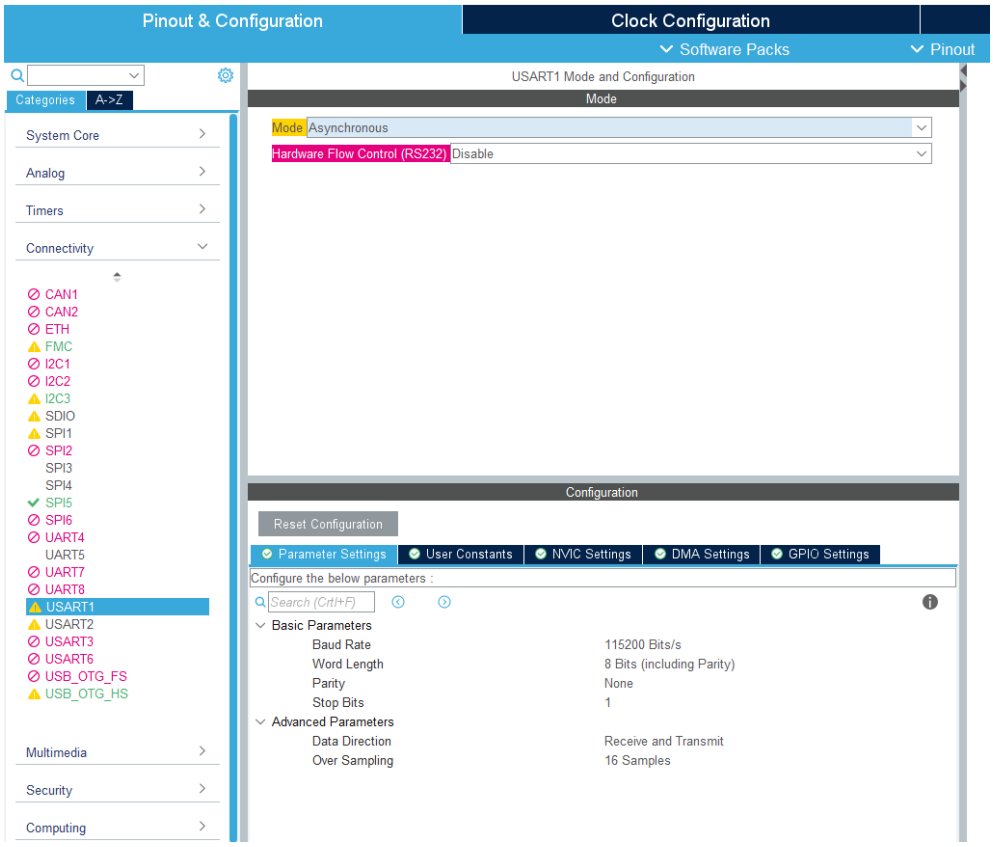


FIGURE 4.10 – Configuration de l'USART.

Dans l'option **GPIO Settings**, on retrouve les pins utilisés par l'**USART1**(Figure 4.11). Pour assurer une communication USART nous avons relié respectivement les pins PA9 et PA10 de la carte STM32F429I-DISCOVERY aux pins RXI et TXD du tranceiver USB TO UART. Ainsi que la masse GND.

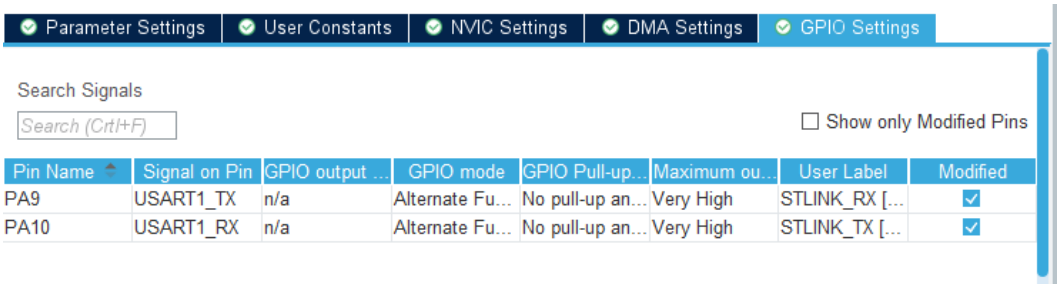


FIGURE 4.11 – les GPIOs de l'USART1.

4.3.3 Configuration de SPI

Pour l'utilisation de l'afficheur tactile de la carte STM32F429I-Discovery, nous avons utilisé le protocole SPI. Dans le menu **Connectivity** on sélectionne **SPI5** avec le mode **FULL-Duplex-Master** et nous gardons les valeurs des paramètres par défaut (Figure 4.12).

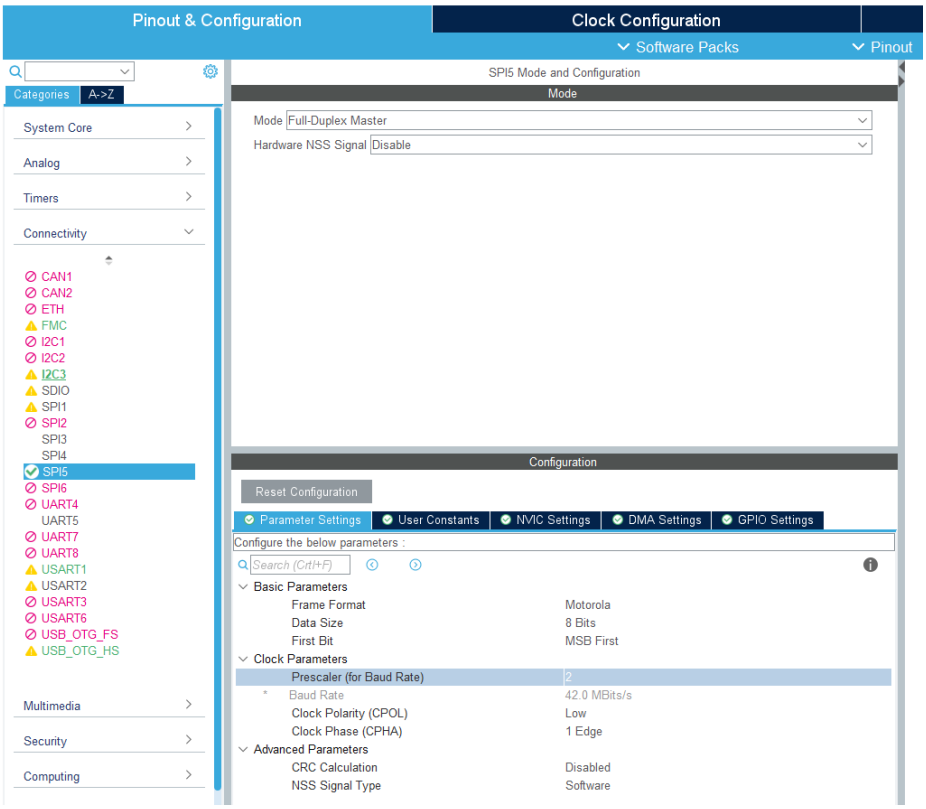


FIGURE 4.12 – Configuration de la fonction SPI.

En activant le SPI5, on aura 3 pins actives (Figure 4.13), dont les pins PF8 et PF9 qui sont reliées respectivement aux broches MISO et MOSI de l'interface SPI. Ainsi que la pin PF7 qui est relié à la broche d'horloge SCK qui sont montrés dans la figure 4.13 :

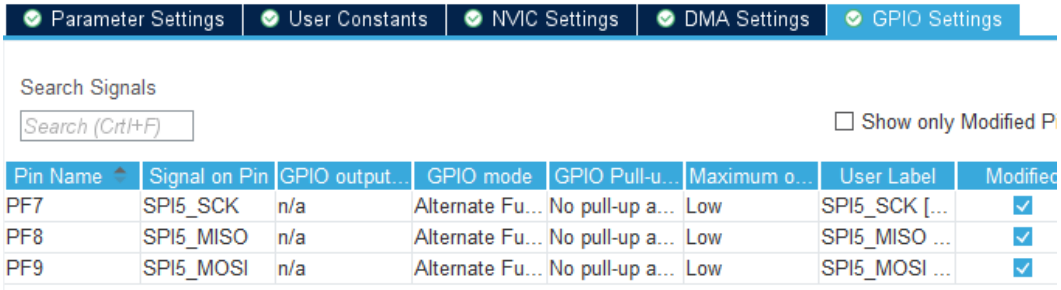


FIGURE 4.13 – les GPIOs de SPI5.

4.3.4 Configuration de I2C

Dans le même menu **Connectivity** on choisit l'option **I2C3** avec le mode **I2C** en gardant les valeurs des paramètres par défaut.(Figure 4.14).

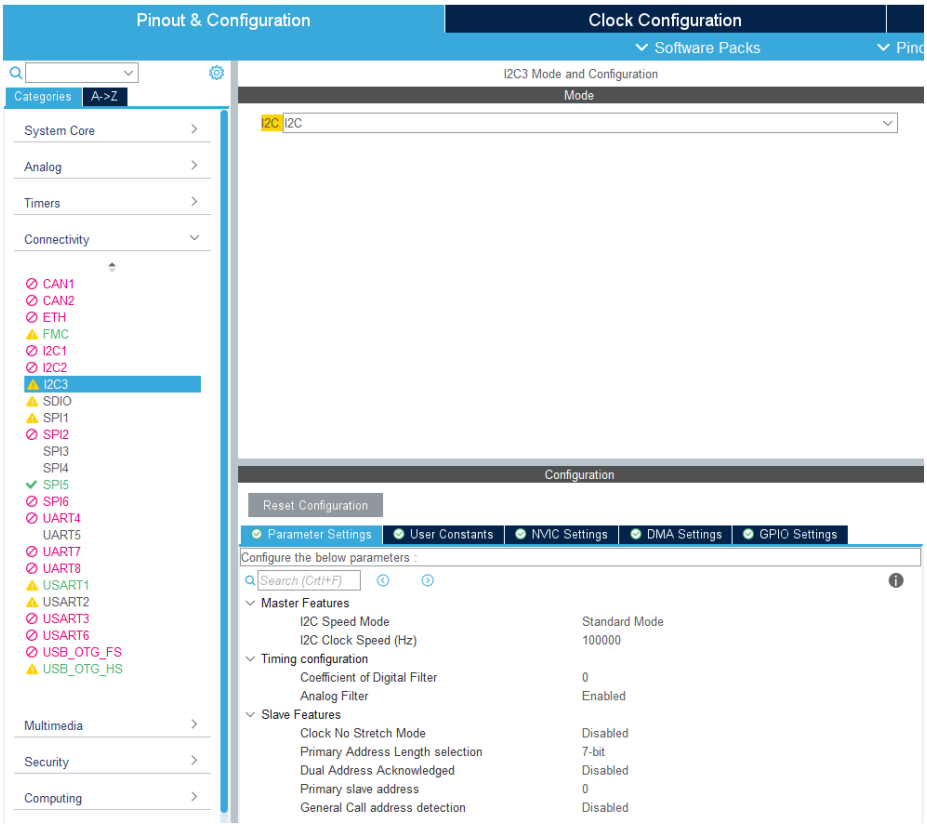


FIGURE 4.14 – les Configuration de l'interface I2C.

En activant la fonction I2C, on aura deux pins actives comme le montre la figure 4.15. Les

pins PA8 et PC9 sont relié respectivement aux broches $I2C3\_SCL$ (clock) et $I2C3\_SDA$ (data) de l'interface du bus I2C3.

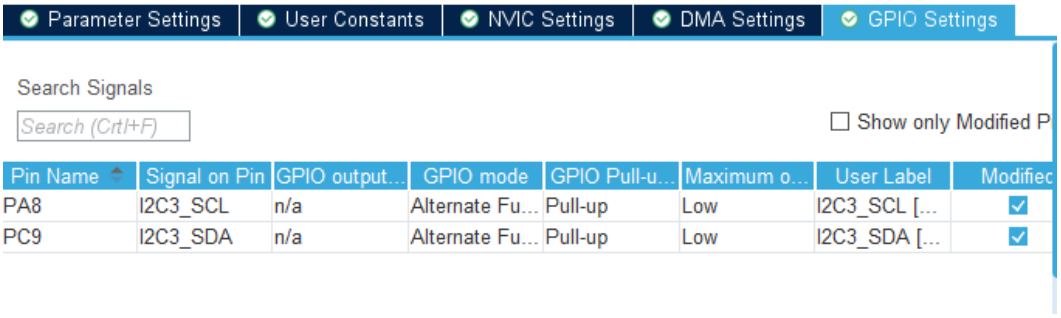


FIGURE 4.15 – les GPIOs de I2C3.

4.3.5 Configuration de FMC

Nous avons configuré le périphérique FMC pour la gestion de la mémoire SDRAM. Aussi, pour faire fonctionner l'afficheur graphique de la carte STM32F429I-Discovery.On sélectionne l'option FMC sous le menu **Connectivity**. Nous gardons les valeurs des paramètres par défaut (Figure 4.16).

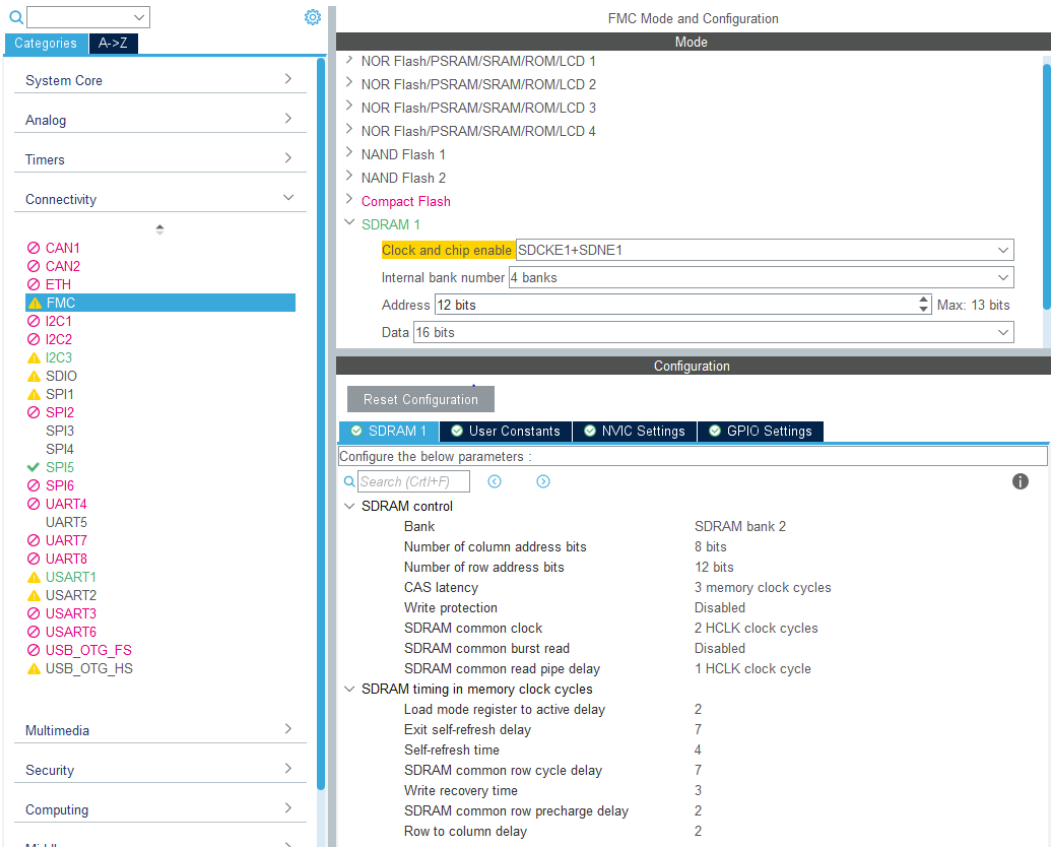


FIGURE 4.16 – Configuration FMC.

La figure 4.17 montre les différentes pins actives lors de la configuration du périphérique FMC :

SDRAM 1 | User Constants | NVIC Settings | GPIO Settings

Search (Crtl+F) — Show only Modified Pins

Pin Name	Signal on Pin	GPIO output...	GPIO mode	GPIO Pull-u...	Maximum o...	User Label	Modified
PB5	FMC_SDCK...	n/a	Alternate Fu...	No pull-up a...	Very High	SDCKE1	☑
PB6	FMC_SDNE1	n/a	Alternate Fu...	No pull-up a...	Very High	SDNE1 [SD...	☑
PC0	FMC_SDN...	n/a	Alternate Fu...	No pull-up a...	Very High	SDNWE	☑
PD0	FMC_D2	n/a	Alternate Fu...	No pull-up a...	Very High	D2	☑
PD1	FMC_D3	n/a	Alternate Fu...	No pull-up a...	Very High	D3	☑
PD8	FMC_D13	n/a	Alternate Fu...	No pull-up a...	Very High	D13	☑
PD9	FMC_D14	n/a	Alternate Fu...	No pull-up a...	Very High	D14	☑
PD10	FMC_D15	n/a	Alternate Fu...	No pull-up a...	Very High	D15	☑
PD14	FMC_D0	n/a	Alternate Fu...	No pull-up a...	Very High	D0	☑
PD15	FMC_D1	n/a	Alternate Fu...	No pull-up a...	Very High	D1	☑
PE0	FMC_NBL0	n/a	Alternate Fu...	No pull-up a...	Very High	NBL0 [SDR...	☑
PE1	FMC_NBL1	n/a	Alternate Fu...	No pull-up a...	Very High	NBL1 [SDR...	☑

FIGURE 4.17 – Les GPIOs du FMC.

Le schéma de la figure 4.18 montre une vue globale de l'ensemble de configurations du projet :

FIGURE 4.18 – vue globale du système.

4.4 Implémentation du RNA sur l'µC

4.4.1 Configuration de package IA

Dans l'onglet **Pinout & Configuration** on sélectionne **Select Component** sous le menu **Software Packs** et on choisit le package X-CUBE-AI version 5.0.0. (voir figure 4.19)

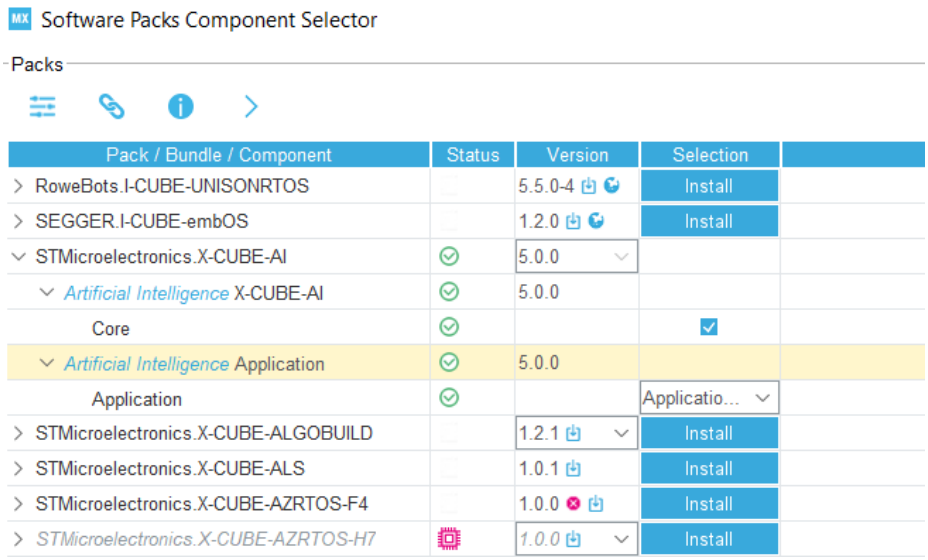


FIGURE 4.19 – Configuration de package IA.

4.4.2 Importation du modèle Keras :

Apres avoir configuré le package X-CUBE.AI, une nouvelle option **Software Packs** s'offre à nous dans le menu **Categories** de l'onglet **Pinout & Configuration** (Figure 4.20). On sélectionne **STMicroelectronics X-CUBE-AI** et on importe notre modèle. Cette étape permet la conversion de notre modèle RNA pré-entrainé en code optimisé pouvant s'exécuter sur le MCU.

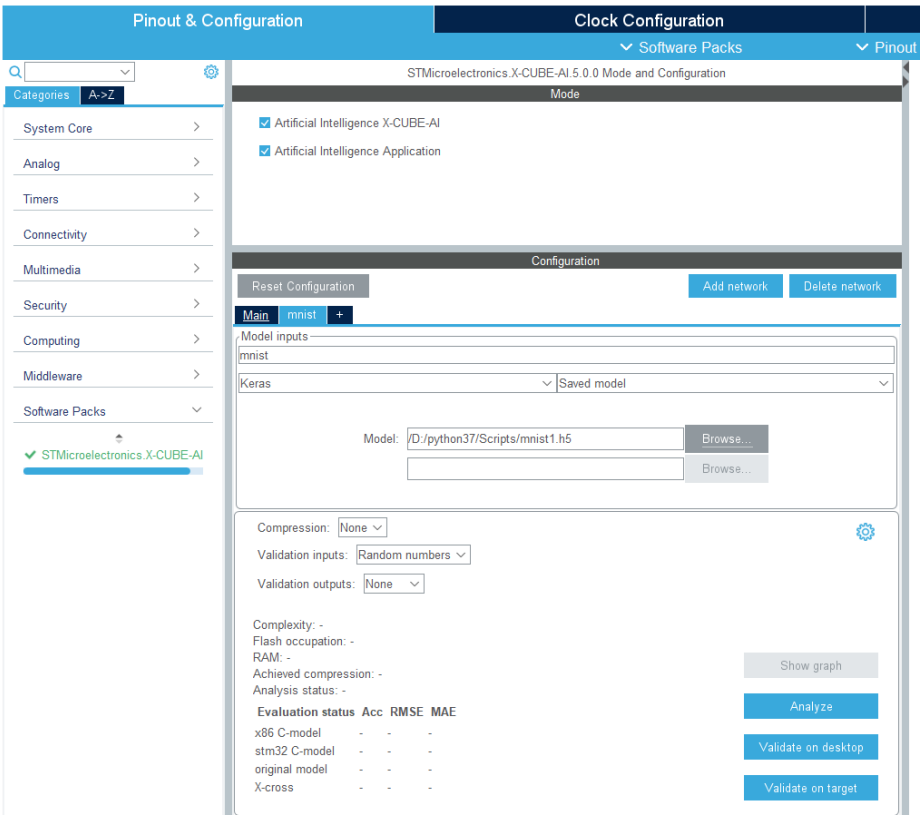


FIGURE 4.20 – Importation du modèle Keras.

Maintenant, nous optons pour une analyse du modèle, ce qui nous donne le résultat montré dans la figure 4.21 :

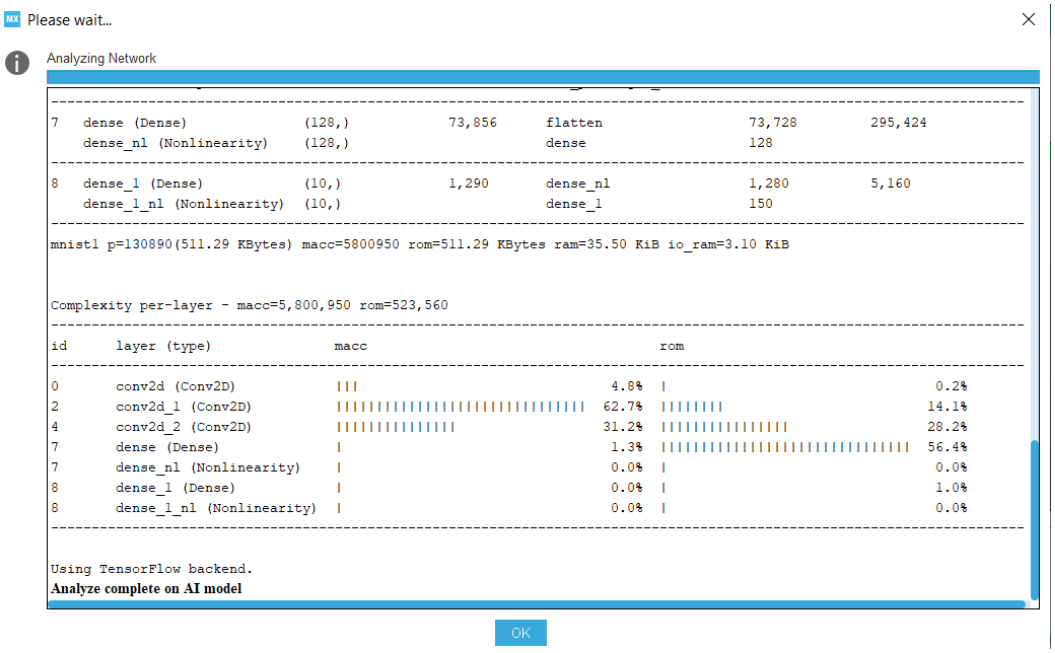


FIGURE 4.21 – Analyse du modèle.

La figure 4.22 montre les exigences matérielles de notre modèle. Ces exigences étant satisfaites par notre matériel, nous pouvons donc embarquer notre modelé sur ce dernier.

Main | mnist | +

Model manager

Name	RAM	Flash	Complexity
mnist	36.35 KBytes	523.56 KBytes	5800950 MACC
Total (1)	**36.35 KBytes**	**523.56 KBytes**	**5800950 MACC**

FIGURE 4.22 – Les exigences matérielles du modèle.

4.5 Interface du système

Afin de faciliter l'utilisation de l'interface de notre système 4.23), nous avons divisé l'écran tactile de la carte STM32F429I-DISCOVERY en 2 parties. La première partie occupe 90% de l'afficheur, dédiée à l'écriture manuscrite d'un chiffre par l'utilisateur. Nous récupérons ce chiffre sous forme d'une image de dimension 240X240 pixels. La partie restante affiche une reproduction de cette transcription sous un format réduit (28X28 pixels), qui est la taille des images de notre modèle d'entrainement. En bas de cette interface nous avons mets deux boutons tactiles « PREDICT » et « ERASE ». En cliquant sur le bouton « PREDICT » on aura la valeur du chiffre prédit à côté du chiffre manuscrit. Tandis que le bouton « ERASE » permet d'effacer le chiffre écrit pour réaliser une nouvelle reconnaissance.

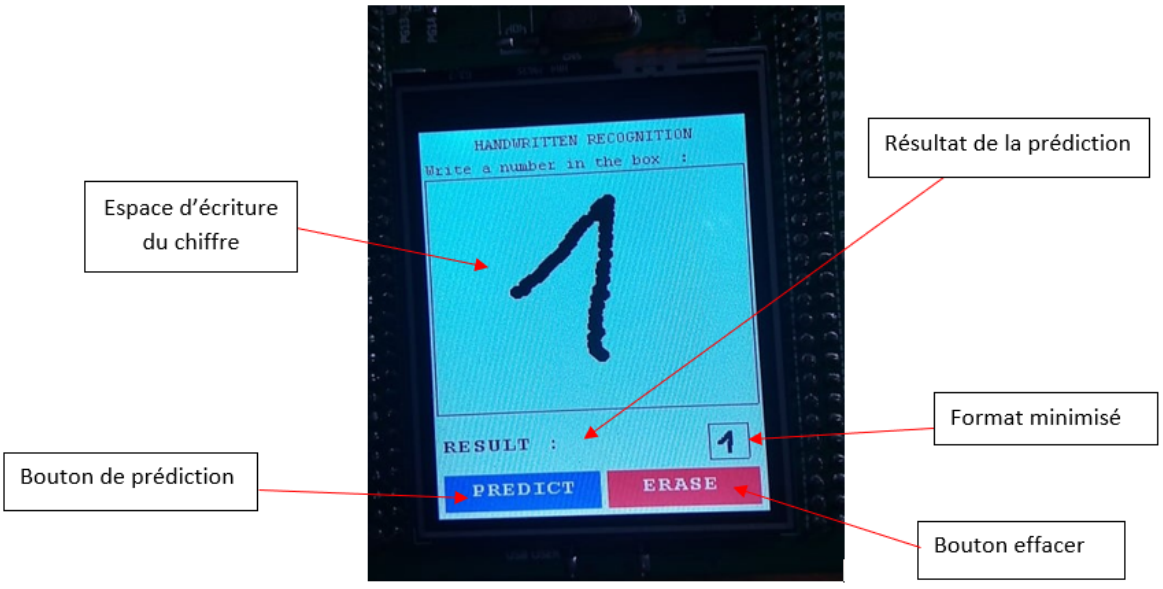


FIGURE 4.23 – Interface du système.

La figure 4.24 représente les outils matériels de notre système. Dont la carte STM32F429I-Discovery reliée à notre PC via un câble, le tranceiver USB TO UART assurant la liaison série USART.

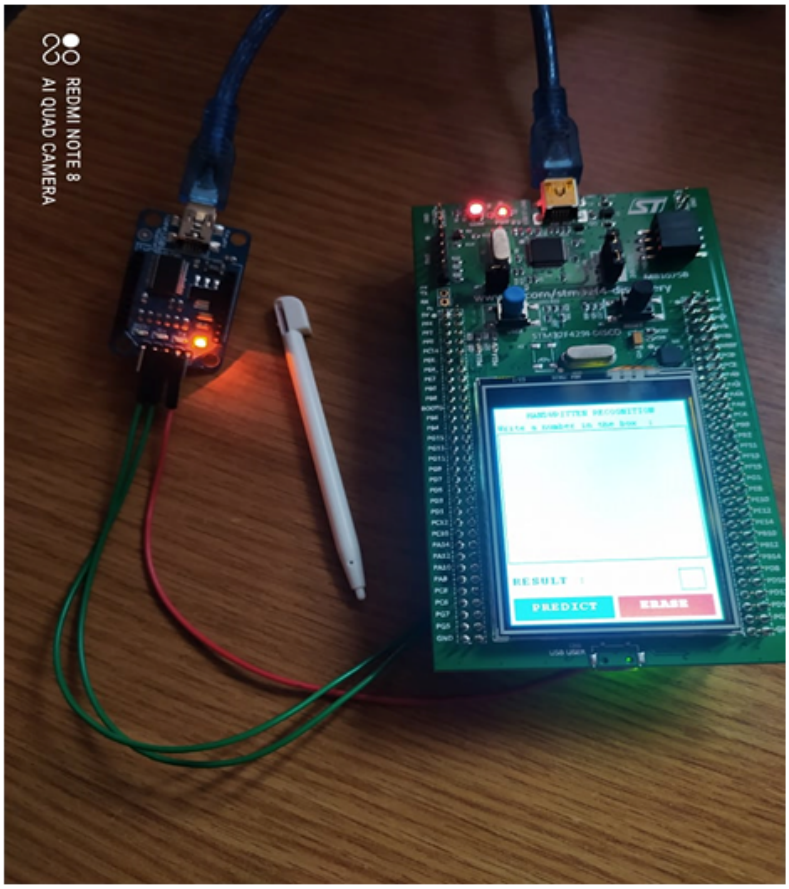
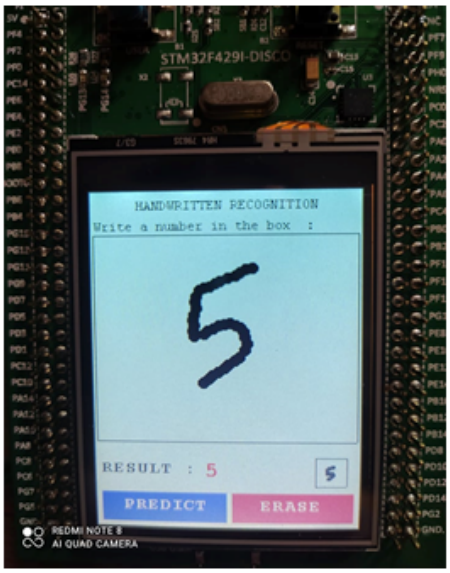
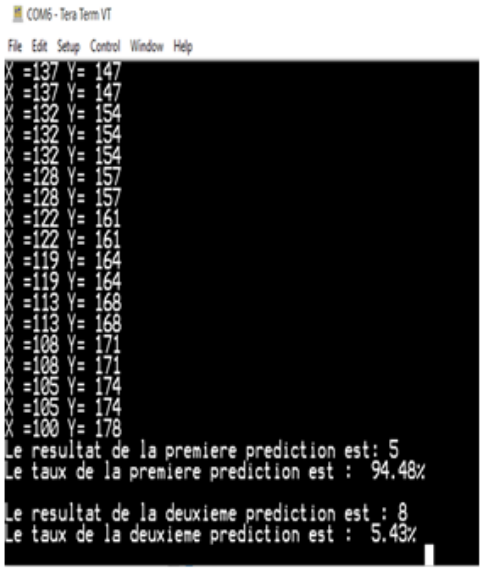


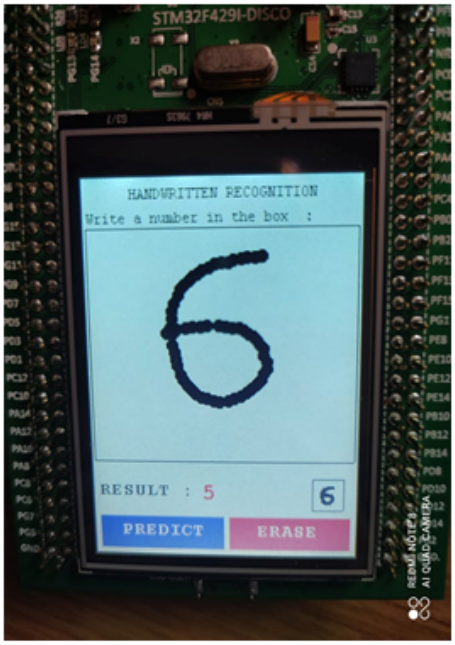
FIGURE 4.24 – les outils matériels du système.

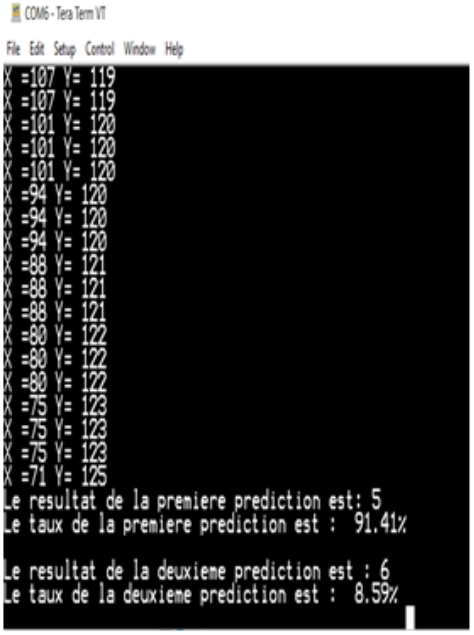
Pour améliorer encore plus la précision, on effectue deux prédictions consécutives et on choisit celle qui a le taux de prédiction le plus élevé. On affiche sur le logiciel d'émulation du terminal TeraTerm la suite des (X, Y) représentant les coordonnés de notre doigt sur l'écran tout au long de l'écriture du chiffre et les deux prédictions avec leurs taux correspondant. Ensuite, on affiche sur l'interface du notre système le chiffre qui a la plus grande valeur de prédiction.

Les figures suivantes représentent un ensemble de résultat obtenu lors de la prédiction de quelques chiffres manuscrits :









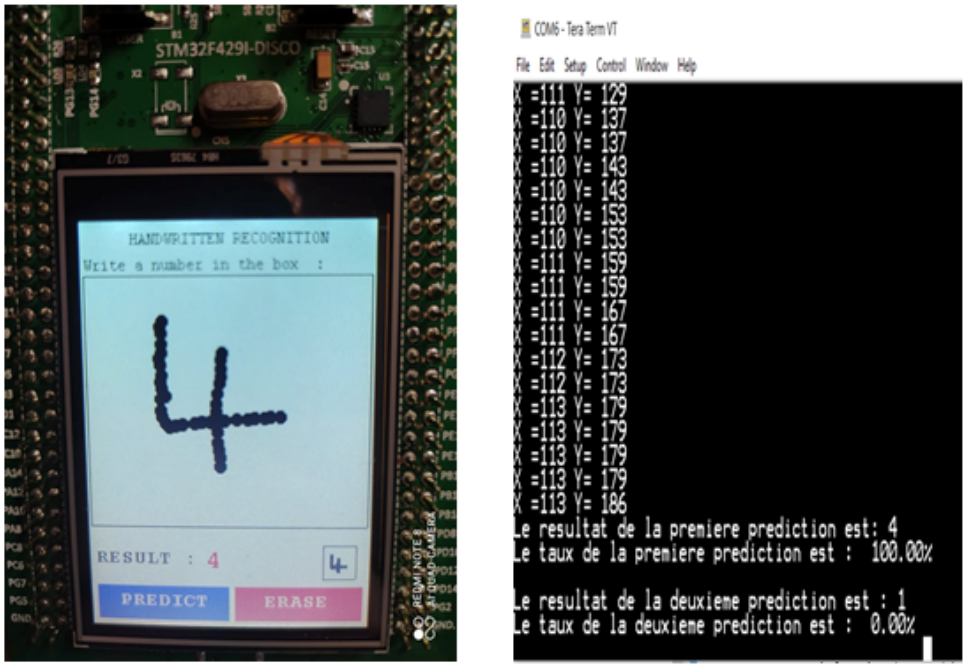


FIGURE 4.25 – Les résulatas de prédiction de quelques chiffres manuscrits.

Conclusion

Dans ce chapitre, nous avons abordé la partie implémentation de notre système. Nous avons présenté la carte STM32F429I-Discovery et ses différentes caractéristiques et fonctionnalités. Nous avons aussi décrit l'environnement de développement ST32CubeMX et son package STM32Cube.AI qui permet de maximiser les performances et minimiser l'empreinte mémoire des réseaux neuronaux sur les cœurs de processeur Arm Cortex-M.

Par la suite nous avons effectué les configurations nécessaires de la carte STM32F429I-Discovery pour pouvoir implémenter notre modèle de reconnaissance de chiffres manuscrits. Ainsi nous avons créé notre système de reconnaissance embarqué sur la carte STM32. Ce système est doté d'une interface tactile pour permettre de faciliter son utilisation.

CONCLUSION GÉNÉRALE

Dans le cadre de notre travail, nous avons pu embarquer un modèle du deep Learning basé sur les réseaux de neurones artificiels sur une carte de développement STM32F429I-Discovery. Pour ce faire, nous avons eu recours à plusieurs notions et concepts notamment en Intelligence artificielle et les fondamentaux des réseaux de neurones. Ainsi, nous avons pris comme exemple d'application : la reconnaissance de chiffres manuscrits.

Dans un premier lieu, nous avons pu créer un modèle de réseau de neurones convolutifs basé sur le principe de la classification des images numériques. Nous avons utilisé la base de données MNIST comme ensemble d'apprentissage de notre modèle. L'entrainement de ce dernier nous a permis d'atteindre un taux de précision supérieur à 90%. Par la suite nous avons réussi à implémenter notre réseau de neurone sur la carte STM32F429I-Discovery. Ceci est fait grâce à la bibliothèque X-Cube.AI qui permet la conversion de notre modèle sous un format compatible avec le microcontrôleur 32 bit à cœur ARM Cortex-M4. Enfin, nous avons réalisé un système de reconnaissance de chiffres manuscrits en utilisant l'afficheur graphique de la carte STM32F429I-Dicsovery. Ce système permet à l'utilisateur d'écrire un chiffre à la main et le reconnaitre.

Notre travail est purement à but pédagogique. Il nous a permis de comprendre les concepts de l'IA, du Deep Learning et des problèmes d'implémentation de ce genre d'outils sur des systèmes embarqués. Toutefois, nous pouvons dire qu'il peut être exploitable dans le monde réel dans les domaines où la saisie des chiffres manuscrits est plus pratique. Comme par exemple :

- Saisie manuscrite d'un numéro de téléphone.

- Tri postal automatique.

Néanmoins, il reste ouvert pour des améliorations. Ceci peut se faire grâce à la flexibilité du microcontrôleur et l'évolution des systèmes embarqués. A titre indicatif, nous pouvons développer les points suivants :

- Amélioration du notre modèle de reconnaissance en utilisant une base de données plus volumineuse et plus variées pour pouvoir afficher un message d'erreur dans le cas où on écrit rien sur l'afficheur tactile de la carte.

- Transformation du modèle du réseau de neurone pour réaliser un système de reconnaissance de caractère (chiffre, lettre, symbole, ...).

BIBLIOGRAPHIE

[1] I.Zara, L'intelligence artificielle principe, outils et Objectifs, Mémoire de master UNIVERSITE BADJI MOKHTAR ANNABA, 2019.

[2] H. AIT ISSAD, Machine Learning, support de cours de programmation MATLAB Master2 RMSE, 2020/2021.

[3] S. Russel et P. Norvig, Intelligence artificielle, 3eme édition, 2010.

[4] Hubspot, Deep learning, Définition, Fonctionnement, Domaines d'application. https ://www.blog.hubspot.fr/marketing/deep-learning, 2021.

[5] Digital Guide IONOS https ://www.ionos.fr/digitalguide/web-marketing/search-engine-marketing/deep-learning-vs-machine-learning, 2020.

[6] Deep learning sur AWS https ://aws.amazon.com/fr/deep-learning/.

[7] H. ANOUAL, Modélisation, Observation et Commande d'un Drone Miniature Birotor Coaxial, thèse de Doctorat, 2012.

[8] F. KADA et A. ZAHAF , Implémentation des algorithmes de traitement des images et vidéos en utilisant la bibliothèque OpenCV , Mémoire de Master, Université Abou Bakr Belkaid de Tlemcen, 2016 .

[9] N. MERABET et M MAHLIA, Recherche d'images par le contenu, mémoire de Master Université Abou Bakrbelkaid–Tlemcen, 2011.

[10] D. Boukhlouf, Généralités sur le traitement d'images, Mémoire de Master, 2005.

[11] H. Guedmim, R. TAIB, Conception et réalisation d'une serre agricole connectée, mémoire de Master, Université Mouloud MAMMERI de Tizi-Ouzou, 2020.

[12] Classifiez les images à l'aide de réseaux de neurones convolutifs. https ://www.openclassrooms.com/fr/courses/4470531-classez-et-segmentez-des-donnees-visuelles/5082166-quest-ce-quun-reseau-de-neurones-convolutif-ou-cnn

[13] S.AIT TAHAR et D. OUDIR, Classification d'images à l'aide des réseaux de neurones convolutif, mémoire de Master, Université Mouloud MAMMERI de Tizi-Ouzou, 2017.

[14] S. Haitaamar, Segmentation de texte en caractère pour la reconnaissance optique de l'écriture arabe, thèse doctorat Université EL-HADJ LAKHDHAR Batna, 2007.

[15] A. Boucher et N.A. Tuan, Reconnaissance d'écriture manuscrite, 2005.

[16] L. Deng, The MNIST Database of Handwritten Digit Images for Machine Learning Research, IEEE Signal Processing Magasine, p. 141-142, 2012.

[17] Discovery kit with STM32F429ZI MCU Data brief. https ://www.alldatasheet.com/datasheet-pdf/pdf/926516/STMICROELECTRONICS/STM32F429I-DISC1.html

[18] M. DAOUI, Les interfaces série UART,support de cours systèmes embarquées,2020.

[19] C. HEMDANI, Systèmes embarqués avec les microcontrôleurs STM32, Les timers, support de cours Master 2 RMSE, 2019.

[20] Page officielle dédiée au package logiciel STM32CubeF4. http ://www.st.com/en/embedded-software/stm32cubef4.html