REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITE YAHIA FARES DE MEDEA

Faculté de Technologie

Département du Génie Electrique

Mémoire de fin d'études de Master

Filière : *Télécommunications.*

Spécialité : *Systèmes de Télécommunications.*

Thème :

Implémentation d'un système de classification des images de digits manuscrits par CNN sur la carte PYNQ-Z1.

Présentés par:

- *Taieb Errahmani Chahinez*
- *Euldji Manel*

Proposés et Dirigés par :

Dr. DALI Mohammed.

Promotion 2024



Remerciement

Nous exprimons notre gratitude envers le Tout-Puissant, Allah, pour nous avoir accordé la volonté, une santé optimale et la force nécessaire pour mener à bien ce travail.

Tout d'abord, nous voulons remercier Dieu de nous avoir aidés et de nous avoir donné patience et patience le courage de terminer ce travail.

Bien sûr, nous commençons par exprimer notre plus profond respect et notre gratitude Nous remercions notre encadrant Dr. DALI Mohammed pour son encadrement et ses conseils faites le travail patiemment.

Nous tenons à remercier les membres du jury de nous avoir accordé le privilège de juger ce travail.

Nous tenons également à remercier tous les enseignants du Département de génie électrique de l'Université de Médéa ainsi que nos familles, en particulier nos parents, pour leur soutien inconditionnel.

Et nous n'oublions pas nos frères et amis proches qui étaient toujours là et ne se sont jamais arrêtés encourager.



Dédicace

Tout d'abord, je remercie le Dieu, notre créateur de m'avoir donné la force, la volonté et le courage afin d'accomplir ce travail modeste.

*Je dédie ce travail à ma mère « **ZAHRA** », la source de tendresse et la lumière qui guide mes routes et qui m'emmène aux chemins de la réussite, pour tous ses sacrifices consentis et ses précieux conseils, pour toute son assistance et sa présence dans ma vie.*

*À mon père « **NACER EL DIN** » que je le remercie énormément pour ses efforts, ses conseils et sa surveillance.*

*Sans oublier ma grand-mère « **FARODJA** » que j'aime .*

*A mon cher frère **chafie ,zohir***

*À mon meilleure amie: **FEDOUA, EL BATOUL, IMANE, IBTISAM et HOUDA.***

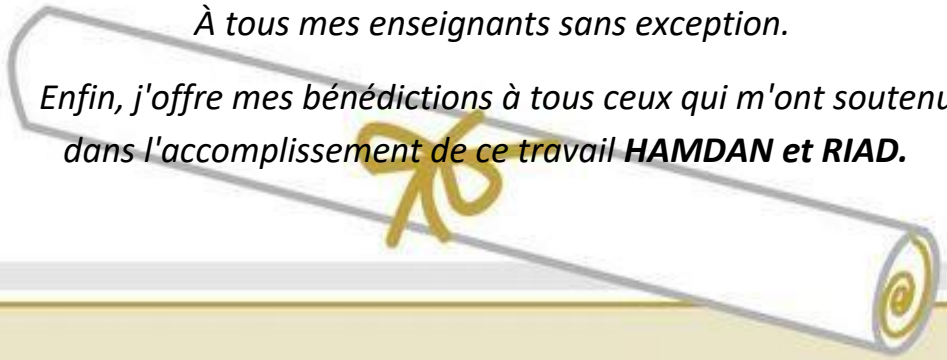
*Pour toute la famille **TIEB ERRAHMANI** et **CHAÂBEN**.*

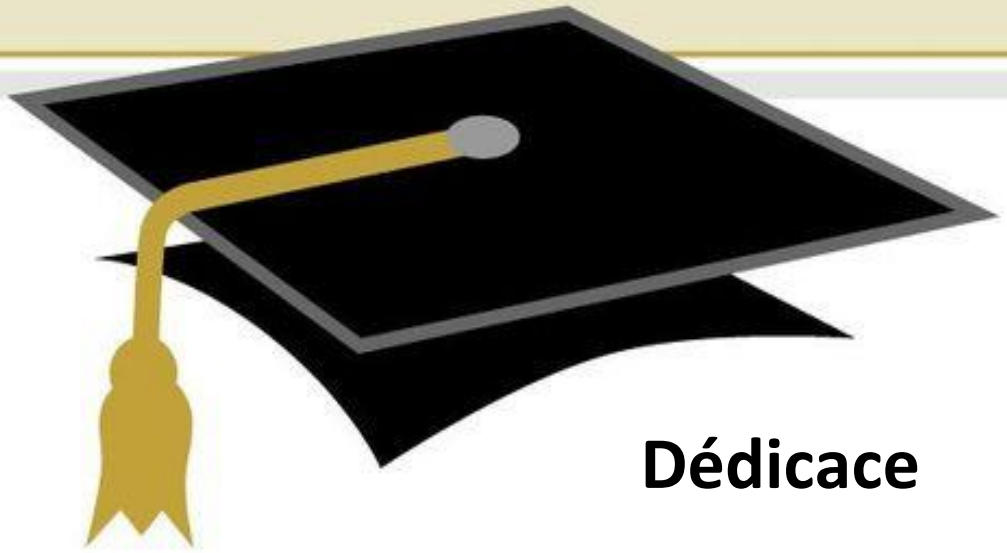
A tous mes amis de promo Télécom 2024.

À tout ce que je connais sans exceptions.

À tous mes enseignants sans exception.

*Enfin, j'offre mes bénédictions à tous ceux qui m'ont soutenu dans l'accomplissement de ce travail **HAMDAN et RIAD.***





Dédicace

Je dédie ce projet :

A ma chère mère,

A mon cher père,

Qui n'ont jamais cessé, de formuler des prières à mon égard, de me soutenir Et de m'épauler pour que je puisse atteindre mes objectifs.

A mon cher frère

A mes chères sœurs

Pour ses soutiens moraux et leurs conseils précieux tout au long de mes études.

A mon cher binôme

Pour sa entente et sa sympathie

A mes chères ami(e)s

Pour leurs aides et supports dans les moments difficiles ***A toute ma famille,***

A tous ceux que j'aime et ceux qui m'aiment.



Manel

ملخص

الشبكات العصبونية الإلتفافية هي شبكات عصبونية متعددة الطبقات وتعتبر نماذج قوية تستخدم لتصنيف الصور في عدة مجالات من رؤية الحاسوب مثل التعرف التلقائي على الأرقام. تعتمد هذه النماذج على عمليات التجميع التراكمي والجداء عبر مصفوفات كبيرة مما يتطلب معالجات قوية ومساحة ذاكرة كبيرة لتنفيذها. إحدى الحلول الفعالة تتمثل في استخدام الدوائر عالية الأداء والمرنة أو القابلة لإعادة التشكيل، مثل دوائر ZYNQ.

في عملنا، قمنا بتصميم وتنفيذ تطبيق لتصنيف الأرقام المكتوبة يدويا باستخدام شبكة **CNN** على لوحة التطوير **PYNQ-Z1**، التي تعتمد على دائرة ZYNQ , تظُهر نتائج التنفيذ أن استخدام إستراتيجية الحساب التسلسلي وتمثيل البيانات بالنقطة الثابتة يسمح بتنفيذ فعال من حيث استخدام الموارد واستهلاك الطاقة على لوحةPYNQ-Z1 .

الكلمات المفتاحية: تصنيف، شبكات (CNN)، أرقام مكتوبة يدوياً، بايثون،System Generator Vivero، ZYNQ ، PYNQ-Z1،

Abstract:

Convolutional Neural Networks (CNNs) are a powerful technique used for image classification in various computer vision domains, such as automatic digit recognition. This technique relies on applying convolutions to a large number of matrices, which requires powerful processors and significant memory. An effective solution for their implementation is to use high performance and flexible or reconfigurable circuits, such as ZYNQ circuits.

In our work, we designed and implemented an application for handwritten digit classification using a CNN on the PYNQ-Z1 development board, which is based on a ZYNQ circuit. The implementation results show that using a serial computation strategy and fixed-point data representation allows for an efficient implementation in terms of resource usage and power consumption on the PYNQ-Z1 board.

Keywords: **classification, CNN, handwritten digits, PYTHON, PYNQ-Z1, System Generator, Vivado ZYNQ**

Résumé :

Les Réseaux de Neurones Convolutifs (CNN) est une technique puissante utilisée pour la classification d'images dans divers domaines de la vision par ordinateur comme la reconnaissance automatique de chiffres. Elle repose sur le calcul de convolutions appliqué à un grand nombre de matrices, ce qui requiert des processeurs puissants et une mémoire

conséquente. Une solution efficace pour leur mise en œuvre consiste à utiliser des circuits performants en calcul et flexibles ou reconfigurables comme les circuits ZYNQ.

Dans notre travail, nous avons fait la conception et l'implémentation d'une application de classification des digits manuscrits par réseau CNN en utilisant la carte de développement PYNQ-Z1 qui est basée sur un circuit de type ZYNQ. Les résultats d'implémentation montrent que l'utilisation d'une stratégie de calcul série et la représentation des données en virgule fixe permettent une implémentation efficace en ressource et en puissance sur la carte PYNQ-Z1.

Mots clés : classification, CNN, digits manuscrits, PYTHON, PYNQ-Z1, System Générateur, Vivado, ZYNQ.

Table des matières

Dédicace
Remerciement
ملخص
Liste figures
Liste des tableaux
Liste des abréviations
Introduction générale…….. 02

CHAPITRE I : Classification des images par CNN

I.Introduction ……. 05
I.2 Notions élémentaire sur les images …….. **05**
I.2.1 Définition d'une image …… **05**
I.2.2 Représentation des images ……..….. **05**
I.2.3Image numérique(numérisée) ……..…..… **06**
I.2.4 Types d'images numériques ……. **07**
I.2.5Caractéristiques d'une image numérique ……... **07**
I.2.5.1La résolution ……... **08**
I.2.5.2 Dimension ……... **08**
I.2.5.3Bruit …… **08**
I.2.5.4 Histogramme …….…….…….. **08**
I.2.5.5 Luminance …….…… **09**
I.2.5.6Contraste et textures …….…… **09**
I.2.6 traitement numérique des images …….. **10**
I.2.6.1 Filtrage Numérique …….…….. **10**
I.2.6.2 Domaines d'application …….…….…….. **13**
II.La Classification d'images ……... **14**
II.1 Les étapes d'une classification …….…… **14**
II.2. Quelques méthodes classiques de classification …….……..... **15**
II.2.1 La classification SVM (Support Vector Machine) ……..….. **16**
II.2.2 Classification par k plus proches voisins (KNN) ……....…… **17**
III. Classification par réseau CNN ……...…… **18**

III.1.De l'intelligence artificielle au réseaux CNN…… **18**
III.1.1 Intelligence artificielle ……..……..... **18**
III.1.2. Le Machine Learning ML..... **19**
III .2. Réseaux de neurones artificiels …….. **23**
III.2.1 Structures d'un réseau de neurones artificielles ……..……..…….. **23**
III.2.2 La fonction de combinaison et la fonction d'activation **…….……** **24**
III.2.**3**La rétropropagation du gradient de l'erreur …….…….. **26**
III.3 Réseaux de neurones convolutifs (CNN) ……. **28**
III.3.1 Architecture de réseaux CNN …….……. .. **29**
III.3.2 Couche convolutive (Convolutional layer) …….…… **30**
III.3.3 Couche d'activation…….…….…….. **31**
III.3.4 Couche de sous-échantillonnage (Pooling layer) ……... **32**
III.3.5 Couche entièrement connectée (Fully-Connected Layer) ……..... **32**
III .4 Exemples de réseaux CNN …….. **33**
IV. Conclusion …… **37**

Chapitre II : Etude des circuits logiques programmables FPGA

I. Introduction…… **39**
II. Evolution des circuits intégrés numériques ……. **39**
II.1 Microprocesseurs ……... …….…… **39**
II.2 Les circuits programmables PLD (Programmble Logic Device) …….…… **40**
II.3 Les circuits spécifiques à l'application ASIC (Application Specific Integration circuit)……... **41**
III. Circuits logiques programmables FPGA …… **42**
III.1 Introduction …….…. **42**
III.2 Architecture des circuits FPGA …….…. **42**
III.2.1 Blocs logiques programmables …….. **43**
III.2.2 Blocs d'entrées/sorties …….. **44**
III.2.3 Réseau d'interconnexion …….. **45**
III.2.4 Blocs de mémoires et multiplicateurs …….…….……. **46**
III.3. Exemples de circuits FPGAde Xilinx-AMD …….……. **47**
III.3.1 La famille FPGA Virtex-7 …….... **48**
III.3.2 La famille Zynq-7000 …… **51**
III.3.2.a Système de traitement (processing system PS)…… **51**
III.3.2.b partie PL (Programmable Logic) ……...…... **56**
III.3.2.c Interconnexion entre PS et PL…….…….…….. **57**
III.3.3 cartes de développement PYNQ-Z1 …….…… **57**
III.3.4 Concept général du PYNQ …….……. **58**
IV. Conclusion…….……. **60**

Chapitre II : Implémentations, optimisation et résultats

I. Introduction……...…….…… **62**
II Logiciels utilisés …….……... **62**
II.1-Matlab …….…….. **62**
II.2-Simulink …….…….. **63**
II.3-Python …….…….. **63**
II.4- Xilinx System generator (XSG)..... **64**
II.5 Logiciels vivado……..…… **66**
III. Application de reconnaissance de digits manuscrit …….……. **66**
III.1 Explication de l'application …….…… **66**
III.2 La base de données MNIST …….…… **66**
III.3 Architecture du réseau CNN …….…….. **67**
III.4 Programmation, apprentissage et évaluation du réseau …….…… **69**
III.4.1 En utilisant matlab …….…….. **69**
III.4.2 En utilisant PYTHON …….…….. **73**
IV. Conception des circuits de calcul sous System Generator ……..…..…… **76**
IV.1 Circuit de calcul de la convolution 1 ……..…… **78**
IV.3 Circuit de calcul ReLU 1 et maxpoling 1 …….. **81**
IV.2. Circuit de calcul de la convolution 2 …… **81**
IV. 4 Circuit de calcul série de ReLU 2 et Max pooling 2…… **83**
IV.5 Circuit de calcul série de la convolution 3…….. **83**
IV.6 Circuit de calcul de la couche FC et classification…….. **84**
V. Résultats de l'implémentation …… **85**
V.1 Etapes d'implémentation ……. **85**
V.2 Résultats d'implémentation en virgule flottante …….. **87**
V.3 Résultats d'implémentation en virgule fixe ……. **88**
VI. Conclusion…… **90**
Conclusion générale……. **92**
Références

Liste des Tableau

Tableau I.1 :différentes dimensions des images……..... **05**

Tableau II.1 : résume les ressources disponibles pour les différentes versions de virtex-7…….…….. **50**

Tableau III.1 : Caractéristiques des différentes couches du réseau choisi… **68**

Tableau III.2 : Explication des fonctions Matlab utilisées pour la détermination des différentes couches du réseau CNN……..... **70**

Tableau III.3 : Résumé de l'architecture du réseau utilisé pour la classification de digits manuscrits …….. **75**

Tableau III.4 : Les ressources et la puissance dynamique consommées par les différents circuits de calcul implémenté sur ZYNQ XC7Z020-1CLG400C en utilisant la représentation en virgule flottante……..… **87**

Tableau III.5: Ressources et puissance consommées par les différentes couches de calcul avec les données en virgule fixe…….. **89**

Liste des figures

Figure I.1 : Matrice de représentation d'une partie d'une image numérique……..*06*
Figure I.2 *:* Exemples d'images numérique naturelle et artificielle ……*06*
Figure I.3 *:* Exemples de types d'images ……..……..…….……06
Figure I. 4:exemple de bruit dans les images (a) : image sans bruit, (b) : image avec bruit ……...08
La figure I.5 : donne l'histogramme des couleurs RVB de l'image parrots…….09
Figure I.6 : Composition d'un système de traitement numérique ……10
Figure I.7 : Principe du filtre médian……..…..…12
Figure I.8 :. Principe du filtre maximum …….....12
Figure I.9 : Principe du filtre minimum …….……....12
Figure I.10 : Exemples de Domaines d'Applications de traitement d'images…….....14
Figure I.11 : principe de classification par la méthode SVM…….…….....17
Figure 1.12 : classification par la méthode KNN…….....17
Figure I.13: les types de L'intelligence Artificielle…….…….....19
FIGURE I.14 : Apprentissage supervisé……..…..….20
FIGURE I.15 : Apprentissage non supervisé…….…….……...….20
FIGURE I.16 : Apprentissage par renforcement…….……...…….21
FIGURE I.17 : algorithmes du Deep Learning……..……22
FIGURE I.18 : Schéma d'un réseau de neurones artificiel…….…….……..….23
FIGURE I.19 : Structure d'un neurone artificiel…….……24
Figure I.20 principe de la fonction d'activation…….…..……25
FIGURE I.21**:** La rétropropagation du gradient de l'erreur ……..……...…...27
Figure I .22 : architecture de base d'un réseau CNN…….…..…29
Figure I .23 : les différentes couches d'un réseau CNN…….……...….30
Figure I.24 : illustration du principe de fonctionnement d'une couche de convolution à trois entrées et deux sorties……..……....….31
Figure I.25: les deux types de sous échantillonnage (Pooling……..…..……32
Figure I.26 : structure de la couche Fully Connected (FC)…….……...…...33
Figure I.27: architecture du réseau LeNet-5……..…...….34
Figure I .28 *: architecture du réseau*AlexNet……..……...…..……34
Figure I.29 : architecture du réseau ZF **Net**……..……....35
FigureI.30 : architecture du réseau Google **Net**……..……..…...……35
Figure I.31 : architecture du réseau VGGNet…….…….…..…36
Figure I.32 : architecture du réseau ResNet…….....…36
Figure II. 1 : Architecture globale d'un circuit PLD (a) : SPLD (b) : CPLD..…….41
Figure II.2 : Architecture générale d'un FPGA ……...….43
Figure. II.3 : Structure simplifiée d"un bloc logique programmable (CLB)…..……..…..44
Figure. II.4 : Structure d'un bloc d'entrée/sorties (IOB)……45
Figure. II.5 : Disposition des matrices d"interconnexion dans un FPGA……...……45
Figure. II.6 : Disposition des blocs mémoires et des multiplicateurs dans un FPGA ……47

Figure II.7 : Architecture interne de la famille Virtex 7 de Xilinx ……..……....49
Figure II.8 : Arrangement des slices dans un CLB pour la série 7 de Xilinx …..……....49
Figure II.9 schéma simplifié d'un circuit ZYNQ…….…….….51
Figure II.10 : architecture de la partie PS d'un circuit ZYNQ ……..……52.
Figure II .11: architecture de la partie PL des circuits ZYNQ ……..……....54
Figure II .12 : Composition d'un bloc logique configurable (CLB) de la famille Zynq-7000.....55
Figure II.13 Interface entre PS et PL……..…….…..….56
Figure II.14. Face avant (A) et face arrière (B) de la carte PYNQ-Z1.……..……..…..58
Figure II .15 : Concept général du cadre open source PYNQ.……..…….…....59
Figure III.1 Fenêtre de développement de l'outil XSG……...……65
Figure III.2 : Exemple d'mage de chiffres de la base de données MNIST…….67
Figure III.3 Architectures du réseau CNN choisie pour la reconnaissance de digits manuscrits……68
Figure III.4 Graphes affichés durant la phase d'apprentissage du réseau CNN…..……...72
Figure III.5 Schéma synoptique de l'architecture globale du réseau CNN……...……....…77
Figure III.6 structures des entrées de la première couche de convolution…….....….....78
Figure III.7 schéma du circuit de calcul de la première convolution…….……..…...79
Figure III.8 Architecture détaillée de l'une des huit branches du circuit de calcul de la première convolution……..…….……..…..80
Figure III 9 Schéma de conception d'une branche du circuit de calcul de la première convolution…….....80
Figure III.10 Structure de l'une des huit branches du circuit de calcul du max pooling…….81
Figure III.11 structures des entrées de la deuxième couche de convolution…….....…82
Figure III.12 circuit de conception du circuit de calcul de la convolution 2……..83
Figure III.13 structures des entrées de la troisième couche de convolution……83
Figure III.14 Schéma de conception du circuit de calcul de la convolution 3…….....84
Figure III.15 Schéma de conception du circuit de la couche FC et la couche de classification…….….85
Figure III.16 :capture de la fenêtre Vivado après l'étape de synthèse de l'un des circuits de calcul……..…...…86
Figure III.17: Fenêtre Vivado à la fin de l'étape d'implémentation……..……...…86

Liste des abréviations

AI : Artificial Intelligence

ANN: Artificial Neural Networks

APIs: Automates Programmables Industrials

APSoCs: All Programmable Systems on Chips

ASIC: Application Specific Integrated Circuit

BNN: Binarized Neural Network

BRAM: Block Random Access Memory

CLB: Configurable Logic Block

CMOS: Complementary metal oxide Semiconductor

CNN: Convolutional Neural Network

CONV: Convolutional Layer

CPU: Central Processing Unit

DCM: Digital Clock Manager

DL: Deep Learning

DNNs: Deep Neural Networks

DSP: Digital Signal Processing

EPROM: Erasable programmable read only memory

FC: Fully-connect Layer

FF: Flip flop

FIFO: First in first out

FPGA: Field Programmable Gate Array

GPU: Graphics Processing Unit

HDL: Hardware Description Language

IoT: Internet of Things

IOB: Input Output Block

JPEG: Joint Photographic Experts Group

JTAG: Joint Test Action Group

LUT: Look-up Table

MAC: Multiply and accumulate

ML: Machine Learning

MNIST: Modified National Institute of Standards and Technology

MLP: Multilayer Perceptron Network

MUX: Multiplexer

NN: Neural Network

PS: Processing System

PL: Programmable Logic

PNG: Portable Networks Graphics

RAM: Random Access Memory

ReLU: Rectified Linear Unit

ROM: Read Only Memory

RTL: Register Transfer Level

SoCs: Systems on Chips

SRAM: Static Random-Access Memory

TTL: Transistor-transistor logic

VGG: Visual Geometry Group

VHDL: Very high-speed Integrated Circuits Hardware Description Language

XSG: Xilinx System Generator

Introduction générale

Introduction générale

La classification d'images est essentielle et joue un rôle crucial dans la vie contemporaine. Ce domaine clé de la vision par ordinateur propose des solutions novatrices pour de nombreuses applications, telles que la sécurité et la défense, la santé, l'automatisation industrielle, la recherche d'information, ainsi que le marketing et la publicité. Dans le domaine de la sécurité, la classification des images permet d'identifier les objets suspects et de détecter les menaces potentielles. En médecine, elle est employée pour analyser des images telles que images de rayons X et images IRM afin de repérer des anomalies ou des signes de maladies. Dans le secteur de l'automatisation industrielle, la classification des images sert à inspecter la qualité des produits, à détecter les défauts et à assurer leur conformité aux normes de qualité.

L'intelligence artificielle (IA) représente une avancée majeure dans le domaine de la science informatique, offrant des capacités de traitement et de raisonnement similaires à celles de l'être humain. Au cœur de l'IA se trouve l'une des techniques innovantes du Deep Learning (DL) appelée réseaux de neurones convolutionnels (Convolution Neural Networks : CNN). L'utilisation de cette technique dans la classification et la détection a permis une avancée considérable en termes de précision.

L'architecture des réseaux CNN est basée sur des opérations de convolution, d'activation et de sou échantillonnage réalisées et organisées sous forme de plusieurs couches. Cette architecture requiert une complexité arithmétique considérable pour effectuer les diverses opérations de calcul notamment le calcul des convolution et elle demande un espace mémoire conséquent pour stocker les paramètres du réseau et les résultats des calculs des différentes couches. Par conséquent, l'implémentation des réseaux CNN sur des circuits numériques représente un véritable défi pour les concepteurs de systèmes.

Les plateformes ZYNQ offrent une solution efficace pour l'implémentation en temps réel des réseaux CNN. Elles combinent des ressources logiques reconfigurables (Programmable Logic : PL) et un système programmable de traitement (Processing System : PS) basé sur un processeur ARM. Cette combinaison innovante permet de tirer parti des capacités de calcul parallèle du PL tout en bénéficiant de la flexibilité du processeur ARM. Plusieurs cartes de développement et frameworks ont été mis sur le marché pour faciliter les applications basées sur les circuits ZYNQ, telles que les populaires cartes PYNQ-Z1 et PYNQ-Z2.

Notre travail s'inscrit dans le cadre de l'implémentation d'un réseau de classification de digits manuscrits sur la carte PYNQ-Z1. Afin de mieux présenter le contenu de notre travail, nous avons organisé le reste de ce mémoire comme suivant :

Dans le premier chapitre, nous présentons une introduction au traitement d'image numérique et à la classification d'images, domaine essentiel de la vision par ordinateur. Nous allons mettre l'accent sur l'apprentissage profond, notamment les réseaux de neurones convolutifs (CNN), qui ont révolutionné la classification d'images.

Le deuxième chapitre est consacré à la présentation des circuits FPGA des circuits ZYNQ et de la carte PYNQ-Z1. Dans ce chapitre nous allons mettre la lumière sur l'architecture de circuits FPGA , et plus particulièrement ceux de Xilinx. Nous présentons aussi une étude détaillé sur l'architecture des circuits de type ZYNQ en expliquant les ressources des parties PS et PL ainsi que le rôle de l'interface PS-PL.

Dans le troisième chapitre, nous présentons les étapes de programmation du réseau CNN en utilisant le Matlab et le langage Python, les différents schémas de conception des couches du réseau en utilisant l'outil System Generator de Xilinx et les résultats d'implémentation des différentes couches en utilisant le logiciel Vivado.

Finalement, une conclusion générale résumant le fruit de ce projet est fournie à la fin de ce mémoire

CHAPITRE 1:

Classification des images par CNN

I. Introduction

En théorie de l'information, l'image est reconnue comme l'une des sources les plus riches en informations. De ce fait, le traitement d'image est un domaine crucial dans de nombreuses disciplines de la vie moderne, notamment la vision par ordinateur, l'imagerie médicale, la surveillance et la sécurité. La classification d'images constitue une tâche fondamentale du domaine de la vision par ordinateur visant à comprendre une image dans son ensemble et à lui attribuer une étiquette spécifique.

L'apprentissage profond, en particulier avec l'utilisation fréquente des réseaux de neurones convolutifs (CNN), a révolutionné la classification d'images.

Dans ce chapitre nous allons commencer par la présentation de quelques notions de base du domaine de traitement d'image numérique. Ensuite nous allons mettre l'accent sur les réseaux de neurones de type CNN.

I.2 Notions élémentaires sur les images

I.2.1 Définition d'une image

Une image est une représentation visuelle, d'objets, d'être vivants ou de concepts [1].

En traitement d'image, la majorité du temps, on considère qu'il s'agit d'une fonction mathématique de R^2 dans R où le couplet d'entrée est considéré comme une position spatiale alors que la sortie représente l'intensité (couleur ou niveaux de gris) du phénomène physique. Il arrive cependant que l'image soit dite "3D" donc la fonction est de R^3 dans R.

I.2.2 Représentation des images

Une image est une fonction d'au moins deux variables réelles, le tableau I.1 donne les différentes dimensions de représentation des images.

Tableau I.1 différentes dimensions des images.

Image : f (x, y)	Image 2D
Volume : f (x, y, z)	Image 3D
Séquence d'image : f (x, y, t)	Image 3D
Séquence de volumes : f (x, y, z, t)	Image 4D

Les valeurs prises par f (.) peuvent être :

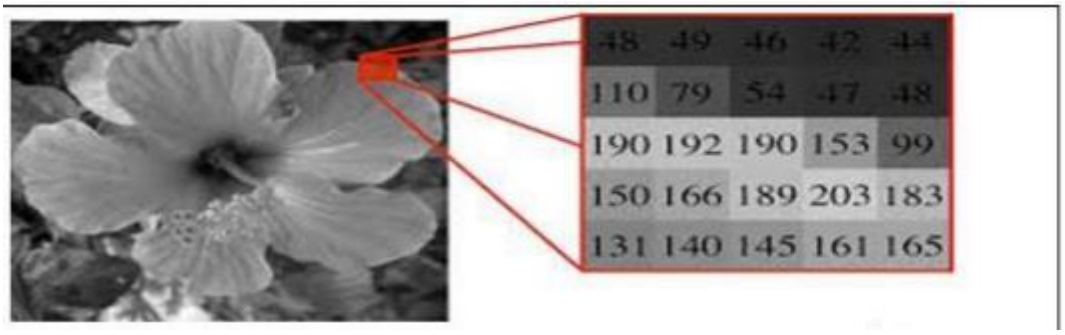
- Scalaires (intensité lumineuse)
- Vectorielles (couleur (RVB), …), imagerie multispectrale, image de paramètres…) [2]

I.2.3 Image numérique(numérisée)

L'image numérique est l'image dont la surface est divisée en éléments de taille fixe appelés cellules ou pixels, ayant chacun comme caractéristique un niveau de gris ou de couleurs. [3]

Après sa numérisation, une image est représentée par une matrice bidimensionnelle de valeurs numériques F (x, y) comme le montre la figure I.1 :

- x,y: coordonnées cartésiennes d'un point de l'image.
- F (x, y) : niveau d'intensité. La valeur en chaque point exprime la mesure d'intensité lumineuse perçue par le capteur.



La figure I.1 : Matrice de représentation d'une partie d'une image numérique.

L'image peut être de source naturelle ou artificielle (figure I .2). Une image naturelle est le résultat d'une acquisition d'une scène réelle (caméra, microscope, tomographie, infra-rouge, satellite) alors qu'une image artificielle est une image générée ou produite numériquement (synthèse d'images, réalité virtuelle, jeux…) [3]



Figure I.2 *:* Exemples d'images numérique naturelle et artificielle.

I.2.4 Types d'images numériques

En réalité il existe, (03) trois principaux types d'images qui sont illustrés dans la figure I.3 :

- Les images binaires : ne contiennent que deux valeurs de pixels (noir et blanc), donc chaque pixel est représenté par un seul bit. Ce type nécessite un minimum d'espace de stockage.
- Les images en niveaux de gris : elles utilisent un ensemble de niveaux de gris pour représenter la luminosité des pixels. Le nombre de bits nécessaire à la représentation d'un pixel est donnée par :

$$\boldsymbol{n = log_2(N)} \qquad \textbf{I.1}$$

Où N : le nombre de niveau et n est le nombre de bits.

Pour une image en niveaux de gris à 8 bits chaque pixel est représenté par l'un des 256 niveaux allant de 0 à 255.

- Image couleur dans l'espace RVB :En peut représenter une image couleur numérique par addition de (03) trois couches des couleurs représentatifs du spectre qui sont : Rouge, Vert, Bleu.

Pour une image en couleur RVB à 24 bits chaque pixel est représenté par trois valeurs allant de 0 à 255.

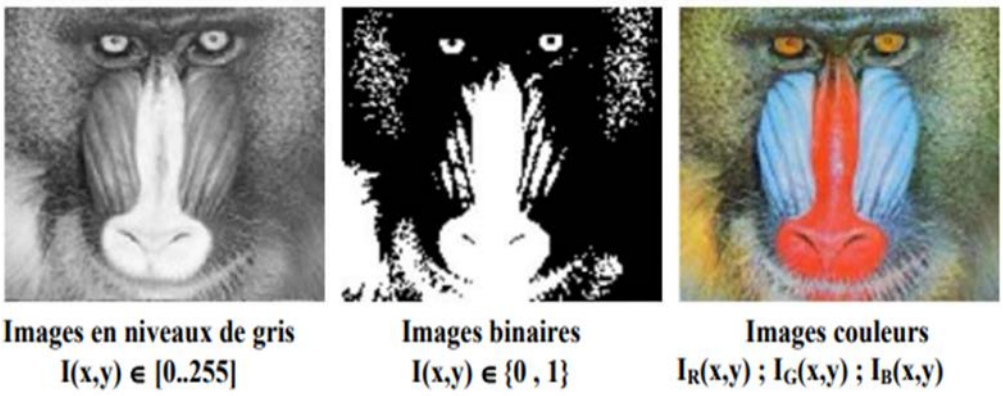


Figure I.3 : Exemples de types d'images.

I.2.5 Caractéristiques d'une image numérique

Comme nous l'avons vu, l'image est un ensemble structuré d'informations parmi ses caractéristiques nous pouvons citer les paramètres suivants :

I.2.5.1 La résolution

La résolution d'une image correspond au niveau de détail qui va être représenté sur cette image. C’est le nombre de pixels par unité de longueur dans l'image à numériser. Elle est en dpi (dots per inch) ou en ppp (points par pouce).

Plus le nombre de pixels est élevé par unité de longueur de l'image à numériser, plus la quantité d'inflation qui décrit l'image est importante et plus la résolution est élevée. [4]

I.2.5.2 Dimension

C’est la taille de l’image qui est le résultat de multiplication du nombre de lignes par le nombre de colonnes de la matrice de représentation de l’image. Par exemple : une image de 640x480 correspond à une image présentant 640 pixels en largeur et 480 en hauteur.

I.2.5.3 Bruit

Un bruit (parasite) dans une image est considéré comme un phénomène de variations aléatoires de la couleur ou de la luminosité des pixels, qui ne sont pas liées au contenu de l'image mais résultent plutôt de facteurs tels que les imperfections des capteurs d'image, les conditions d'éclairage, les interférences électromagnétiques.[5]

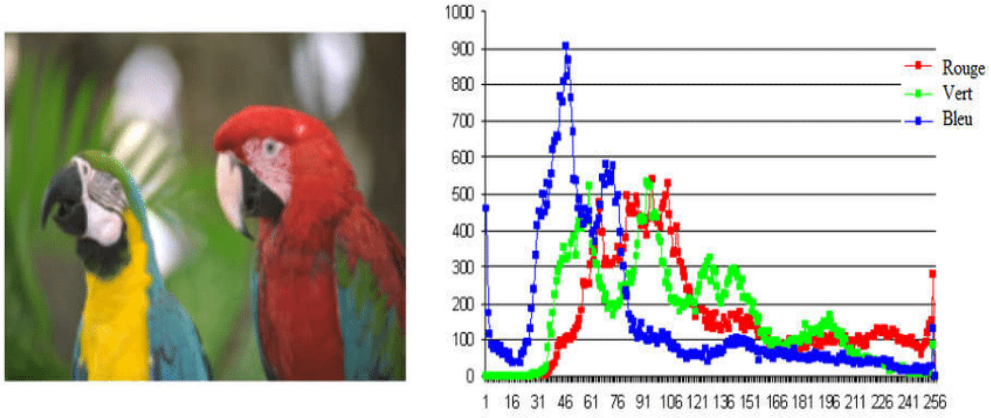


a **b**

Figure I.4 : Exemple de bruit dans les images (a) : image sans bruit, (b) : image avec bruit

I.2.5.4 Histogramme

L’histogramme des niveaux de gris ou des couleurs d’une image est une fonction qui donne la fréquence d’apparition de chaque niveau de gris (couleur) dans l’image. Il permet de donner un grand nombre d’information sur la distribution des niveaux de gris (couleur) et de voir entre quelles bornes est repartie la majorité des niveaux de gris (couleur) dans le cas d’une image trop claire ou d’une image trop foncée.[6]



La figure I.5 : Histogramme des couleurs RVB de l'image parrots.

I.2.5.5 Luminance

C'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'aire apparente de cette surface, pour un observateur lointain, le mot luminance est substitué au mot brillance, qui correspond à l'éclat d'un objet. Une bonne luminance se caractérise par [6] :

- 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 parasites.[6]

I.2.5.6 Contraste et textures

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. Si L1 et L2 sont les degrés de luminosité respectivement de deux zones voisines A1 et A2 d'une image,

Le contraste C est défini par le rapport :

$$C=\frac{L1-L2}{L1+L2} \qquad \text{I.2}$$

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.[6]

I.2.6 Traitement numérique des images

Les différentes techniques de traitement d'images sont destinées à l'exploitation des informations contenues dans les images pour améliorer la qualité des images et les rendre plus facilement interprétables, d'autre termes elles permettent d'augmenter la qualité visuelle de l'image. [5]

Comme le montre la figure I.6, un système de traitement d'image est généralement composé des unités suivantes :

- Un système d'acquisition et de numérisation qui permet d'effectuer l'échantillonnage et la quantification d'une image.
- Une mémoire de masse pour stocker les images numérisées.
- Un système de visualisation.
- Une unité centrale permettant d'effectuer les différentes opérations de traitement d'images. [7]

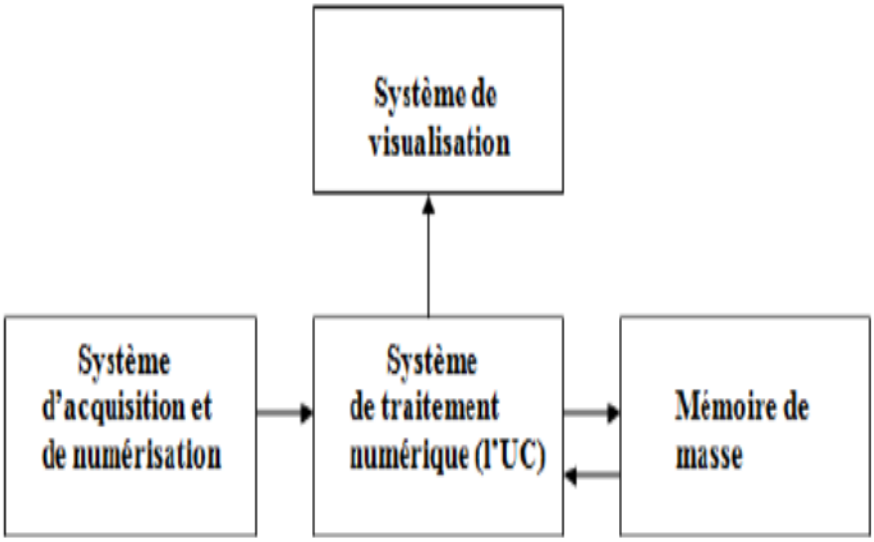


Figure I.6 : Composition d'un système de traitement numérique d'images

Les applications du traitement numérique des images sont vastes et variées et couvrent de nombreux domaines. Dans le paragraphe suivant, nous présentons des exemples courants de ce domaine.

I.2.6.1 Filtrage Numérique

Les images numériques telles qu'elles sont acquises, sont très souvent inexploitables pour le traitement d'images. Elles contiennent des signaux bruités. Pour remédier à cela, différents prétraitements pour l'amélioration ou la correction sont effectués. Le filtrage est l'une des

techniques permettant l'amélioration de la qualité de l'image. On peut scinder les filtres en deux grandes catégories [8] :

- **Filtres linéaires** :

Les filtres linéaires transforment un ensemble de données d'entrée en un ensemble de données de sortie par une convolution bidimensionnelle qui est une opération mathématique. Ils permettent de supprimer le bruit dans l'image. Chaque filtre a une taille N x N avec N impair. Les filtres linéaires les plus connus sont les filtres passe-bas, passe-haut. [9]

Les filtres passe-bas ou de lissage n'affectent pas les composantes de basse fréquence dans les données d'une image, mais doit atténuer les composantes de haute fréquence. L'opération de lissage est souvent utilisée pour atténuer le bruit et les irrégularités de l'image. En pratique, il faut choisir un compromis entre l'atténuation du bruit et la conservation des détails et contours significatifs. [10]

Les filtres Passe-haut ou d'accentuation sont utilisés pour le renfoncement des contours et leur extraction. Ces filtres n'affectent pas les composantes de haute fréquence d'un signal, mais doit atténuer les composantes de basse fréquence.[10]

- **Filtres non linéaires :**

Ils sont conçus pour régler les problèmes des filtres linéaires, leur principe est le même que celui des filtres linéaires, il s'agit toujours de remplacer la valeur de chaque pixel par la valeur d'une fonction calculée dans son voisinage. La différence majeure, est que cette fonction n'est plus linéaire mais une fonction quelconque (elle peut inclure des opérateurs de comparaisons ou de classification). Un exemple typique de filtres non linéaires est le filtre médian. Ce filtre est très utilisé pour éliminer le bruit sur une image qui peut être de différentes origines (poussières, petits nuages, baisse momentanée de l'intensité électrique sur les capteurs, ...). L'avantage de ce filtre réside dans le fait qu'il conserve les contours alors que les autres types de filtres ont tendance à les adoucir. [10]

L'algorithme de filtre médian est le suivant :

1. Trier les valeurs par ordre croissant.
2. Remplacer la valeur du pixel centrale par la valeur située au milieu de la triée.
3. Répéter cette opération pour tous les pixels de l'image.

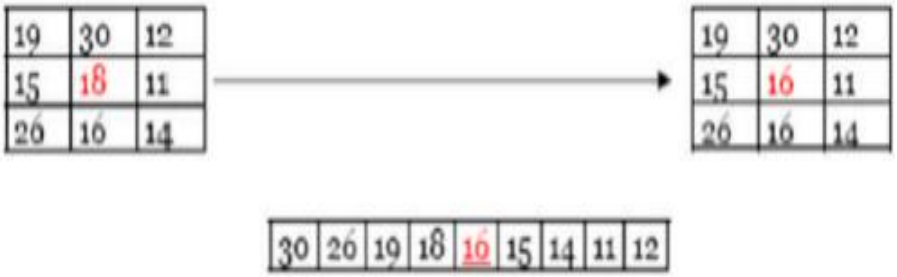


Figure I.7 : Principe du filtre médian.

a. Filtre maximum :

On applique le même traitement que celui du filtre médian mais la valeur du pixel du centre comme la montre la figure I.8, va être changée par le maximum.

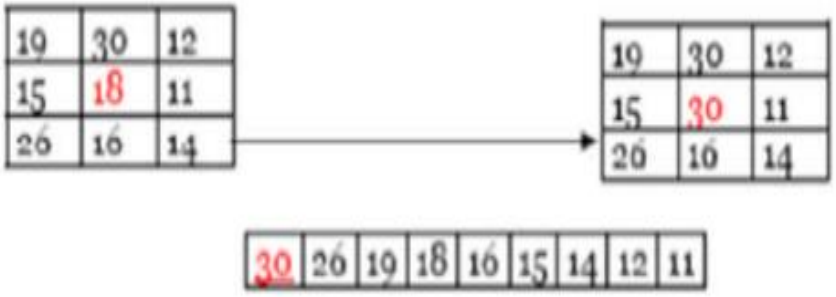


Figure I.8 : Principe du filtre maximum

b. Filtre minimum :

On applique le même traitement que celui du filtre maximum mais, cette fois, la valeur du pixel du centre comme la montre la figure va être remplacée par le minimum. [10]

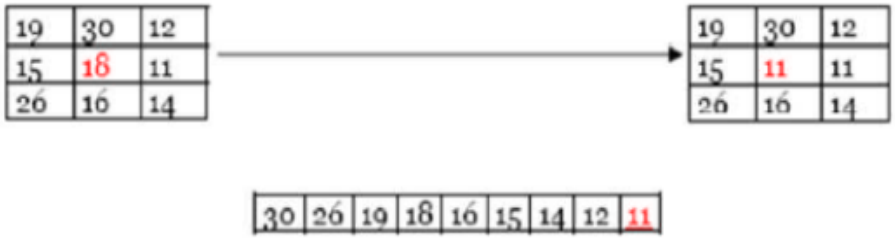


Figure I.9 : Principe du filtre minimum

I.2.6.2 Domaines d’application

- Contrôle de présence / absence : Sur des chaînes de production, on vérifie en bout de chaîne avec une caméra vidéo la présence d'une pièce dans un ensemble plus complexe. Pour cela bien souvent il suffit de faire un simple seuillage dans une région spécifique.
- Contrôle du niveau de maturation des fruits sur une chaîne de conditionnement. Il s'agit de reconnaître à la couleur et à la texture du fruit son degré de maturité et donc la catégorie sous laquelle il sera emballé puis vendu.
- Construction et correction de cartes géographiques d'après des images satellites ou des images aériennes. On recale d'après des informations topographiques les images reçues, puis on les met sur la carte en correspondance avec les informations trouvées dans l'image : voies de communication, voies et plans d'eau, parcelles agricoles...
- Surveillance et évaluation de la production agricole. Il est possible de déterminer le degré de maturation des cultures, la quantité d'eau nécessaire pour l'irrigation, le rendement moyen... On peut ainsi établir des prévisions à large échelle de la récolte à venir.
- Reconnaissance de l'écriture. La reconnaissance de l'écriture manuscrite progresse de jour en jour. Elle est suffisamment opérationnelle pour que la majorité des adresses, même manuscrites, soient reconnues automatiquement sur le courrier postal.
- Recherche d'image par le contenu. L'objectif de cette technique est de rechercher, parmi une base de données d'images, les images similaires à une image exemple, ou ayant certaines caractéristiques, par exemple rechercher toutes les images comportant un vélo.
- Analyse de la vidéo. L'objectif de cette technique devenue une discipline depuis les années 2000 (lorsque la puissance des processeurs peu onéreux et en particulier des PC a permis des traitements puissants en temps réel) est d'interpréter les faits observés à l'image afin de signaler ou d'enregistrer des faits marquants. Le plus souvent, la caméra est fixe et observe les mouvements d'une scène. Les applications sont nombreuses : Protection des biens (détection d'intrusion, détection d'objet abandonné ou déposé...), Identification (biométrie faciale), Sécurité des personnes (détection de chutes de personnes, franchissement de rambardes, ...), Animations (planchers animés selon les mouvements des danseurs en boîte de nuit), Détection de feux (industriel, forêts, tunnels, ...), Surveillance de tunnels (comptage, mesure de vitesse, détection de fuites/anomalies dans les plafonds), surveillance de tuyaux et autres process industriels.[6]



Figure I.10 : Exemples de domaines d'applications du traitement d'images.

II.La Classification des images

La classification automatique des images consiste à attribuer automatiquement une classe à une image à l'aide d'un système de classification. On retrouve ainsi la classification d'objets, de scènes, de textures, la reconnaissance de visages, d'empreintes digitale et de caractères. Il existe deux principaux types d'apprentissage : l'apprentissage supervisé et l'apprentissage non-supervisé. Dans l'approche supervisée, chaque image est associée à une étiquette qui décrit sa classe d'appartenance. Dans l'approche non supervisée les données disponibles ne possèdent pas d'étiquettes.[11]

II.1 Les étapes d'une classification :

Les étapes de classifications peuvent varier d'une méthode à l'autre mais généralement parlant les étapes sont les suivantes :

1. **Collecte des données :** rassembler un ensemble de données d'images représentatives pour l'entraînement du modèle de classification. Ces données peuvent

provenir de différentes sources telles que des bases de données publiques, des captures d'appareil photo, des collections d'images en ligne, etc.

2. **Prétraitement des données :** pour mettre les images dans un format approprié pour l'entraînement du modèle. Cela peut inclure le redimensionnement des images, la normalisation des valeurs de pixel, la conversion en niveaux de gris ou en couleurs RVB, etc.

3. **Extraction des caractéristiques :** cette étape consiste à extraire des caractéristiques pertinentes des images qui seront utilisées par le modèle de classification pour distinguer les différentes classes. Selon l'approche utilisée, ces caractéristiques peuvent être extraites manuellement (par exemple, descripteurs de texture, descripteurs de forme) ou automatiquement à l'aide de techniques d'apprentissage automatique.

4. **Sélection du modèle de classification :** il s'agit de choisir le modèle à utiliser pour réaliser l'opération de classification. Cela peut inclure des modèles classiques tels que SVM, k-NN, des réseaux neuronaux peu profonds ou des architectures de réseaux neuronaux profonds pré-entraînées comme des réseaux de neurones convolutionnels (CNN).

5. **Entraînement du modèle :** Utilisez pour entraîner le modèle de classification. Pendant l'entraînement, le modèle utilise l'ensemble de données d'entraînement pour ajuster les paramètres du modèle de classification. Généralement, l'entrainement est basé sur la minimisation d'une fonction de perte en comparant les prédictions du modèle aux étiquettes de classe réelles des images.

6. **Evaluation et validation :** Évaluez les performances du modèle sur un ensemble de validation pour estimer sa capacité à généraliser à de nouvelles données. L'évaluation est effectuée sur un ou plusieurs paramètres d'évaluation. Une fois que le modèle a été entraîné et optimisé, on fait une deuxième évaluation de ses performances sur un ensemble de test indépendant pour estimer sa performance réelle sur de nouvelles données non vues. Après évaluation du modèle sur l'ensemble de test et sa validation, il peut être déployé dans un environnement de production pour classer de nouvelles images en utilisant les connaissances acquises lors de l'entraînement.

II.2. Quelques méthodes classiques de classification

Les méthodes classiques de classification des images font référence aux différentes techniques établies qui ont été largement utilisées avant l'avènement des réseaux neuronaux profonds et de l'apprentissage profond. Ces méthodes peuvent être catégorisées comme

supervisées ou non supervisés.

Pour **les méthodes supervisées** on dispose au départ d'un ensemble d'échantillons dit d'apprentissage dont le classement est connu. Ces échantillons sont utilisés pour l'apprentissage des règles de classement. Il est nécessaire d'étudier la fiabilité de ces règles pour les comparer et les appliquer, évaluer les cas de sous apprentissage ou de sur apprentissage (complexité du modèle). On utilise souvent un deuxième échantillon indépendant, dit de validation ou de test.

Les méthodes non supervisées procèdent de la façon contraire. C'est-à-dire ne nécessitent aucun apprentissage et aucune tâche préalable d'étiquetage manuel. Elles consistent à représenter un nuage des points d'un espace quelconque en un ensemble de groupes appelé Cluster. Un « Cluster » est une collection d'objets qui sont « similaires » entre eux et qui sont « dissemblables » par rapport aux objets appartenant à d'autres groupes. [12]

II.2.1 La classification SVM (Support Vector Machine) :

C'est une technique d'apprentissage supervisé basé sur la séparation des différentes classes en cherchant un hyperplan dans un espace de caractéristiques qui sépare de manière optimale les données de différentes classes. L'hyperplan choisi est celui qui maximise la marge, c'est-à-dire la distance entre l'hyperplan et les points de données les plus proches de chaque classe, appelés vecteurs de support. Cela permet de maximiser la robustesse du modèle aux nouvelles données. La méthode SVM utilise une fonction de décision pour attribuer une classe à de nouvelles données. Durant la phase d'apprentissage, l'algorithme SVM ajuste les paramètres de l'hyperplan en fonction des données d'apprentissage pour minimiser une fonction de coût, qui mesure à quel point les prédictions du modèle correspondent aux étiquettes de classe réelles. La figure I.11 illustre le principe de la classification par la méthode SVM.

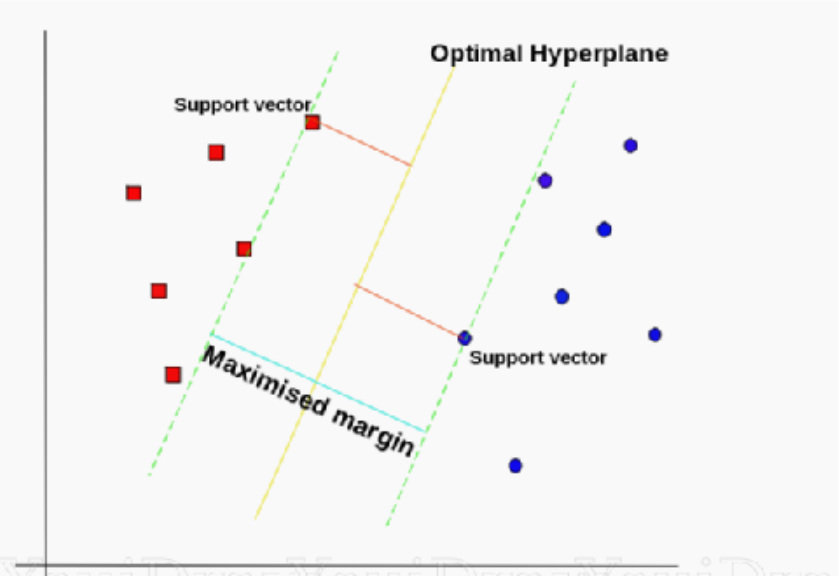


Figure I.11 : principe de classification par la méthode SVM.

II.2.2 Classification par k plus proches voisins (KNN) :

Le but de cette classification est de trouver les k éléments les plus proches de l'élément à classer et qui sont déjà classés comme le montre la figure I.12. Le voisinage est déterminé en calculant la distance entre deux éléments en suite l'élément à classer est affecté à la classe la plus représentée parmi ces k voisins. Cette méthode est simple à mettre en œuvre et robuste au bruit, mais nécessite de stocker un grand nombre de données et peut être lent pour de grandes bases de données.

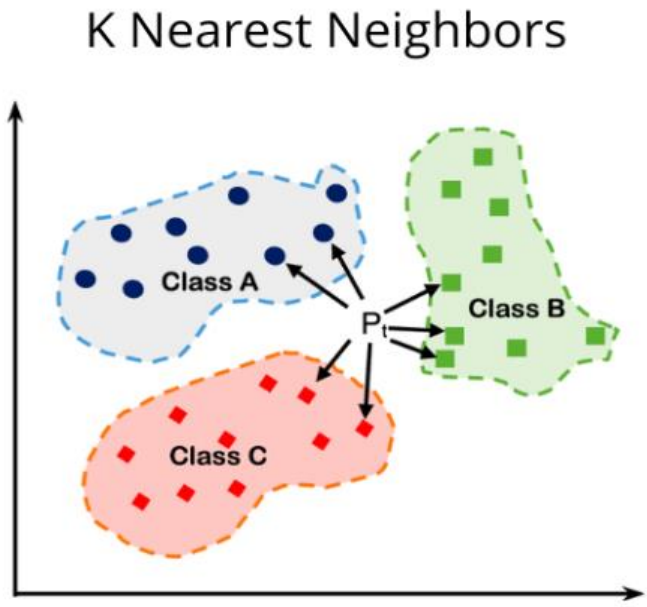


Figure 1.12 : classification par la méthode KNN.

II.3 Domaines d'application de la classification :

La classification est une méthode polyvalente qui trouve des applications dans de nombreux domaines, facilitant l'organisation, la compréhension et la prise de décision basée sur des similitudes et des caractéristiques communes. Parmi les domaines d'application de cette technique on peut citer :

1. Informatique et traitement de l'information : La classification est largement utilisée dans le domaine de l'informatique pour organiser des fichiers, des données et des informations. Elle est utilisée dans la recherche d'informations sur le web, la classification automatique de documents, etc.
2. Biologie et taxonomie : En biologie, la classification est utilisée pour regrouper les organismes en catégories basées sur leurs caractéristiques communes. La taxonomie, par exemple, est un système de classification utilisé pour classer les organismes vivants.
3. Bibliothéconomie et documentation : La classification est utilisée dans les bibliothèques pour organiser les livres et autres ressources en catégories spécifiques, facilitant ainsi la recherche des utilisateurs.
4. Marketing : Dans le domaine du marketing, la classification des clients, des produits ou des marchés peut être utilisée pour mieux cibler les stratégies commerciales.
5. Médecine : En médecine, la classification des maladies et des symptômes est essentielle pour diagnostiquer et traiter les patients.
6. Écologie : La classification est utilisée pour regrouper les espèces d'organismes en écosystèmes, ce qui aide à comprendre les relations écologiques et à prendre des mesures de conservation.
7. Finance : La classification peut être utilisée dans le domaine financier pour catégoriser.

III. Classification par réseau CNN :

III.1. De l'intelligence artificielle au réseaux CNN :

III.1.1 intelligence artificielle :

L'intelligence artificielle (IA) correspond à un ensemble de technologies « mises en œuvre en vue de réaliser des machines capables de simuler l'intelligence humaine », comme le raisonnement ou l'apprentissage. L'IA s'intéresse à tous les cas où le traitement des informations ne peut être ramené à une méthode simple, précise, algorithmique.

Depuis l'émergence des techniques associées à l'intelligence artificielle dans notre vie quotidienne, un nouveau lexique inonde les articles scientifiques, et il est parfois difficile de comprendre de quoi il s'agit. Lorsqu'on parle d'intelligence artificielle, on fait très souvent l'allusion aux technologies associées comme le Machine Learning ou le Deep Learning. Deux termes extrêmement utilisés avec des applications toujours plus nombreuses, mais pas toujours bien définis.

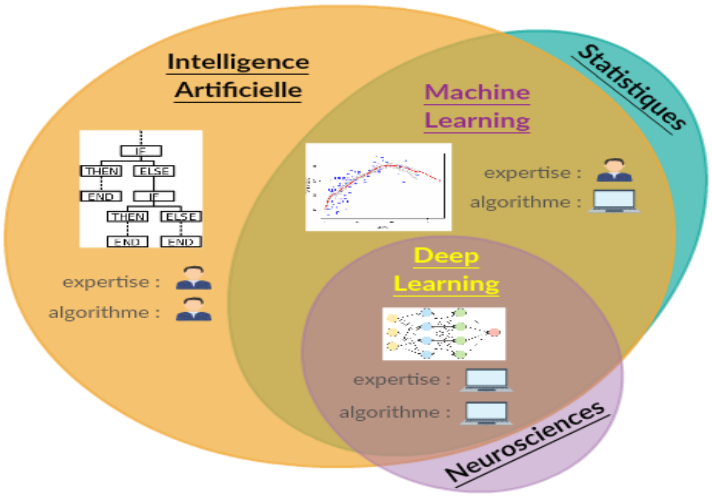


Figure I.13: les types de L'intelligence Artificielle

III.1.2. Le Machine Learning ML:

Le Machine Learning ou Apprentissage automatique est une branche de l'intelligence artificielle qui concerne la conception, l'analyse, le développement et l'implémentation de modèles, permettant à une machine d'apprendre à partir des données par un processus systématique afin de remplir une tâche. Le Machine Learning empreinte donc les techniques de Datamining permettant de classer et trier les objets de notre monde (Classification), de prédire des événements (Prédiction), d'identifier des règles sous-jacentes à des données, d'analyser des séries temporelles, d'interpréter des textes (Analyse des sentiments sur les réseaux sociaux, NLP) … [13]. On distingue principalement trois types d'apprentissages :

a) **L'apprentissage supervisé** : C'est l'application la plus directe, on cherche un modèle qui prédit (Régression ou classification) notre variable expliquée (output, responsa) préalablement connue à partir des variables explicatives (Feutres, inputs).

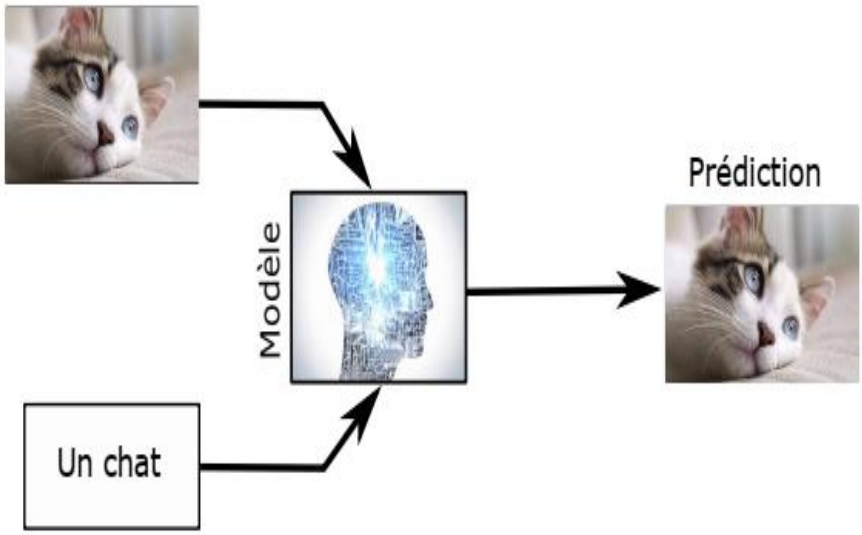


FIGURE I.14 : Apprentissage supervisé.

b) **L'apprentissage non supervisé :** Quand on ne dispose pas de variable à prédire, et que le nombre de classes et leur nature n'ont pas été prédéterminés, on parle d'apprentissage non supervisé ou **Clustering**. L'algorithme doit découvrir par lui-même la structure plus ou moins *cachée* des données et séparer les données en différents groupes (clusters) homogènes (notion de similarité).

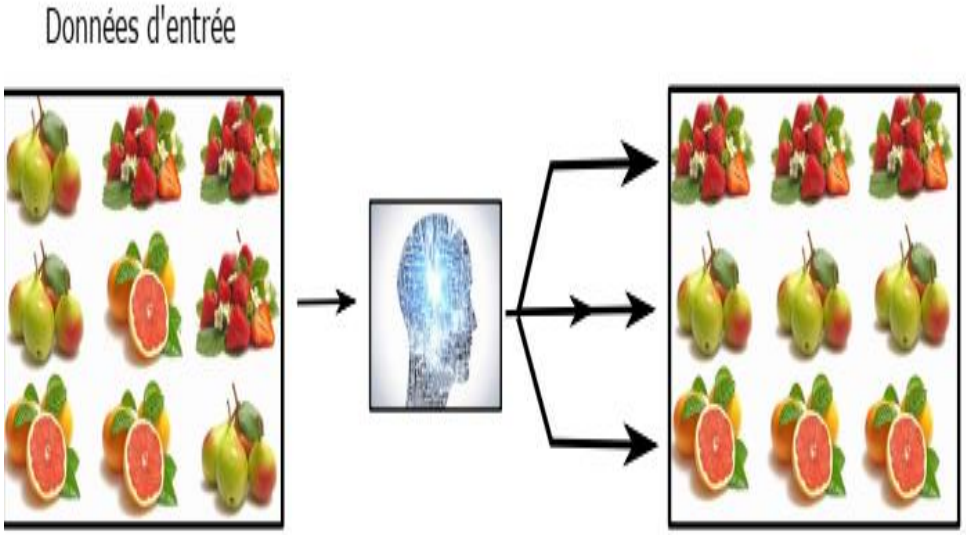


FIGURE I.15 : Apprentissage non supervisé

c) **L'apprentissage par renforcement** : Le but du "Renforcement Learning" est d'apprendre en permanence, à partir d'expériences, ce qu'il convient de faire en différentes

situations, de façon à optimiser une récompense quantitative au cours du temps. L'action de l'algorithme sur l'environnement produit une valeur de retour qui guide l'algorithme d'apprentissage (Essaie, Erreur).

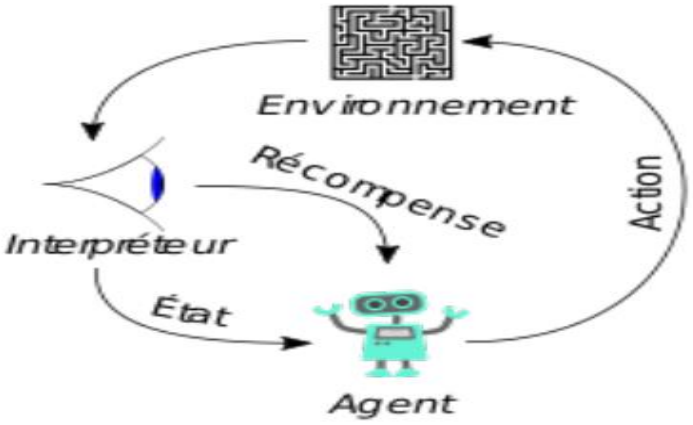


FIGURE I.16 : Apprentissage par renforcement

III.1.3 Deep Learning :

Le Deep Learning ou apprentissage profond, est une technique d'intelligence artificielle qui utilise des réseaux de neurones artificiels pour résoudre des problèmes complexes. Ces réseaux sont composés d'un nombre important de couches interconnectées qui permettent d'extraire des informations et des caractéristiques à partir de grandes quantités de données. Le Deep Learning est utilisé pour des tâches comme la reconnaissance d'images (traitement d'image), la traduction automatique, la synthèse vocale, et bien d'autres. [13]

La première couche du réseau DL correspond aux neurones d'entrée et la dernière transmet les résultats de sortie. Entre les 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.

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.[13]

Plusieurs types d'algorithmes sont **utilisés en apprentissage profond**. Chaque algorithme possède ses propres spécificités et applications. La figure I.17 donne une vue globale sur les algorithmes les plus connues du Deep Learning. [13]

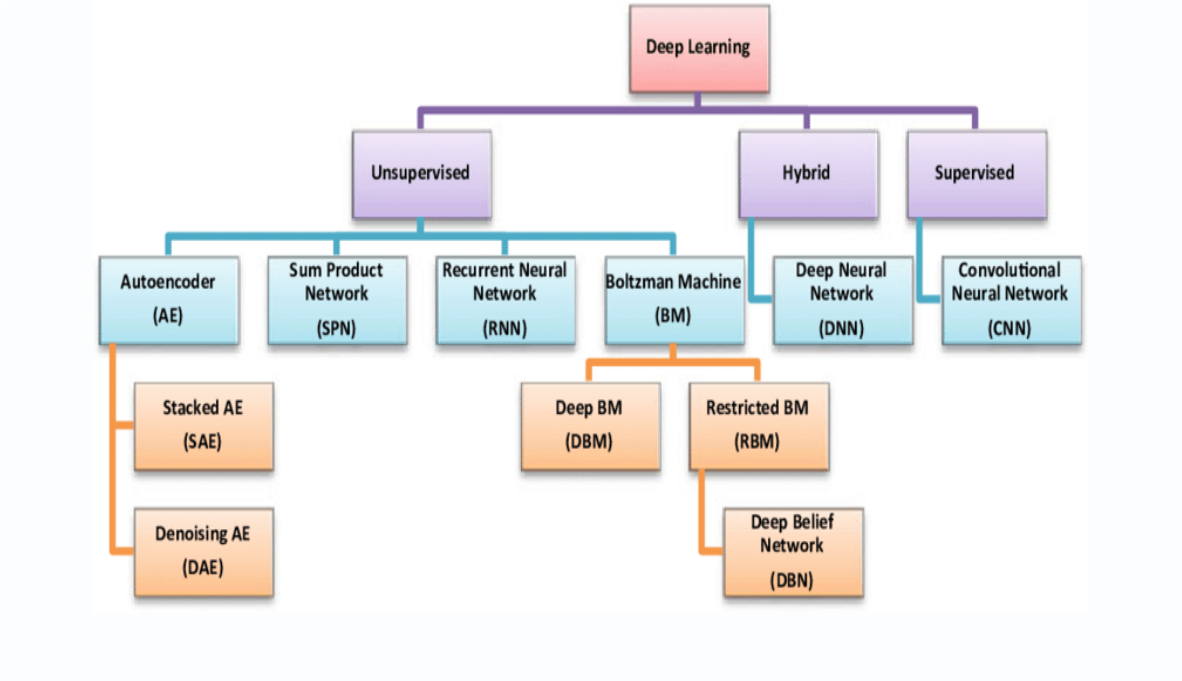


FIGURE I.17 : algorithmes du Deep Learning

De nos jours, l'apprentissage profond sert à développer de **nombreuses technologies révolutionnaires**. Celles-ci sont très utiles dans la vie de tous les jours [13]. On peut citer à titre d'exemples :

- **La conduite autonome :** Le Deep Learning est l'intelligence artificielle qui a permis de mettre en place la conduite autonome pour les véhicules. Certains algorithmes identifient en effet les panneaux de signalisation tandis que d'autres localisent les piétons. Cette technologie améliore fortement la sécurité routière et l'expérience de conduite des automobilistes.

- **Dans le domaine de la médecine :** L'IA de Deep Learning permet de distinguer les tumeurs cancéreuses de celles qui ne le sont pas. Elle scanne les photos de radiographie avec une plus grande précision que l'œil humain et permet donc d'anticiper la prise en charge pour allonger les chances de guérison du malade.

- **Dans le secteur de l'agriculture :** L'agriculture biologique s'appuie sur des drones intelligents capables d'identifier les mauvaises herbes en scannant au survol plusieurs hectares de plantation. Cela permet aux agriculteurs de concentrer uniquement leur énergie sur les zones qui nécessitent un désherbage.

III .2. Réseaux de neurones artificiels :

III.2.1 Structures d'un réseau de neurones artificielles :

Un réseau de neurones est constitué de couches de neurones virtuels connectés entre elles. On parle de neurones « fully connected » dans le sens ou chaque neurone d'une couche est connecté avec tous les neurones de la couche précédente et de la couche suivante. Les neurones entre la couche d'entrée (où l'on injecte les variables) et la couche de sortie (résultat à prédire) sont appelés des couches cachées (hidden layers). [12-14]

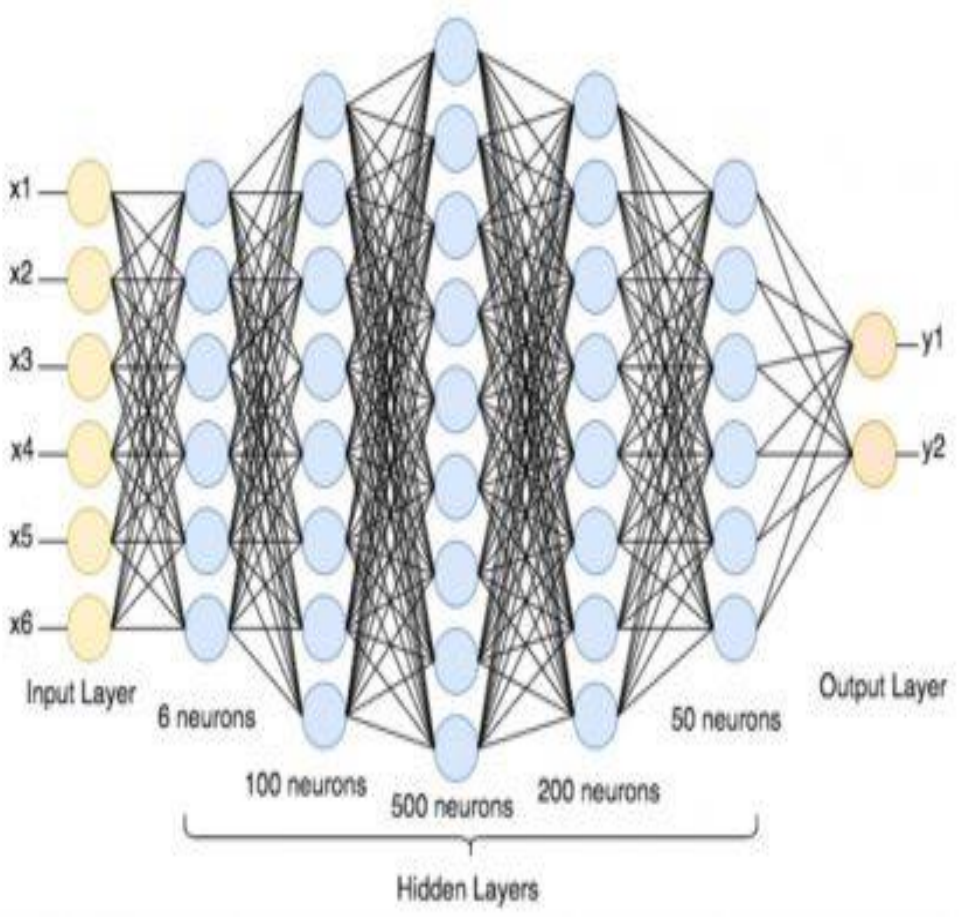


FIGURE I.18 : Schéma d'un réseau de neurones artificiel.

L'information dans le réseau se propage d'une couche à l'autre, on dit qu'ils sont de type "feed-forward ". Nous distinguons trois types de couches :

- **Couche d'entrée (**input) : les neurones de cette couche reçoivent les valeurs d'entrée du réseau et les transmettent aux neurones cachés. Chaque neurone reçoit une valeur, il ne fait pas donc de sommation.
- **Couches cachées :** chaque neurone de cette couche reçoit l'information de plusieurs couches précédentes, effectue la sommation pondérée par les poids, puis la transforme selon sa fonction d'activation qui est en général une fonction sigmoïde. Par la suite, il envoie cette réponse aux neurones de la couche suivante.
- **Couche de sortie** (output) : elle joue le même rôle que les couches cachées, la seule différence entre ces deux types de couches est que la sortie des neurones de la couche de sortie n'est liée à aucun autre neurone.

III.2.2 La fonction de combinaison et la fonction d'activation :

Comme le montre la figure I.19, chaque neurone est caractérisé par une fonction de combinaison et une fonction d'activation.

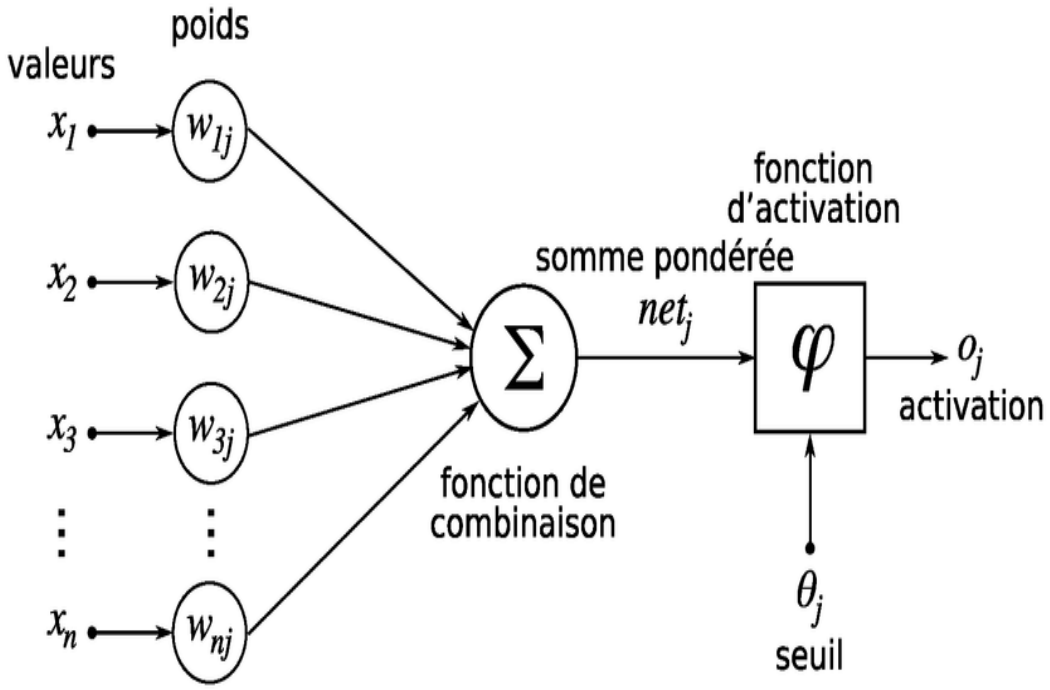


FIGURE I.19 : Structure d'un neurone artificiel

➢ Fonction de combinaison :

La fonction de combinaison réalise une combinaison linaire de l'ensembles des signaux d'entrée x_i en utilisant un vecteur de poids (Wi) selon l'équation suivante :

$$Z=\sum_{i=1}^{n} w_i \cdot x_i + b \qquad \text{I.3}$$

Où : Z est le résultat de la fonction de combinaison, xi les entrées du neurone, wi les poids associés au neurone et b est le biais (un terme constant ajouté pour ajuster la sortie)

- **Fonction de d'activation :**

Le terme de fonction d'activation est synonyme de potentiel d'activation pour les neurones biologiques. La fonction d'activation applique une opération non linaire sur la sortie de la fonction de combinaison ce qui est similaire au fonctionnement des neurones biologiques.

La figure I.20 illustre le principe de la fonction d'activation. En fonction des résultats de la fonction d'activation, le neurone peut être dans 3 états : inactif (fonction d'activation à zéro), partiellement actif (entre 0 et 1) ou complètement actif (fonction d'activation à 1)

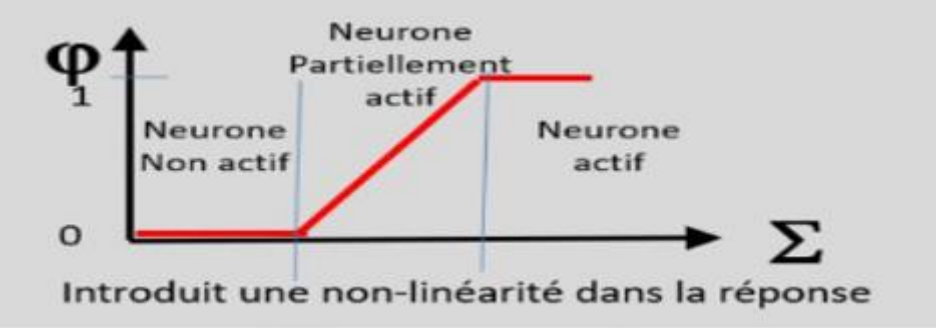


Figure I.20 principe de la fonction d'activation.

Plusieurs fonctions d'activation ont été mises au point, les plus utilisées sont :

1. Fonction Sigmoïde (Marche Douce ou Logistique) : Elle est souvent utilisée comme fonction d'activation pour les réseaux de neurones, dans lesquels les valeurs de sortie désirées sont soit binaires soit dans un intervalle compris entre 0 et 1. Le but de cette fonction est de réduire la valeur d'entrée à un intervalle entre 0 et 1.

Si la valeur en entrée est un très grand nombre positif, la fonction convertira cette valeur en une probabilité de 1. A l'inverse, si la valeur en entrée est un très grand nombre négatif, la fonction convertira cette valeur en une probabilité de 0. L'équation ci-dessous représente la fonction Sigmoïde :

$$\sigma(x) = \frac{1}{1+e^{x}} \qquad \text{I.4}$$

2. Fonction Tanh (Tangente Hyperbolique) : Cette fonction ressemble à la fonction Sigmoïde. La différence avec la fonction Sigmoïde est que la fonction Tanh produit un résultat compris entre -1 et 1. La fonction Tanh est en terme général préférable à la

fonction Sigmoïde car elle est centrée sur zéro. Les grandes entrées négatives tendent vers -1 et les grandes entrées positives tendent vers 1.

3.Fonction ReLU (Unité de Rectification Linéaire) : Elle est inspirée par des motivations biologiques. Elle a été introduite pour la première fois en 2011 par Glorot et al. Elle est favorisée pour les réseaux neuronaux de type CNN, L'équation ci-dessous représente la fonction ReLU [15], [16] :

$$\mathbf{F(x) = max\ (0,\ x)\ I.5}$$

Si l'entrée est négative la sortie est 0. Et si elle est positive alors la sortie est x.

4. Fonction Softmax : Est également appelée fonction exponentielle normalisée. Elle est couramment utilisée pour représenter une distribution de probabilité pour les K sorties possibles.

Ainsi, la fonction softmax a été fortement utilisée dans les tâches de classification. Elle est définie comme :

$$\boldsymbol{F(x)_i = \frac{e_i^x}{\sum_{k=1}^{k} e^x k}} \qquad \mathbf{I.6}$$

III.2.3 La rétropropagation du gradient de l'erreur :

Les poids respectifs de chaque neurone sont déterminés pendant la phase d'apprentissage avec l'algorithme de rétropropagation du gradient de l'erreur (SGD pour Stochastic Gradient Descent). L'erreur est l'écart entre la valeur de sortie attendue et la valeur prédite. Elle est très grande en début d'apprentissage car les poids des neurones ont été générés aléatoirement (distribution gaussienne), puis elle diminue progressivement. Ainsi, les poids des neurones qui contribuent à engendrer une erreur importante se verront modifiés de manière plus importante que les poids qui contribuent à une erreur marginale. Cette phase d'apprentissage est relativement longue et consommatrice de temps de calcul pour les réseaux importants. Le calcul peut nécessiter un accès au cloud. Par contre, une fois le réseau de neurones entrainé et validé, son utilisation ne nécessite pas de performance particulière en puissance de calcul. La figure I.21 illustre le fonctionnement d'un réseau de neurones artificiels (RNA) et l'optimisation des poids à l'aide de la méthode de descente de gradient stochastique.

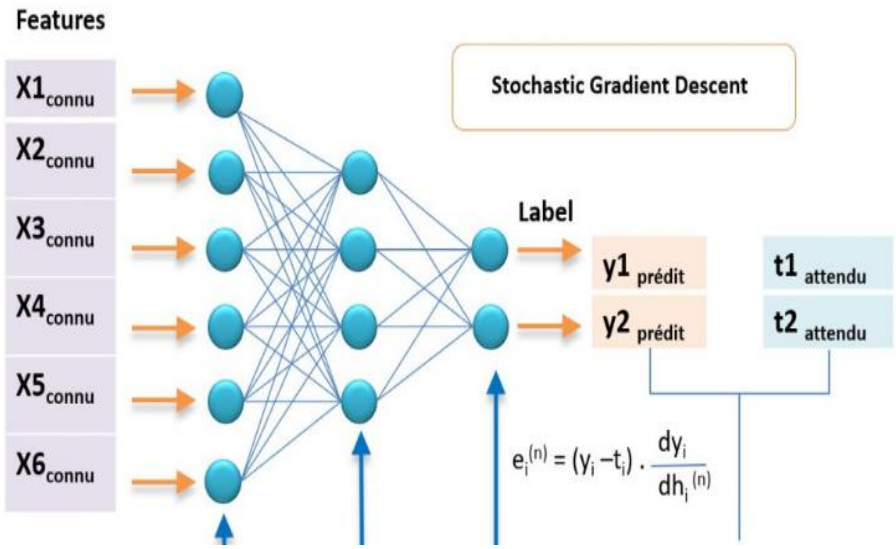


FIGURE I.21 : **La rétropropagation du gradient de l'erreur**

Voici une explication des différents paramètres et de l'équation présente dans la figure 21 :

1. Features (Caractéristiques) :

-$X1_{connu}$,$X2_{connu}$,$X3_{connu}$,$X4_{connu}$,$X5_{connu}$,$X6_{connu}$: Ce sont les variables d'entrée connues (ou caractéristiques) qui alimentent le réseau de neurones. ChaqueX_i représente une caractéristique spécifique des données d'entrée.

2. Réseau de Neurones :

- Les cercles bleus représentent les neurones. Ils sont organisés en plusieurs couches : une couche d'entrée, des couches cachées et une couche de sortie.

- Les flèches indiquent les connexions (ou poids) entre les neurones des différentes couches. Chaque connexion a un poids associé qui sera ajusté pendant l'apprentissage.

3. Label (Étiquette) :

• $y1_{\text{prédit}}$Et$y2_{\text{prédit}}$: Ce sont les valeurs de sortie prédites par le réseau de neurones pour les entrées données.

• $t1_{\text{attendu}}$Et$t2_{\text{attendu}}$: Ce sont les valeurs attendues (ou vraies valeurs) pour les sorties correspondantes.

Équation présente dans la figure :

$$e_i^{(n)} = (y_i - t_i).\frac{\partial_y}{\partial h_i^{(n)}} \qquad \text{I.7}$$

Cette équation représente la mise à jour des poids lors de l'apprentissage par la méthode de descente de gradient stochastique.

- $e_i^{(n)}$: Erreur associée au neurone ià l'itération **n**.

- y_i**:** Sortie prédite par le réseau pour le neurone i.

- t_i: Valeur attendue pour le neurone$i$.

- $\frac{\partial y_i}{\partial h_i^{(n)}}$: Dérivée partielle de la sortie$y_i$par rapport à l'entrée du neurone $h_i^{(n)}$ à l'itération **n**.

Processus d'Optimisation (Stochastic Gradient Descent) :

- La méthode de (descente de gradient stochastique) est utilisée pour minimiser la fonction de coût (ou erreur) en ajustant les poids du réseau.
- À chaque itération, les poids sont mis à jour en utilisant l'erreur calculée et la dérivée partielle correspondante.
- Cela permet au réseau de s'ajuster progressivement afin de réduire les erreurs de prédiction et de mieux correspondre aux valeurs attendues.

III.3 Réseaux de neurones convolutifs (CNN) :

Les réseaux de neurones convolutionels sont à ce jour parmi les modèles les plus performants pour classer des images. Désignés par l'acronyme CNN, de l'anglais Convolutional Neural Network, ils comportent deux parties bien distinctes comme le montre la figure I.22.

La première partie d'un CNN 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. Au final, les cartes de convolutions sont mises à plat et concaténées en un vecteur de caractéristiques appelé code CNN afin de les exploiter par la deuxième partie. [24]

La deuxième partie du réseau CNN réalise l'opération de classification où plus précisément de prédiction de la classe en se basant sur le vecteur de caractéristiques de la première partie. Elle est composée essentiellement d'une ou plusieurs couches de réseaux entièrement connectées (FC : Fully connected).

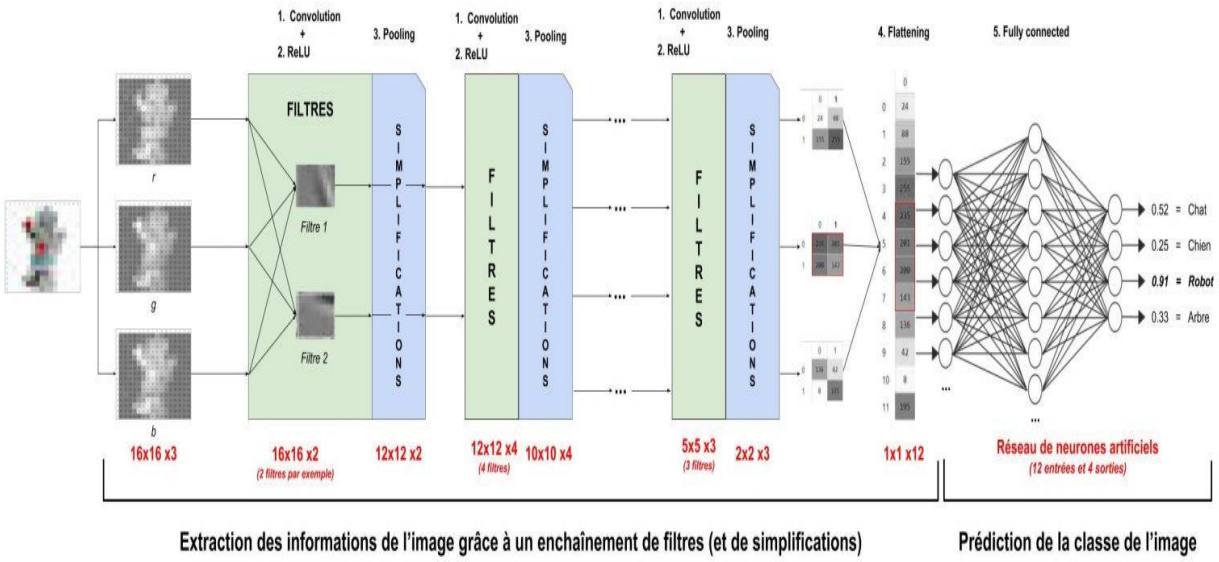


Figure I.22 : architecture de base d'un réseau CNN

III.3.1 Architecture de réseaux CNN :

Les réseaux de neurones convolutionnels sont basés sur le perceptron multicouche (MLP). /es MLP ont beaucoup de mal à gérer des images de grande taille, ce qui est dû à la croissance exponentielle du nombre de connexions avec la taille de l'image. Les CNN visent à limiter le nombre d'entrées tout en conservant la forte corrélation « spatialement locale » des images naturelles.

Comme illustré dans la figure I.23, l'architecture d'un réseau CNN se repose fondamentalement sur les couches suivantes :

- Couche d'entrée (Input layer).
- Couche de convolution (Convo layer : Convolution + ReLU).
- Couche de Pooling.
- Couche entièrement connectée (Couche Fully connected).
- Couche Softmax/logistique.
- Couche de sortie (Output layer).

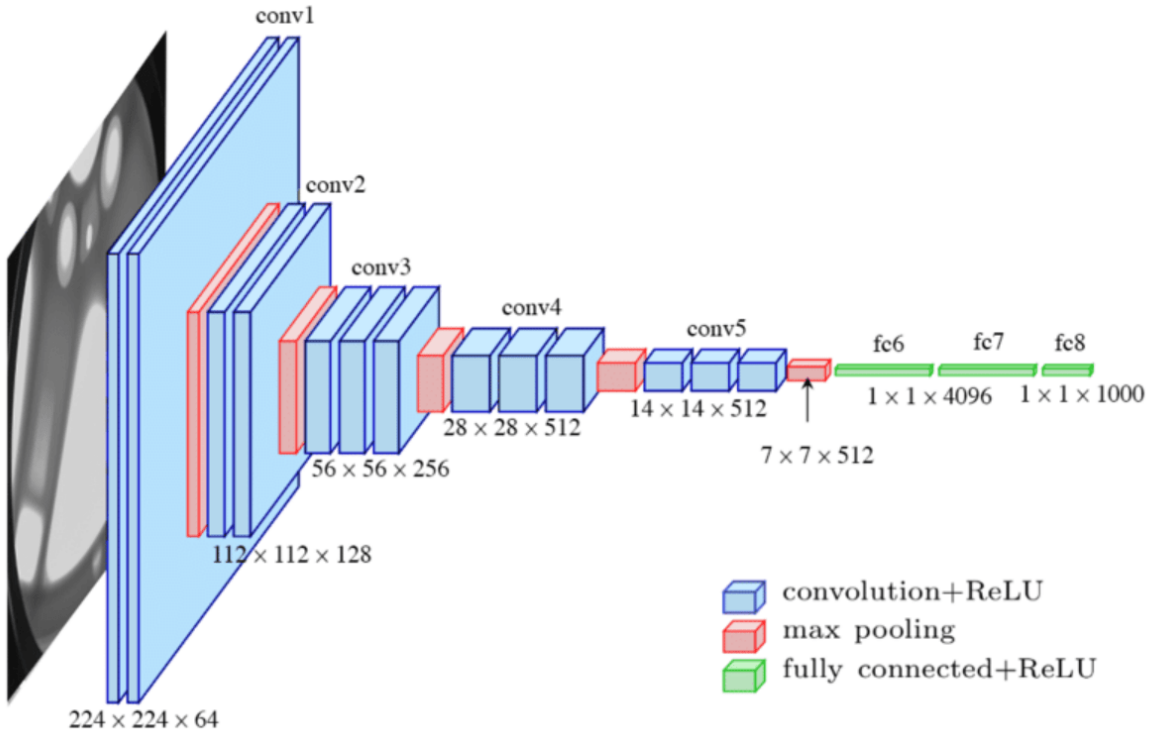
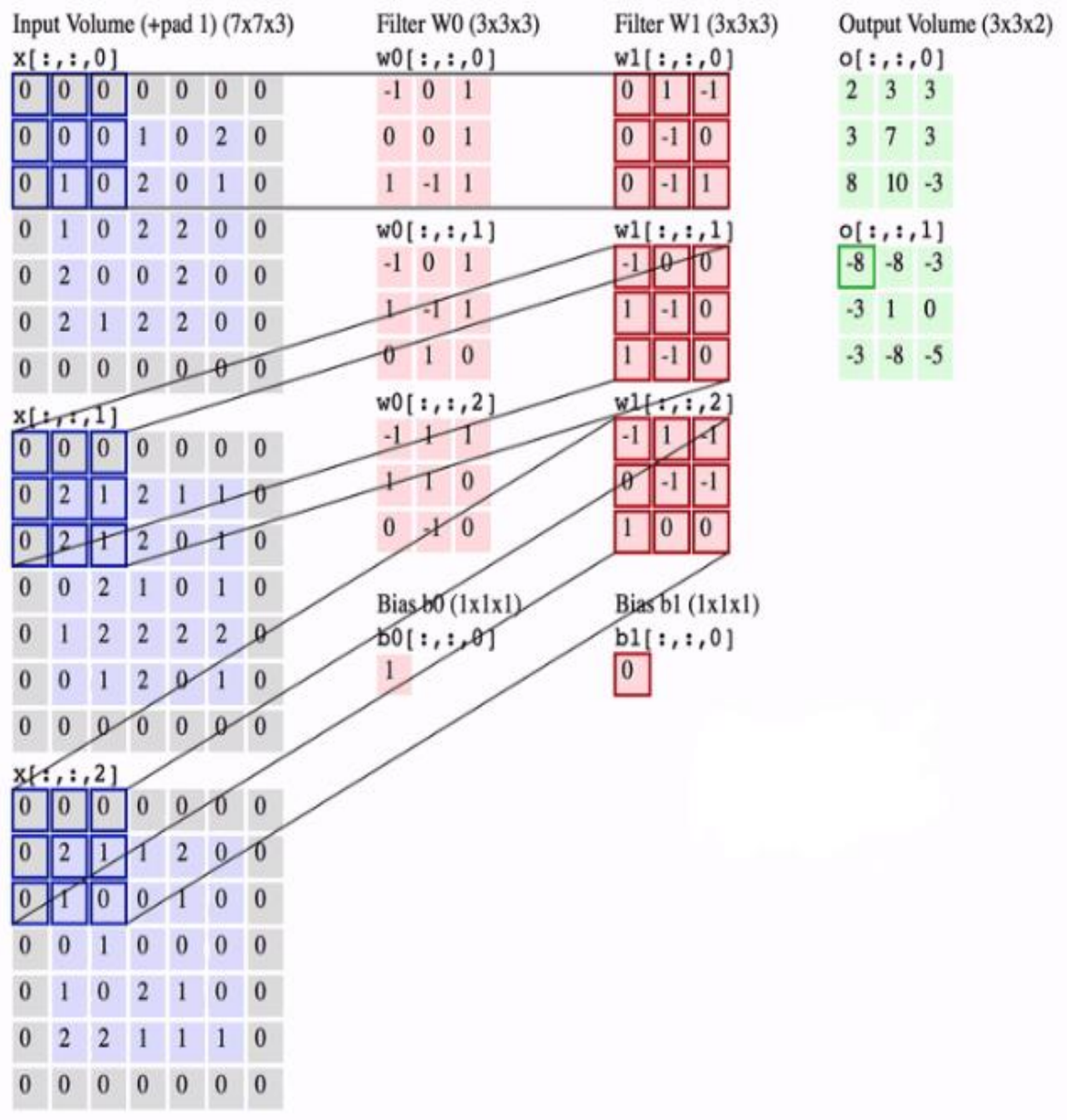


Figure I.23 : les différentes couches d'un réseau CNN

III.3.2 Couche convolutive (Convolutional layer) :

L'objectif de la couche de convolution est d'extraire les différentes entités et caractéristiques de l'image d'entrée en appliquant une opération de convolution entre les entités de caractéristiques d'entrées et les filtres spécifiques de la couche. L'idée de convolution est d'appliquer le Kernel (filtre) sur les pixels de l'image en le déplaçant par une entité de mesure appelée Stride. Si il est nécessaire de garder la même taille de la convolution on applique une opération de Padding qui consiste à ajouter des valeurs connues (zéro par exemple) sur les bordures de l'image. Après calcul de la convolution de chaque sortie on ajoute une valeur constante à chaque pixel de la sortie, cette valeur est appelée biais.

La figure I.24 met en évidence le principe de fonctionnement d'une couche de convolution de trois entrées et deux sorties. La couche possède deux groupes de filtre, dont chaque groupe est utilisé pour calculer une sortie. Chaque groupe contient trois filtres dont chaque filtre est convolué avec une entrée. Chaque sortie de convolution est additionnée à son propre biais. Dans cette exemple la taille de chaque filtre est 3×3 alors que la valeur du stride est égale à 2.



FigureI.24 : illustration du principe de fonctionnement d'une couche de convolution à trois entrées et deux sorties

III.3.3 Couche d'activation

Chaque couche de convolution envoie la sortie à une couche de fonction d'activation. La couche de fonction d'activation est composée d'une fonction d'activation qui prend la carte de caractéristiques créée par la couche de convolution et produit la carte d'activation comme sortie. En introduisant une combinaison non linéaire d'entités, la couche d'activation accélère le processus d'apprentissage des cartes d'entités.

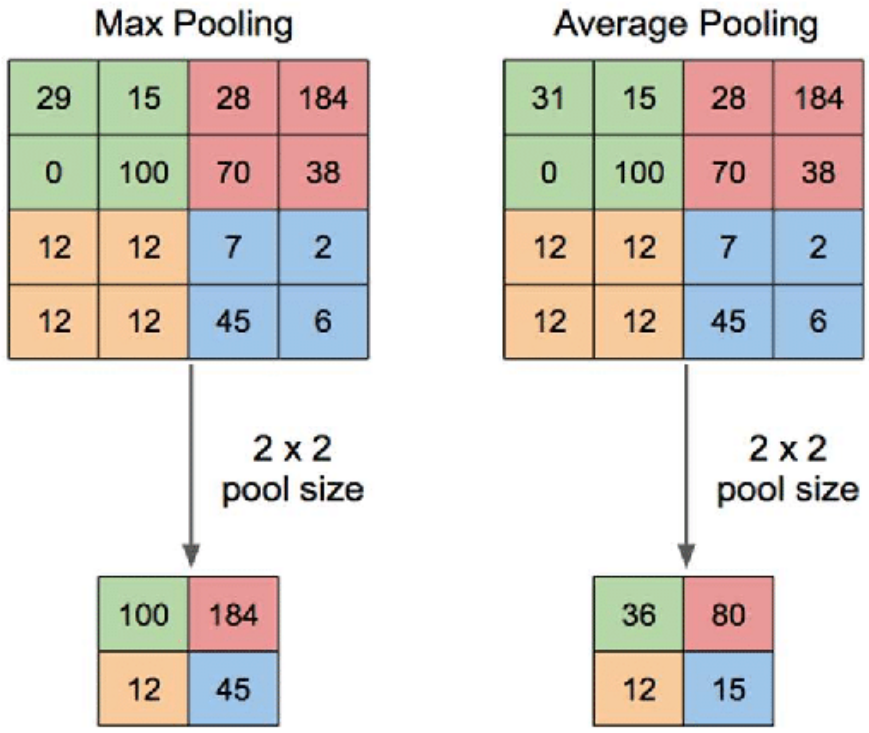
On a plusieurs types de fonctions d'activation, mais la plus utilisés dans les réseaux CNN est la fonction ReLU (Rectified linear unit). Elle est très efficace et permet un

apprentissage plus rapide. Une charge de calcul plus faible est le principal avantage de ReLU par rapport aux autres.

III.3.4 Couche de sous-échantillonnage (Pooling layer) :

La couche de pooling permet de réduire les dimensions des maps de caractéristiques calculées par la couche de convolution et d'activation. Elle supprime toutes les fonctionnalités et les caractéristiques redondantes capturées lors de la convolution et garde les plus importantes. Cela réduit progressivement la taille spatiale de la représentation afin de réduire le nombre de paramètres et de calculs dans le réseau. Pour réduire l'image, il existe deux types de couches de pooling :

- **Max Pooling :** où on définit un voisinage spatial (un filtre) qui se glisse dans l'entrée et on prend l'élément le plus important de la région couverte par le filtre.
- **Average Pooling** : comme son nom l'indique, il conserve la moyenne des valeurs rencontrées dans le filtre.



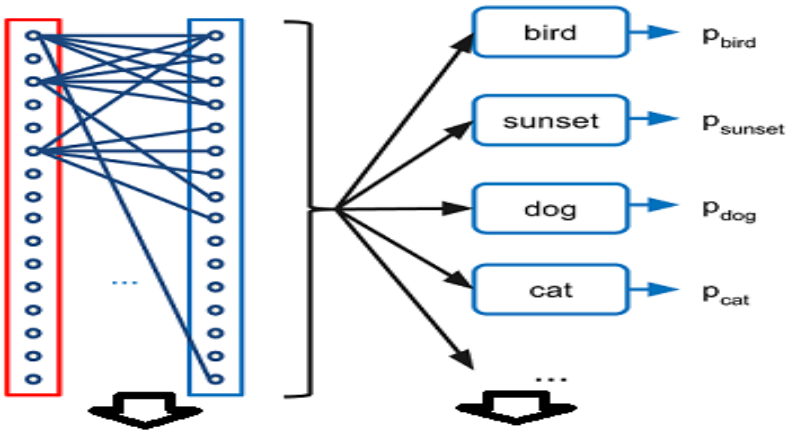
FigureI.25 : les deux types de sous échantillonnage (Pooling)

III.3.5 Couche entièrement connectée (Fully-Connected Layer) :

La couche entièrement connectée, ou couche dense, est une est une couche pour laquelle chaque neurone est connecté à tous les neurones de la couche précédente. Cela signifie que chaque sortie de la couche précédente est pondérée et sommée par chaque neurone de la

couche entièrement connectée. Cette couche joue un rôle crucial dans la transformation des caractéristiques extraites par les couches convolutionnelles en une représentation adaptée à la tâche de classification.

Les poids et les biais sont les principaux paramètres appris par une couche entièrement connectée pendant le processus d'entraînement du réseau neuronal. Un réseau CNN possède une ou plusieurs couches FC. La sortie de la dernière couche entièrement connectée du réseau CNN est souvent utilisée pour générer des prédictions ou des scores pour différentes classes comme le montre la figure I.26. Le calcul du score ou de la probabilité d'appartenance d'une entrée aux différentes classes est assuré en utilisant une fonction d'activation appelée Softmax donnée précédemment dans l'équation I.6.



FigureI.26 : structure de la couche Fully Connected (FC).

III .4 Exemples de réseaux CNN :

En se basant sur la structure standard discutée ci-dessus, les réseaux CNN ont connu plusieurs transformations et évolutions. Plusieurs architectures de réseaux CNN ont été proposées pour divers types d'application. Voici quelques exemples de ces architectures.

- **LeNet** : Les premières applications réussies des réseaux convolutionnels ont été développées par Yann Lacune dans les années 1990. Parmi ceux-ci, le plus connu est l'architecture LeNetutilisée pour lire les codes postaux, les chiffres, etc. la figure I.27 donne l'architecture du réseau LeNet-5

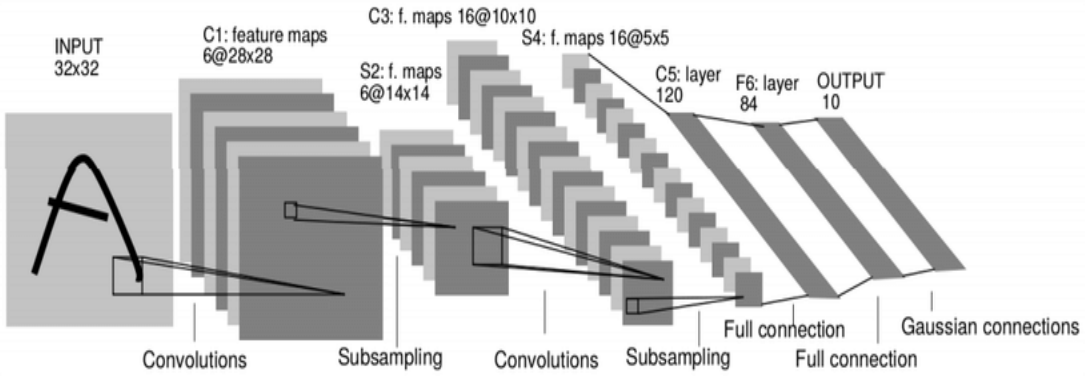


Figure I.27 : architecture du réseau LeNet-5

- **AlexNet** : c'est le premier travail qui a popularisé les réseaux convolutionnels dans la vision par ordinateur, il est développé par Alex Krizhevsky, Ilya Sutskever et Geoff Hinton. L'AlexNet a été soumis au challenge ImageNet ILSVRCen 2012 et a nettement surperformé le deuxième finaliste (erreur de top 5 de 16% par rapport à la deuxième place avec une erreur de 26%). Le réseau avait une architecture très similaire à celle de LeNet, mais il était plus profond, plus grand et comportait des couches convolutives empilées les unes sur les autres (auparavant, il était courant de n'avoir qu'une seule couche CONV immédiatement suivie d'une couche POOL)

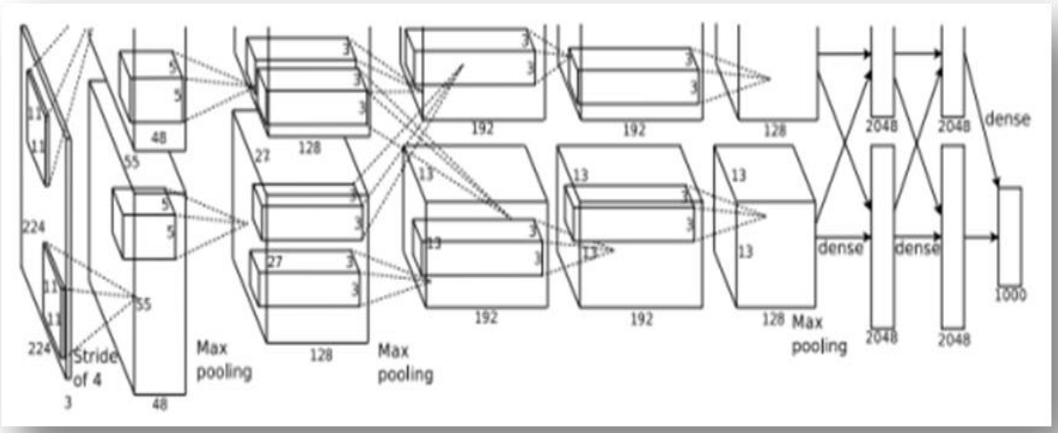


Figure I .28 : architecture du réseau AlexNet

- **ZF Net** : Le gagnant de l'ILSVRC 2013 était un réseau convolutionnel de Matthew Zeiler et Rob Fergus. Il est devenu connu sous le nom de ZFNet (abréviation de Zeiler & Fergus Net). C'était une amélioration de AlexNet en modifiant les hyperparamètres de l'architecture, en particulier en augmentant la taille des couches de convolution moyennes et en réduisant la taille des pas et des filtres sur la première couche.

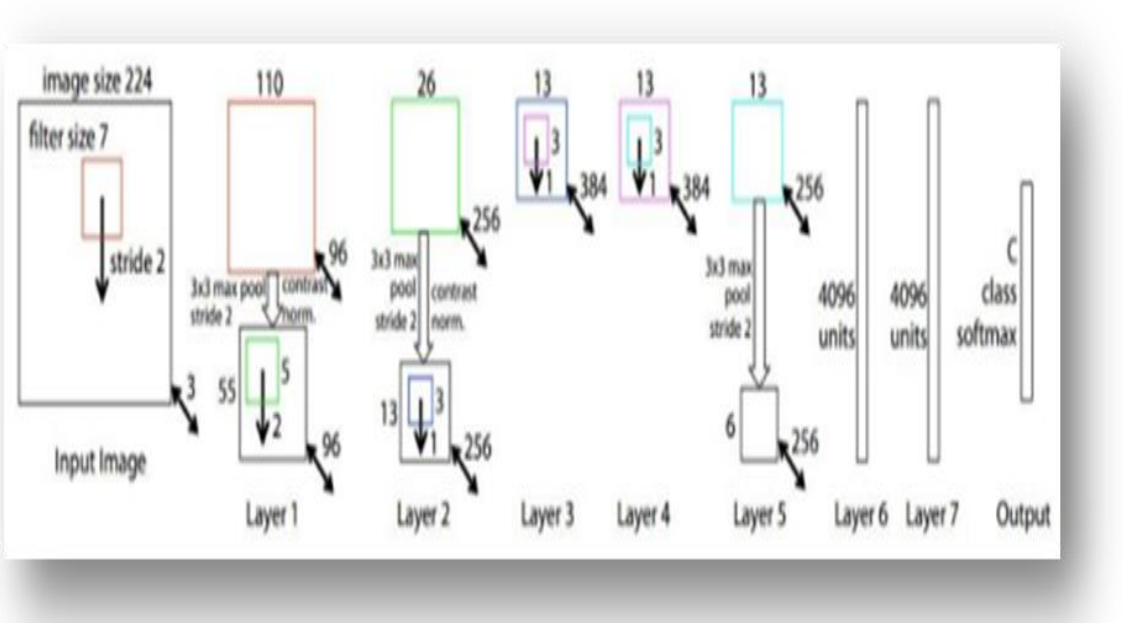


Figure I.29 : architecture du réseau ZFNet

• **Google Net : Le** gagnant de l'ILSVRC 2014 était un réseau convolutionnel de Szeged et al. Sa principale contribution a été le développement d'un *module de lancement* qui a considérablement réduit le nombre de paramètres dans le réseau (4M, comparé à Alex Net avec 60M). En outre, ce réseau utilise le regroupement moyen au lieu des couches entièrement connectées en haut du ConvNet, éliminant une grande quantité de paramètres qui semblent peu importants.

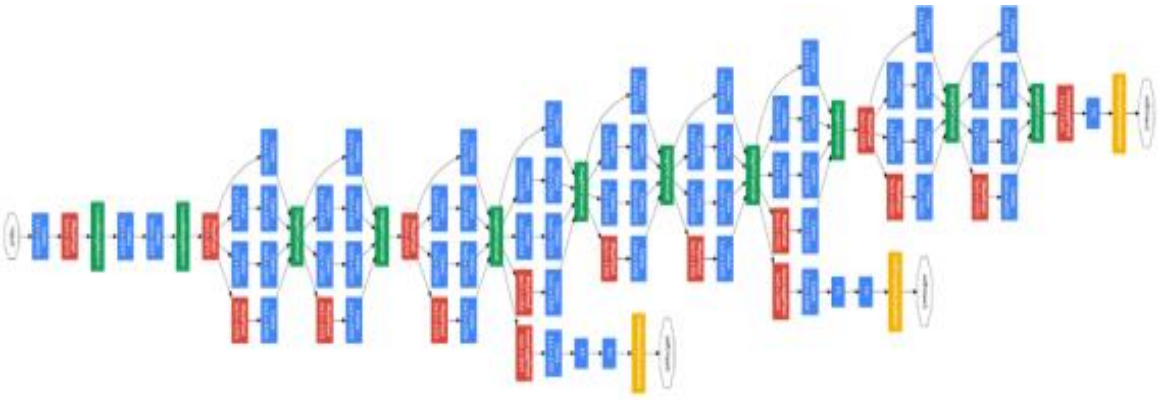


Figure I.30 : architecture du réseau Google Net

• **VGGNet : La** deuxième place dans ILSVRC 2014 était le réseau de Karen Simonyan et Andrew Zisserman, connu sous le nom de VGGNet .Sa principale contribution a été de montrer que la profondeur du réseau est un composant essentiel pour de bonnes performances. Leur meilleur réseau final contient 16 couches CONV / FC et, de manière attrayante, présente une architecture extrêmement homogène qui ne réalise que 3 x 3 convolutions et un regroupement 2x2 du début à la fin. Leur modèle prétentieux est disponible

pour une utilisation plug and Play dans Caffe. Un inconvénient du VGGNet est qu'il est plus coûteux à évaluer et utilise beaucoup plus de mémoire et de paramètres (140M). La plupart de ces paramètres se trouvent dans la première couche entièrement connectée, et il a depuis été constaté que ces couches FC peuvent être supprimées sans dégradation des performances, ce qui réduit considérablement le nombre de paramètres nécessaires.



Figure I.31 : architecture du réseau VGGNet.

- **ResNet** : Réseau résiduel développé par Kaiming He et al. A été le gagnant d'ILSVRC2015. Il propose des *connexions* spéciales et un usage intensif de la normalisation des lots . L'architecture ne contient pas de couches entièrement connectées à la fin du réseau. ResNets sont actuellement des modèles de réseaux neuronaux convolutifs de pointe et constituent le choix par défaut pour l'utilisation de ConvNets dans la pratique.



Figure I.32 : architecture du réseau ResNet

Conclusion :

La classification des images est l’une des disciplines les plus importante du domaine de la vision par ordinateur. Elle consiste à attribuer une classe à une image parmi un ensemble de classe possible. La classification d'images est omniprésente dans de nombreux aspects de la vie moderne, facilitant la recherche, l'organisation, la sécurité et l'automatisation dans divers domaines. Dans ce chapitre nous avons présenté les bases de la classification d'images à l'aide des réseaux de neurones convolutifs (CNN), soulignant leur rôle crucial dans le traitement d'image moderne. Nous avons vu comment les CNN ont transformé notre capacité à analyser et à classer des images, grâce à leur architecture spécifique qui permet d'extraire des caractéristiques complexes. Les architectures de CNN, telles que LeNet, AlexNet, et ResNet, ont été évoquées, illustrant l'évolution constante de ce domaine. Ces réseaux ont non seulement amélioré la précision de la classification d'images mais ont également ouvert la voie à de nouvelles applications et recherches.

Chapitre 2 :

Etude des circuits logiques programmables FPGA

I. Introduction

L’implémentation des réseaux de neurones convolutionnels (CNN) sur un circuit intégré pose plusieurs défis majeurs. Les CNN se distinguent par leurs opérations de calcul intensif, notamment les convolutions, qui requièrent une grande quantité de ressources de calcul et entraînent une consommation de puissance significative. De plus, ces réseaux manipulent des volumes de données considérables, nécessitant une gestion de la mémoire efficace pour minimiser les temps d'accès et maximiser l'utilisation des ressources disponibles.

L'implémentation efficace des réseaux CNN sur des plates-formes matérielles est une tache critique pour les ingénieurs et les chercheurs. Les circuits logiques programmables de type FPGA offrent une flexibilité et une architecture massivement parallèle ce qui les rend très bénéfiques pour accélérer les opérations intensives des réseaux CNN.

Les circuits de type Zynq combinent des processeurs programmable ARM avec des ressources logique configurables (FPGA) offrant ainsi une plate-forme convenable pour les réseaux CNN. En effet, les circuits Zynq offrent une combinaison unique de performances élevées, de flexibilité, de programmabilité, de reconfigurabilité et de faible consommation de puissance.

Ainsi, au cours de ce chapitre on va commencer par la présentation des circuits logiques programmables FPGA notamment du constructeur Xilinx. Puis, on va donner un aperçu général de la famille Zynq où on abordera l’architecture et composants de base de cette architecture. Enfin, on examinera de plus près le cadre open source Pynq où on discutera de son concept général et de ses éléments de bases.

III. Evolution des circuits intégrés numériques

L’évolution des circuits intégrés numériques a commencé avec l’invention du Transistor qui a remplacé les tubes à vide en 1947 par John Bardeen, Walter Brattain et William Shockley. Depuis cet évènement historique, plusieurs inventions en marqués le processus d’évolution. Dans ce qui suit on donne un aperçu de ces inventions :

II.1 Microprocesseurs (1970s)

Le premier microprocesseur commercial, l'Intel 4004, a été introduit en 1971. Il a intégré l'unité centrale de traitement (CPU) sur une seule puce, ouvrant la voie à la révolution des

micro-ordinateurs. Depuis cette date, des entreprises comme Intel, Motorola et Zilog ont développé des microprocesseurs de plus en plus puissants, rendant les ordinateurs personnels accessibles au grand public. Les architectures des microprocesseurs sont basées sur les éléments suivants :

1. Unité Centrale de Traitement (CPU) :

La CPU est responsable de l'exécution des instructions et de la gestion des opérations logiques et arithmétiques. Elle se compose de plusieurs sous-éléments comme : l'unité de Contrôle (Control Unit), l'unité Arithmétique et Logique (ALU), les registres le compteur de Programme (Program Counter)

2. Mémoire : sont rôle est de stocker les données et les instructions d'une façon permanente ou temporaire. On trouve plusieurs types de mémoire dans les micro processeurs comme : mémoire Cache, mémoire RAM (Random Access Memory),mémoire ROM (Read-Only Memory)

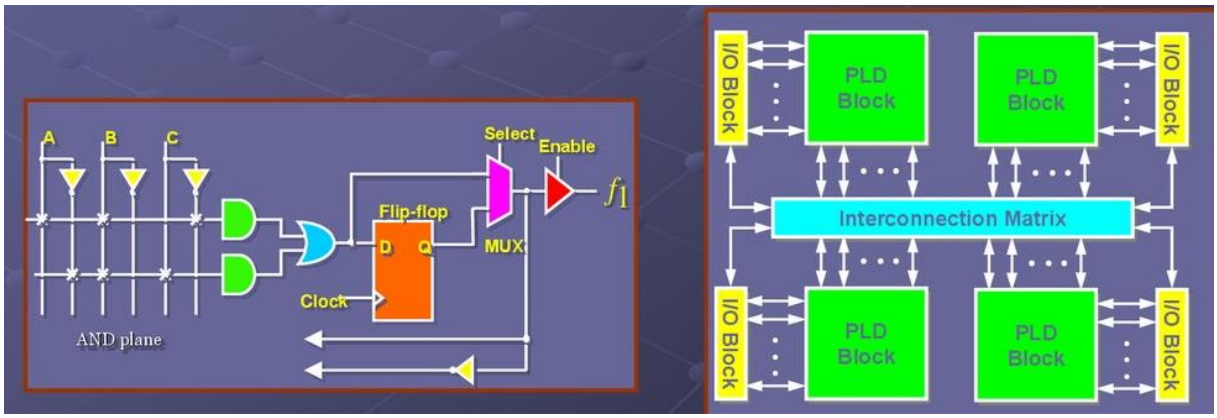
3. Bus de Communication : Les bus sont des lignes qui transportent les données, les adresses et les signaux de contrôle entre les différentes parties du microprocesseur et d'autres composants du système. Ainsi on trouve : des bus de Données, des bus d'Adresses, et des bus de Contrôle.

II.2 Les circuits programmables PLD (Programmble Logic Device) :

Les circuits PLD sont des circuits formés de cellules logiques (comportant des fusibles, des antis fusibles ou de la mémoire) utilisés pour implémenter des fonctions logiques personnalisées et qui peuvent être programmés et parfois reprogrammés par l'utilisateur.

Selon leur niveau de complexité, les PLD peuvent être de type simple (SPLD) ou complexe comme la montre la figure II.1. Les SPLD à leur tour se décline en PLA (Programmable Logic Array) et PAL (Programmable Array Logic). Les PLA comprennent une matrice programmable de portes AND suivie d'une matrice programmable de portes OR. Les PAL sont similaires au PLA, mais avec une matrice de portes OR fixe et ils sont plus rapides et plus simples que les PLA.

Les CPLDs sont formés par une combinaison de plusieurs blocs de type SPLD sur une seule puce, interconnectés par une matrice de commutation programmable. Ils offrent une plus grande capacité de logique que les SPLD et peuvent implémenter des fonctions logiques plus complexes.



a b

Figure II. 1 : Architecture globale d'un circuit PLD (a) : SPLD (b) : CPLD

II.3 Les circuits spécifiques à l'application ASIC (Application Specific Integration circuit) :

Les circuits ASIC constituent la troisième génération des circuits intégrés, apparu au début des années 80. Ils sont des circuits personnalisés par leur fabriquant et conçues pour accomplir des tâches spécifiques plutôt que d'exécuter des opérations générales comme les microprocesseurs ou les FPGA. Ces puces sont couramment utilisées dans des produits nécessitant des performances élevées, une consommation d'énergie réduite, et un coût par unité faible pour des volumes de production importants.

Les circuits ASIC sont connus par leurs performances optimisées pour des tâches spécifiques, par leurs efficacité Énergétique et par leur protection contre la contrefaçon.

Vue leurs avantages multiples, les circuits ASIC sont utilisés dans plusieurs applications comme :

Électronique Grand Public : par exemple dans le domaine de smartphones les ASIC sont utilisés dans les processeurs d'applications, les modems, et les capteurs, optimisés pour des performances élevées et une faible consommation d'énergie.

Automobile : spécialement dans les systèmes de sécurité et d'assistance au conducteur comme le freinage, les airbags, les systèmes de navigation et de divertissement embarqués.

Télécommunications : Utilisés dans les routeurs, les commutateurs, et les équipements de réseau pour des performances de traitement des données optimisées.

Industrie : Utilisés dans les équipements industriels pour le contrôle de processus, la robotique, et les systèmes de surveillance.

III. Circuits logiques programmables FPGA :

III.1 introduction :

Les FPGAs, soit en Anglais « Field Programmable Gate Array » sont des circuits numériques reconfigurables (ou programmables) par l'utilisateur contrairement au circuits ASIC qui sont configurable une seule fois par leur fabriquant. Les premières versions de circuits FPGAs ont été proposées au début des années 80s. depuis cette date, les FPGAs ont connu une évolution croissante, avec des progrès significatifs dans la densité, la performance et les fonctionnalités, ce qui en fait aujourd'hui des composants essentiels dans de nombreux systèmes électroniques complexes. La configuration des circuits FPGA est réalisée à travers l'utilisation de langages de description matériels comme le cas du langage VHDL (Very high speed integrated circuits Hardware Description Language). [17]

Les circuits FPGA actuels peuvent être composés de plusieurs millions de portes logiques et de bascules. Les dernières générations de FPGA intègrent même de la mémoire vive (RAM), des unités de calculs optimisées et même des processeurs embarqués.

Deux grandes firmes dominent actuellement le marché des FPGA :

- **Xilinx-AMD :** c'est l'un des plus grands fabricants de FPGA au monde. Fondée en 1984, la société a été pionnière dans le domaine des FPGA et reste un leader dans l'industrie. Ses produits phares incluent les séries Virtex, Kintex, Artix et Zynq, offrant une large gamme de performances et de fonctionnalités pour diverses applications.
- **Altera-Intel :** Altera était un autre grand fabricant de FPGA avant d'être acquise par Intel en 2015. La société était connue pour ses séries de produits Cyclone, Arria et Stratix, qui étaient utilisées dans une variété d'applications, de l'informatique embarquée à l'informatique haute performance.

III.2Architecture des circuits FPGA :

Un FPGA est un circuit à densité d'intégration très élevée (VLSI : Very-Large-Scale Integration). Son architecture est basée sur des blocs logiques programmables qui peuvent être interconnectés par programmation au choix de l'utilisateur afin de réaliser les fonctions désirées.

Comme illustré dans la figure II.2, les briques de base dans l'architecture des circuits FPGAs sont : [17].

- ✓ Les blocs logiques programmables.
- ✓ Les blocs d'entrées/sorties.
- ✓ Le réseau d'interconnexion

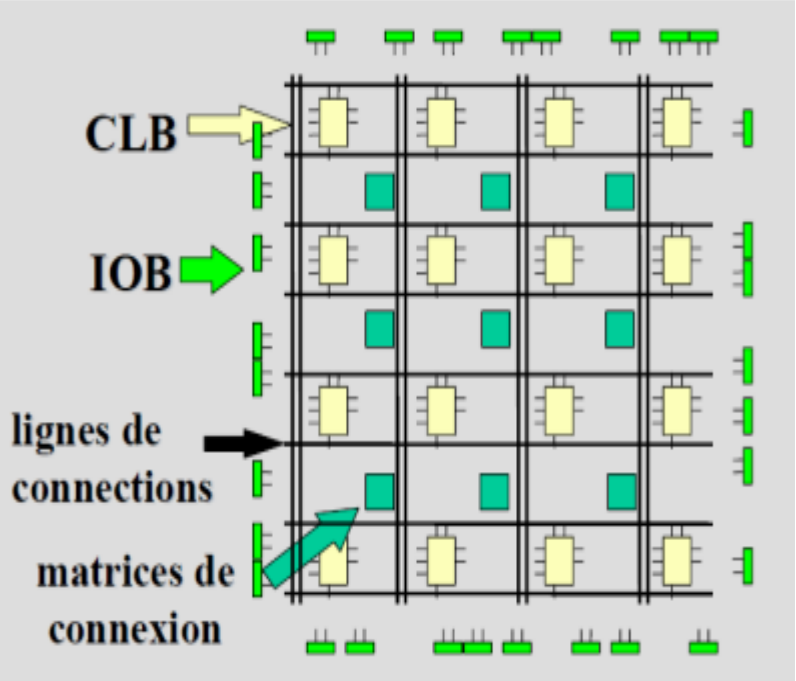


Figure II.2 : Architecture générale d'un FPGA

III.2.1Blocs logiques programmables :

Les blocs logiques programmables ou très précisément configurables (CLB : Configurable Logic Block) sont des blocs conçus pour l'implémentation des fonctions logiques combinatoires et séquentielles.

La figure II.3 donne les éléments de base d'un bloc CLB qui est souvent formés des éléments :

- **LUT (Look-Up Table) :** Le cœur du bloc CLB est généralement une table de recherche (LUT), qui est un élément logique configurable. Cette LUT peut être programmée pour implémenter n'importe quelle fonction logique de 1 à n bits (généralement la valeur de n est de 4 à 6 bits dans les FPGA modernes).
- **Registres :** Un bloc CLB peut également contenir des éléments de stockage, tels que des bascules D (flip-flops), utilisées pour stocker des valeurs intermédiaires ou temporaires dans un circuit. Ces registres peuvent être utilisés pour synchroniser des signaux ou pour mémoriser des données.

- **Multiplexeurs (MUX) :** Des multiplexeurs peuvent être présents pour sélectionner les entrées des LUTs ou des registres, ce qui permet une configuration flexible des interconnexions internes du bloc CLB.

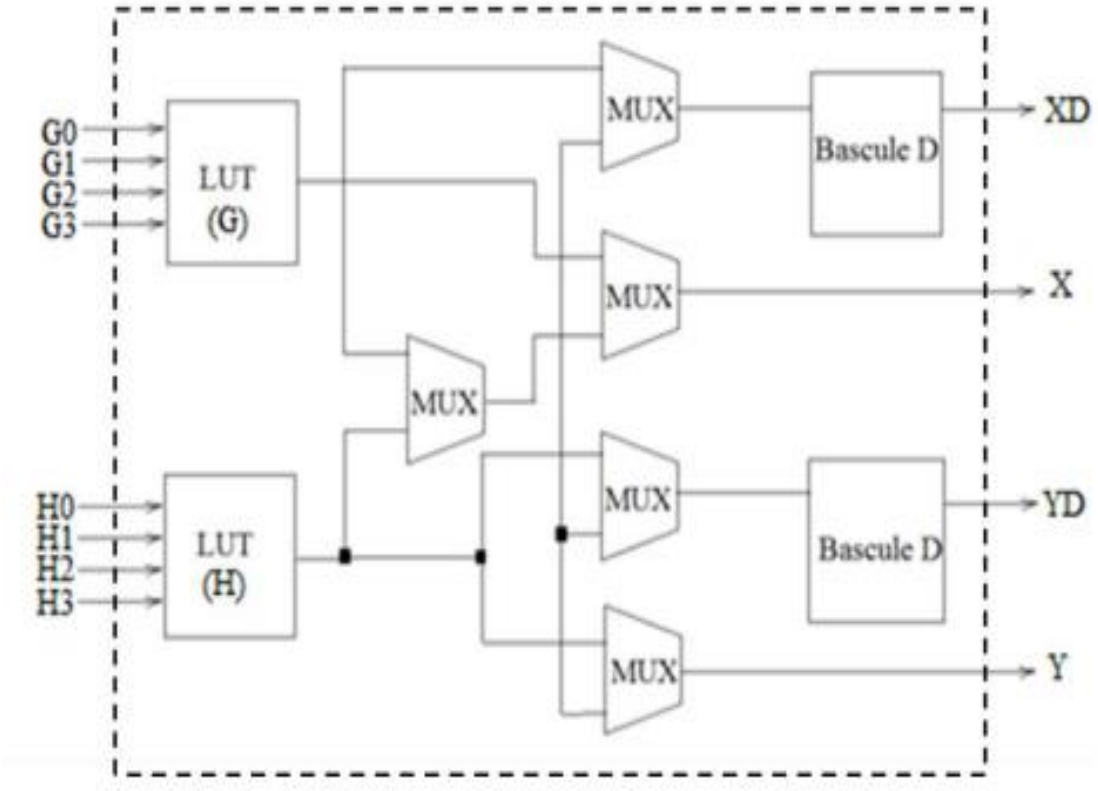


Figure. II.3 : Structure simplifiée d‟un bloc logique programmable (CLB)

III.2.2 Blocs d'entrées/sorties :

Ses blocs permettent l'interfaçage des données entre les broches du circuit et la structure interne du composant. Chaque bloc permet de configurer la pine du circuit en entrée, en sortie ou à l'état bidirectionnel.

La figure II.4 donne la structure d'un IOB dans le cas des FPGAs de la série XC4000E. Ce bloc IOB est basée sur l'utilisation de deux bascules permettent de prendre en charge le signal entrant ou sortant. Ces deux derniers signaux se configurent à l'état combinatoire ou séquentiel. On trouve aussi deux Buffers pour les deux signaux (d'entrée ou sortie) où le buffer de sortie à trois-états. Les deux transistors sont utilisés pour l'adaptation des signaux en TTL ou en CMOS. Un temporisateur est utilisé afin de créer un délai en entrée permettant ainsi de compenser les délais de distribution du signal d'horloge interne.

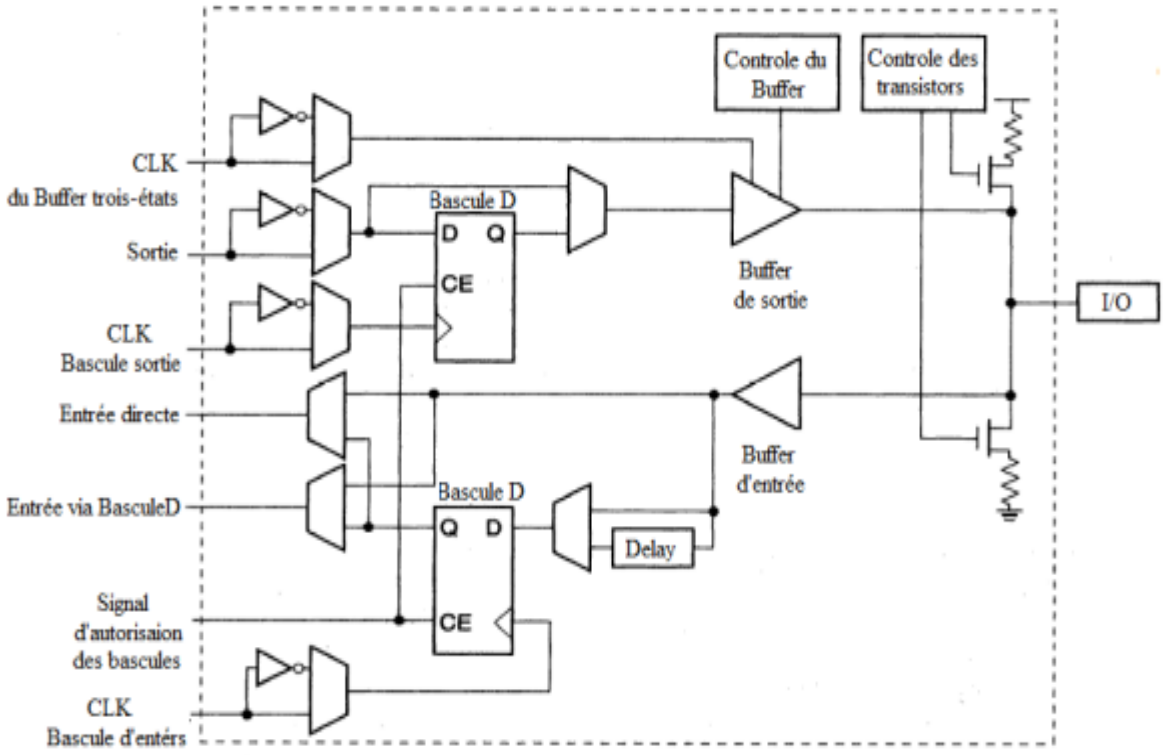


Figure. II.4 : Structure d'un bloc d'entrée/sorties (IOB).

III.2.3 Réseau d'interconnexion :

Les réseaux d'interconnexion dans les FPGA jouent un rôle crucial dans la connectivité entre les différents blocs logiques, les entrées/sorties et d'autres ressources du FPGA. Les interconnexions (ou routage) sont réalisées à travers des matrices souvent appelées matrices d'interconnexion (MC) comme l'indique le schéma de la figure II.5. Ces matrices sont disposées entre les blocs logiques configurables (CLB).

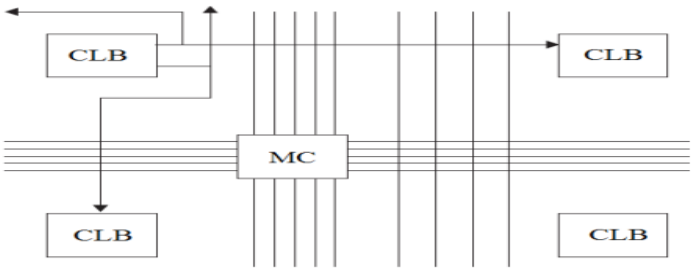


Figure. II.5 : Disposition des matrices d"interconnexion dans un FPGA.

Les circuits FPGA modernes disposent de plusieurs types d'interconnexions comme :

- **Interconnexions Locales (Local Interconnect) :** Les interconnexions locales relient les différents blocs logiques au sein d'une même région du FPGA, tels que les blocs CLB (Configurable Logic Blocks), les blocs DSP (Digital Signal Processing), et les blocs de mémoire. Ces interconnexions sont souvent utilisées pour les communications internes au sein d'une même zone logique.
- **Interconnexions Globales (Global Interconnect) :** Les interconnexions globales relient les différentes régions du FPGA entre elles et avec les entrées/sorties du FPGA. Elles permettent aux signaux de traverser l'ensemble du FPGA, offrant une connectivité étendue entre les différentes ressources du dispositif.
- **Interconnexions Directes (Direct Interconnect) :** Les interconnexions directes permettent de relier directement deux ressources sans passer par les réseaux de routage standard. Cela peut être utilisé pour fournir une communication rapide entre des éléments spécifiques du FPGA.
- **Bus de Communication (Communication Bus) :** Certains FPGA comportent des bus de communication dédiés pour permettre la transmission de données entre différentes ressources, telles que des bus AXI (Advanced eXtensible Interface) ou des bus PCIe (Peripheral Component Interconnect Express).

III.2.4 Blocs de mémoires et multiplicateurs :

L'évolution des performances technologiques des circuits FPGA nécessite l'intégration de différents modules comme les mémoires et les multiplicateurs. Ces blocs sont intégrés dans les circuits FPGA aux voisinages de CLBs et par colonnes (figure II.6).

Les BRAM (Block RAM) sont des blocs de mémoire intégrés directement dans le FPGA. Ils sont généralement organisés en blocs de mémoire de taille fixe (par exemple, 18Kb ou 36Kb) et peuvent être configurés pour fonctionner comme RAM simple port, double port, ou RAM distribuée.

Les multiplicateurs sont disponible sous forme de blocs DSP qui sont des unités de traitement dédiées, optimisées pour les opérations mathématiques courantes comme la multiplication et l'accumulation (Multiply-Accumulate, MAC). Ils peuvent effectuer des multiplications, des additions et d'autres opérations arithmétiques en parallèle et avec une latence faible.

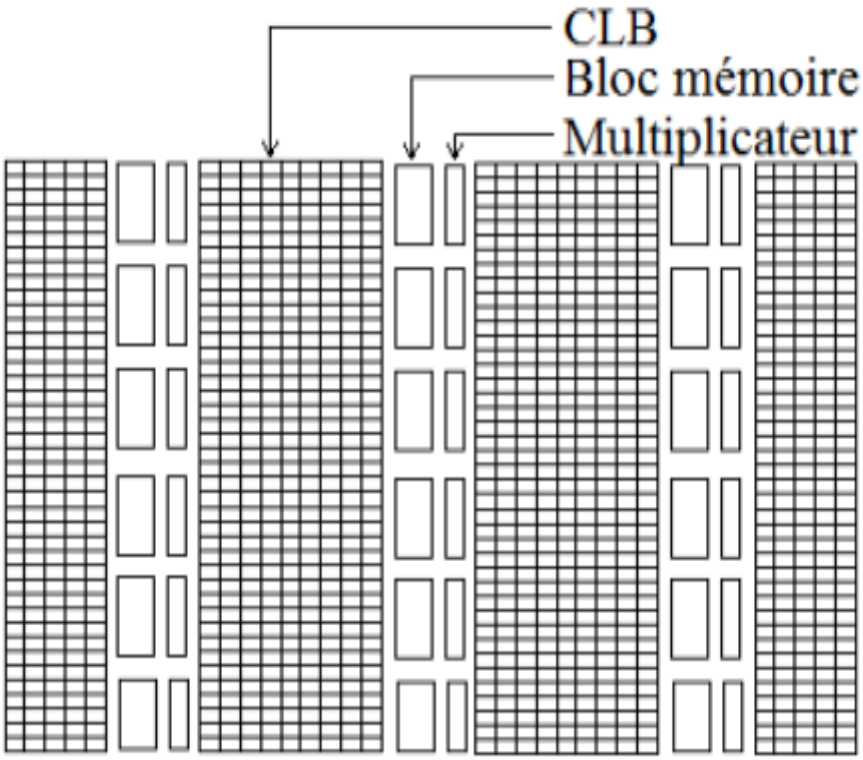


Figure. II.6 : Disposition des blocs mémoires et des multiplicateurs dans un FPGA.

III.3 Exemples de circuits FPGAde Xilinx-AMD :

Le fabricant Xilinx est l'un des pionniers dans le domaine des FPGAs où les performances et les capacités de ses dispositifs programmables couvrent des niveaux modestes à extrêmement élevés. Les circuits FPGA du Xilinx, structurés sous forme de matrices, utilisent deux types de cellules de base, les cellules -d'entrées/sorties appelées IOB (Input/Output Blocks). Ces différentes cellules sont reliées entre elles par un réseau d'interconnexions programmable

- **Virtex UltraScale+ VU19P** : Destiné aux applications nécessitant une capacité et une performance maximales, comme les centres de données et les réseaux de télécommunications.

- **Virtex-7 VX690T** : Offre une haute densité logique et est utilisé dans des systèmes de communication avancés et des équipements de test et de mesure.

- **Kintex UltraScale+ KU040** : Conçu pour un bon équilibre entre performance et coût, utilisé dans des équipements de communication sans fil et des systèmes de vidéosurveillance.

- **Kintex-7 KC705** : Offre une performance élevée avec une consommation d'énergie modérée, adapté aux systèmes de vision industrielle et de transmission de données.

- **Artix UltraScale+ AU10P** : Idéal pour les applications à faible consommation d'énergie avec des performances respectables, comme les dispositifs IoT et les applications industrielles.

- **Artix-7 A200T** : Conçu pour les applications à faible coût et faible consommation d'énergie, utilisé dans les dispositifs portables et les systèmes de surveillance.

- **Spartan-7 XC7S50** : Offre une solution FPGA à faible coût pour les applications d'entrée de gamme, utilisé dans les systèmes de contrôle industriels et les interfaces utilisateur.

- **Zynq UltraScale+ MPSoC ZU9EG** : Combine FPGA et processeurs multicœurs (ARM Cortex-A53, ARM Cortex-R5), idéal pour les systèmes embarqués complexes dans les domaines automobile et médical.

- **Zynq-7000 ZC706** : Intègre un processeur ARM dual-core avec un FPGA, offrant une solution complète pour les systèmes embarqués, utilisé dans les drones et les équipements de test.

- **Alveo U250** : Conçu pour les accélérations de centre de données, offrant une performance maximale pour des charges de travail exigeantes telles que l'apprentissage automatique et le traitement de données en temps réel.

- **Alveo U50** : Offre une accélération polyvalente pour les applications d'IA, de calcul haute performance et de mise en réseau.

- **Versal AI Core VC1902** : Destiné aux applications d'intelligence artificielle et de calcul haute performance, combinant des capacités FPGA avec des cœurs AI dédiés.

- **Versal Prime VM1302** : Offre une haute performance et une grande flexibilité pour les applications de mise en réseau avancées et les systèmes de traitement de signal.

III.3.1 La famille FPGA Virtex-7 :

La série 7 des circuits FPGA de Xilinx-AMD comprennent quatre familles offrant un choix varier et flexibles en fonction de performances voulu, du cout de réalisation et de la puissance consommée.

- La famille Spartan-7 est proposée au prix le moins chère de la série 7 avec un nombre limité de ressources. Elle est donc convenable pour les applications grand public qui ne nécessitent pas de calcul intensif.

- La famille Artix-7 est optimisée pour les performances par watt et la bande passante par watt les plus élevées pour les applications sensibles aux coûts.
- La famille Kintex-7 est une classe innovante de FPGA optimisés pour le meilleur rapport qualité-prix.
- La famille Virtex-7 est connue par l'intégration d'un nombre très élevée de ressources surtout en mémoire et en DSP. Elle est optimisée pour atteindre des performances maximales.

L'architecture interne de la famille Virtex 7 de Xilinx est illustrée dans la Figure II.7.

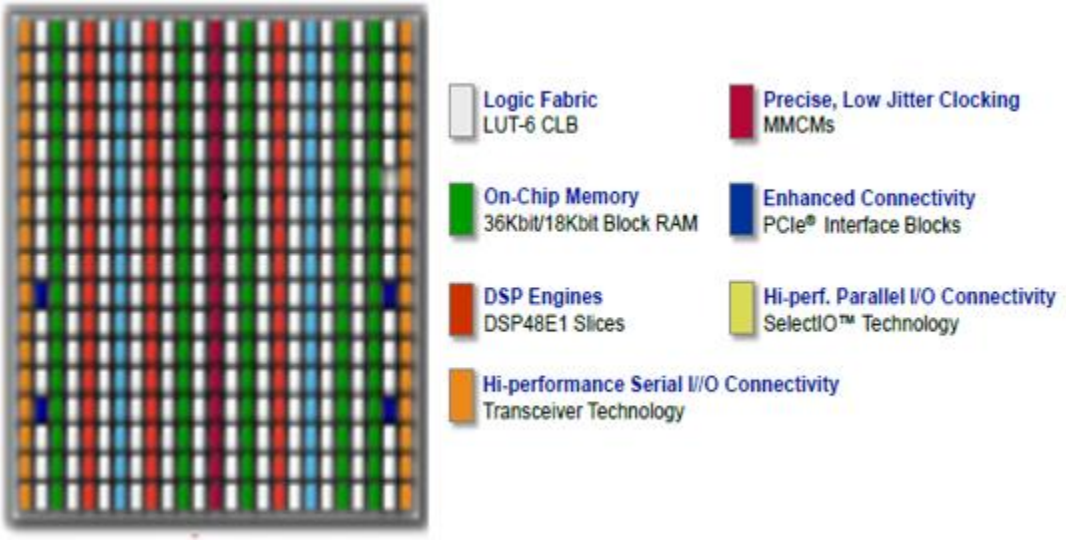


Figure II.7 : Architecture interne de la famille Virtex 7 de Xilinx.

Chaque CLB de la famille Virtex 7 contient deux slices ; nommée slice (1) et slice (0) ; indépendants et organisés en colonne comme la montre la figure II.8.

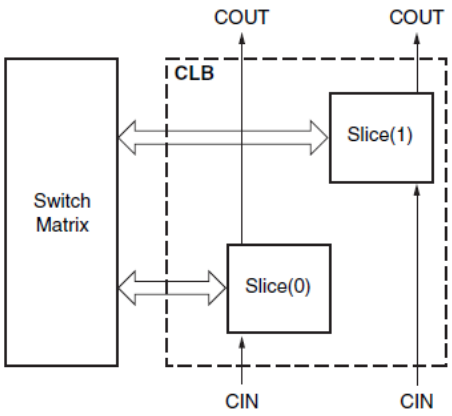


Figure II.8 : Arrangement des slices dans un CLB pour la série 7 de Xilinx.

Selon leur contenu en éléments logiques, les slices sont divisés en deux types :

Slices M : chaque slice M de la série 7 de Xilinx contient :

- Quatre générateurs de fonctions logiques LUT.
- Huit éléments de mémorisation de type Flip Flop.
- Des multiplexeurs
- Une logique de calcul de la retenu

Slices L : en plus des éléments contenus dans les Slices M, les slices L contiennent des mémoires distribuées et des registres à décalage.

Un bloc CLB peut contenir soit deux SLICEL soit une combinaison de SLICEL et SLICEM.

Suivant le besoin en ressources, le concepteur peut choisir entre plusieurs sous famille de la série Virtex-7. Les différentes versions se distinguent par le nombre de ressources disponibles, le niveau de performance, la consommation de l'énergie et bien évidement le prix. Le tableau II.1résume les ressources disponibles pour les différentes versions de virtex-7

Tableau II.1 : Ressources disponibles pour les différentes versions de virtex-7

Device	Logic Cells	Configurable Logic Blocks (CLBs)		DSP Slices(2)	Block RAM Blocks(3)		
		Slices(1)	Max Distributed RAM (Kb)		18 Kb	36 Kb	Max (Kb)
XC7V585T	582,720	91,050	6,938	1,260	1,590	795	28,620
XC7V2000T	1,954,560	305,400	21,550	2,160	2,584	1,292	46,512
XC7VX330T	326,400	51,000	4,388	1,120	1,500	750	27,000
XC7VX415T	412,160	64,400	6,525	2,160	1,760	880	31,680
XC7VX485T	485,760	75,900	8,175	2,800	2,060	1,030	37,080
XC7VX550T	554,240	86,600	8,725	2,880	2,360	1,180	42,480
XC7VX690T	693,120	108,300	10,888	3,600	2,940	1,470	52,920
XC7VX980T	979,200	153,000	13,838	3,600	3,000	1,500	54,000
XC7VX1140T	1,139,200	178,000	17,700	3,360	3,760	1,880	67,680
XC7VH580T	580,480	90,700	8,850	1,680	1,880	940	33,840
XC7VH870T	876,160	136,900	13,275	2,520	2,820	1,410	50,760

III.3.2 La famille Zynq-7000 :

La famille « Zynq-7000 » englobe des circuits de type systèmes sur puce (ou system on chip SoC) développée par la société AMD-Xilinx. Un circuit Zynq combine entre un circuit logique programmable (PL : Programmable Logic) et un processeur ou système de traitement (PS : processing system) qui pilote la partie PL. Les liens entre la partie PL et la partie PS sont établis à l'aide d'une connexion standard de l'industrie connue par l'abréviation AXI (Advanced eXtensible Interface) comme le montre la figure II.9. Cette combinaison permet une grande flexibilité dans la conception de systèmes embarqués, car elle offre à la fois la puissance de calcul des processeurs ARM et la capacité de traitement parallèle des FPGA. Les circuits PYNQ sont particulièrement populaires dans les applications nécessitant un traitement de signal en temps réel, une faible latence ou une grande flexibilité dans la conception du système.

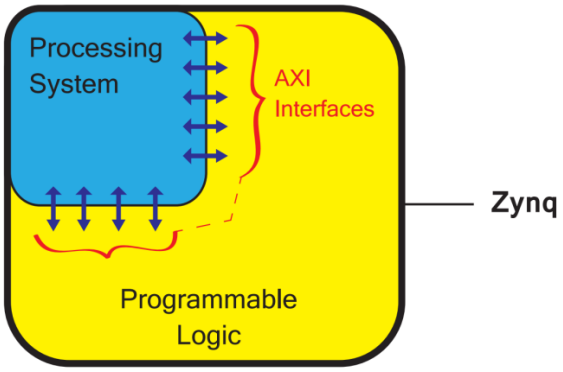


Figure II.9 schéma simplifié d'un circuit ZYNQ

La famille Zynq 7000 contient six circuits : Le Z-7030, Z-7045, 7-100, Z-7010, Z-7015 et Z7020. Les trois premières utilisent comme PL le Kintex-7 de Xilinx, ces derniers sont destinés aux applications qui exigent des performances et un débit d'entrées/sorties relativement élevées. Les autres circuits contiennent l'Artix-7 comme PL, leur utilisation apparaît dans des applications limitées en termes de consommation électrique et de coût.

III.3.2.a Système de traitement (processing system PS) :

Tous les circuits Zynq ont la même architecture de base et contiennent tous, comme base du système de traitement, un processeur ARM Cortex-A9 double cœur.

Il est important de noter que le système de traitement Zynq englobe non seulement le processeur ARM, mais un ensemble de ressources de traitement associées formant une unité de traitement d'application (Application Processing Unit APU), ainsi que d'autres interfaces

périphériques, une mémoire cache, des interfaces mémoire, une interconnexion et des circuits de génération d'horloge. Un schéma fonctionnel montrant l'architecture du PS est présenté dans la figure II.10.

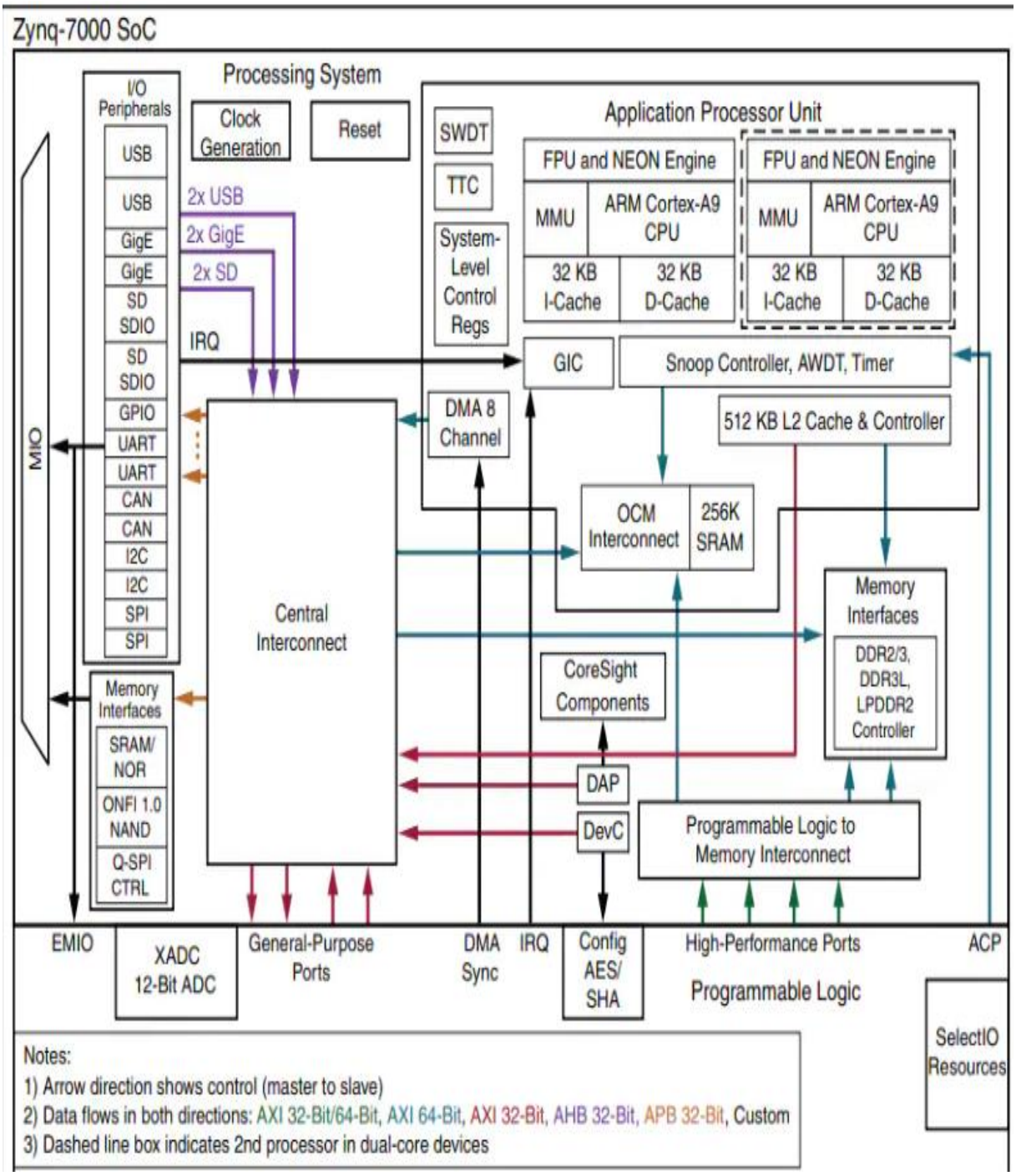


Figure II.10 : architecture de la partie PS d'un circuit ZYNQ [20].

- **Unité de processeur d'application :**APU (Application Processing Unit) est un ensemble de ressources de traitement associé. Il est constitué principalement de deux cœurs de processeur ARM (Advanced RISC Machine) chacun associé à plusieurs unités de traitement, notamment [20].
- **Unité NEON™ :** également appelé Advanced SIMD (Single Instruction Multiple Data) Ou encore MPE (Media Processing Engine), littéralement « moteur de calcul de médias ».
  C'est une unité de calcul de type SIMD, apporte une accélération majeure des algorithmesDSP (Digital Signal Processing) et multimédia au processeur ARM principal.
- **Unité FPU (Floating Point Unit) :** est un coprocesseur arithmétique dédié aux opérations sur les nombres réels en virgule flottante.
- **MMU (Memory Management Unit) :** c'est une unité de gestion de mémoire permettant la gestion efficace de transfert de données de et vers les mémoires.
- **Mémoire cache L1 de niveau 1** de 32 KB pour les données et les instructions.
- **Mémoire cache L2 de niveau 2** de 512 KB pour les données et les instructions.
- **Mémoire SRAM (Static Random Access Memory)** de 256KB on OCM (On Chip Memory).
- **SNOOP (Snoop Control Unit (SCU))** forme une passerelle entre les cœurs ARM et la mémoire cache pour maintenir la cohérence de L1 et L2.
- **Contrôleur DMA (Direct Memory Access)** quatre canaux pour PS (copie de mémoirevers/depuis n'importe quelle mémoire du système) et quatre pour PL (ProgrammableLogic) « mémoire vers PL, PL vers mémoire ».
- **GIC (General interrupt controller)** prend les interruptions des périphériques, les hiérarchise et les transmet au cœur de processeur.
- **SWD (Serial Wire Debug)** processeur implémente le débogage série matériel pris en charge le port de débogage JTAG.
- **TTC (Triple Timer Counter)** fournit une ressource de synchronisation beaucoup plus flexible. Il peut être utilisé comme minuteries ou pour produire des formes d'onde sur les broches EMIO (Extended Multiplexed Input/Output) ou MOI (Multiplexed Input/Output)du Zynq SoC. Sa source d'horloge sélectionnée via le registre de contrôle d'horloge situé en dessous.
- **Interfaces mémoires :** Les interfaces mémoires intègrent plusieurs technologies mémoire aussi bien avec la partie PL qu'avec le monde extérieur

- **Périphériques d'entrées sorties :**La communication entre le système de traitement (PS) et les interfaces des périphériques externes est assurée par un bus d'entrées sorties multiplexé MIO.

III.3.2.b partie PL (Programmable Logic) :

La partie PL se compose principalement d'un tissu logique de semblable au circuits FPGA à usage général comme la montre la figure II.11. Elle est composée d'une matrice de blocs logiques configurables CLB entourés de blocs d'entrée sortie programmable IOB pour l'interfaçage. L'ensemble est relié par un réseau d'interconnexions programmable [20].

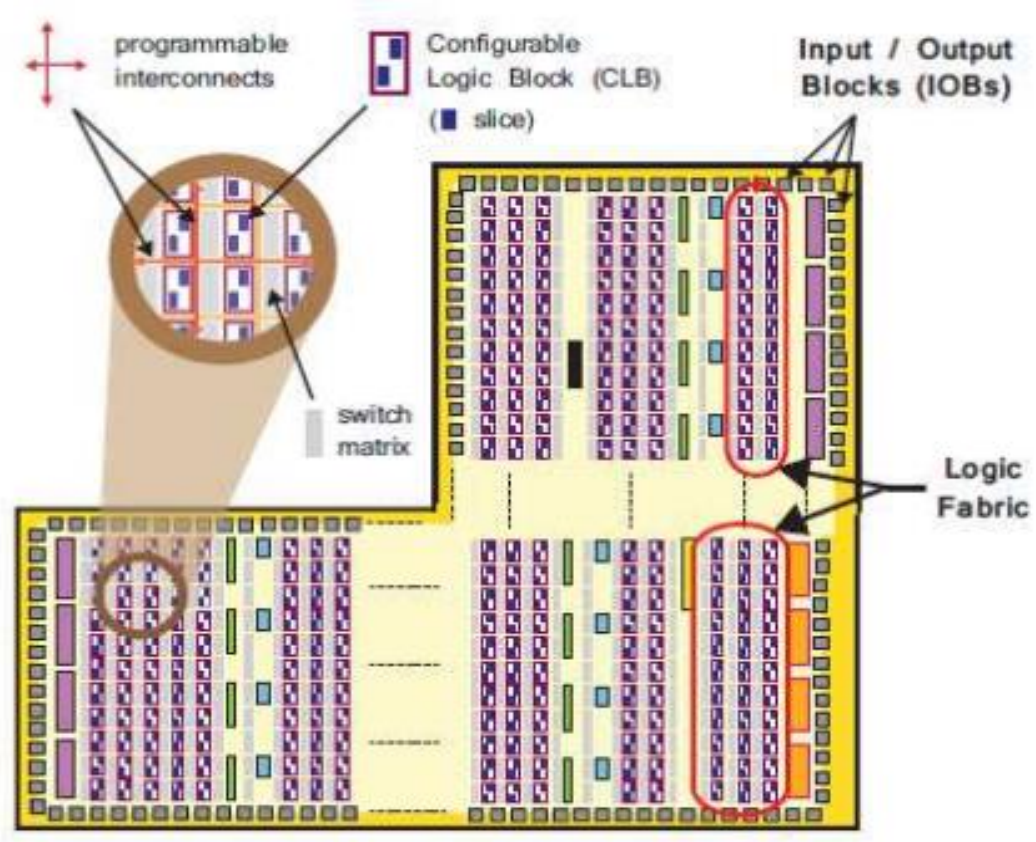


Figure II .11: architecture de la partie PL des circuits ZYNQ [22]

Les caractéristiques de la PL peuvent être résumées comme suit :

- **CLB (Bloc logique configurable) :** Les CLB sont des groupements réguliers d'éléments logiques disposés dans un réseau bidimensionnel sur la PL et connectés à d'autres ressources similaires via des interconnexions programmables. Chaque CLB est

positionné à côté d'une matrice de commutation et contient deux tranches logiques (slice) selon la structure du circuit II.12.

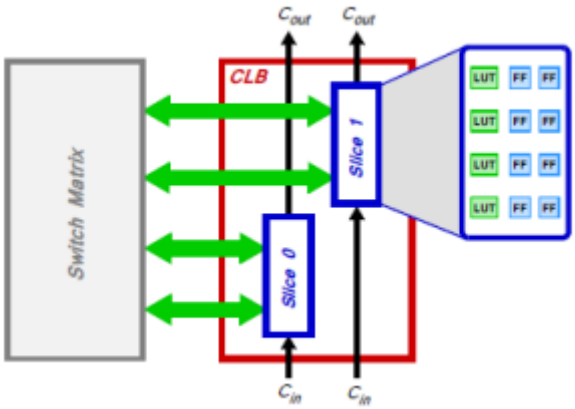


Figure II .12 : Composition d'un bloc logique configurable (CLB) de la famille Zynq-7000.

Slice : c'est une sous-unité dans le CLB, qui contient des ressources pour la mise en œuvre de circuits logiques combinatoires et séquentiels. Les tranches Zynq sont composées de 4 tables de recherche, de 8 flip-flops et d'autres logiques.

Table de recherche (LUT) : c'est une ressource flexible capable de mettre en œuvre soit :

(i) Une fonction logique allant jusqu'à six entrées.

(ii) Une petite mémoire ROM.

(iii) Une petite mémoire d'accès aléatoire (RAM).

(iv) Un registre à décalage horaire. Les LUT peuvent être combinés pour former des fonctions logiques, des mémoires ou des registres à décalage plus importants, au besoin.

Flip-flop (FF) : Un élément de circuit séquentiel implémentant un registre à 1 bit, avec fonctionnalité de réinitialisation.

Switch Matrix : Une matrice de commutation se trouve à côté de chaque CLB, et fournit une installation de routage flexible pour établir des connexions entre les éléments d'un CLB ou d'un CLB à d'autres ressources sur le PL.

- **Blocs d'entrée / sortie (IOB) :**Les IOB sont des ressources qui fournissent une interface entre les ressources logiques PL et les « pads » du périphérique physique utilisés pour se connecter aux circuits externes. Chaque IOB peut gérer un signal d'entrée ou de sortie 1 bit. Les IOB sont généralement situés autour du périmètre de l'appareil.

En plus du tissu général, il existe deux composants spéciaux : les blocs RAM pour les besoins en mémoire dense et les tranches DSP48E1 pour l'arithmétique à haute vitesse.

- **Blocs RAMs :**Ces blocs RAMs peuvent être configurés en mémoire RAM, en mémoire ROM ou en mémoire FIFO. Chaque bloc mémoire est capable de stocker jusqu'à 36Kb.Ils peuvent être configurés comme une mémoire d'une taille de 36 Kb ou comme deux mémoires indépendantes de taille 18 Kb. Les Blocs DSPs.
- **Les blocs DSP48E :** Sont des ressources matérielles à haute performance, capables d'accélérer les opérations arithmétiques d'additions/soustractions, de multiplications, d'additions/soustractions suivies par des multiplications, ou de multiplications suivies par des additions/soustractions.

III.3.2.c Interconnexion entre PS et PL :

La communication d'instructions et de données entre le PS et le PL est réalisée à l'aide d'interfaces matérielles dédiées comme l'illustre la figure II.13.

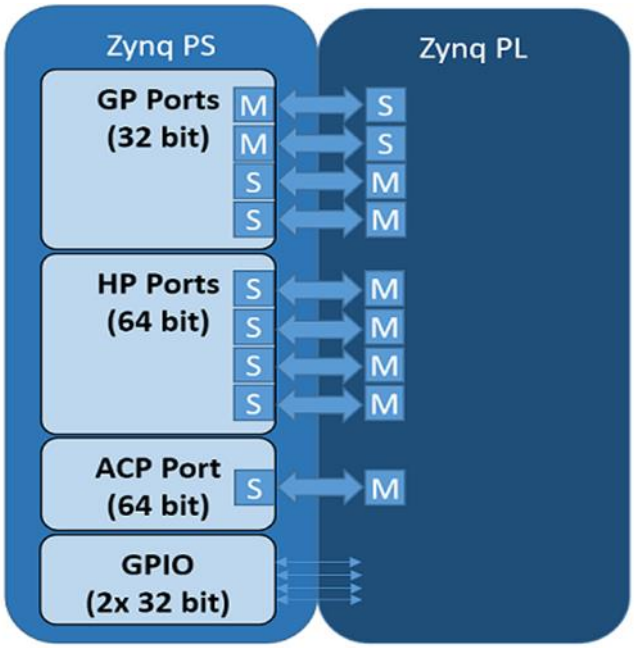


Figure II.13 Interface entre PS et PL

Il existe de nombreux types d'interfaces ; dont le plus courant est le bus AXI de la norme ouverte Arm AMBA.

Il existe trois types de formats pour le bus AXI, chacun avec un protocole de bus différent, qui sont résumés ci-dessous [18] :

• **AXI4 :** Ce protocole est destiné aux connexions nécessitant des liens mappés en mémoire entre les éléments de traitement et les blocs IP. Il est adapté au transfert de grandes quantités d'informations depuis /vers la mémoire principale.

• **AXI-Lite :** Ce protocole est généralement utilisé pour une communication à faible bande passante avec des registres de contrôle de blocs IP et des éléments de traitement.

• **AXI4-Stream :** Ce protocole prend en charge le streaming de données point à point afin de transférer des données à haute vitesse. Il est particulièrement utile pour le traitement du signal dans les applications vidéo, de communication et de mise en réseau.

III.3.3 cartes de développementPYNQ-Z1 :

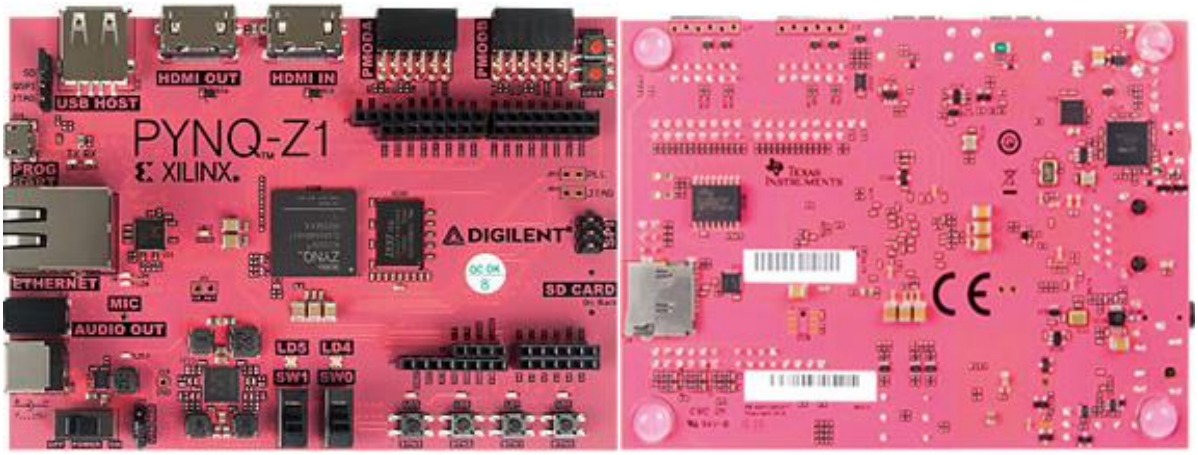
Le mot PYNQ est une concaténation du début du mot Python et de la fin du mot ZYNQ qui signifie *Python Productivity for Zynq.* PYNQ est un projet open source de Xilinx qui rend la conception de systèmes embarqués avec les SoCs Zynq de Xilinx facile. En utilisant Python et des bibliothèques, les concepteurs peuvent exploiter les avantages de la logique programmable et des microprocesseurs dans Zynq pour construire des systèmes embarqués plus performants et passionnants. Les utilisateurs de PYNQ peuvent créer des applications embarquées haute performance avec une exécution matérielle parallèle, un traitement vidéo à haut débit d'images, des algorithmes accélérés par matériel, un traitement de signal en temps réel, une E/S à haut débit et un contrôle à faible latence.

La carte de développement PYNQ-Z1 est une carte électronique à base d'un circuit SoC de type ZYNQ XC7Z020-1CLG400C conçu pour être utilisé suivant le cadre de travail de l'environnement PYNQ.

Les composants de base de cette carte sont :

1. **Zynq-7000 SoC (System on Chip) :** Le cœur de la carte PYNQ-Z1 est le SoC ZYNQ XC7Z020-1CLG400C de Xilinx, qui intègre à la fois un processeur ARM Cortex-A9 dual-core (pour le traitement logiciel) et une matrice FPGA programmable (pour le traitement matériel).
2. **Mémoire** : La carte est équipée de mémoire DDR3 de taille 512MB DDR3, d'une mémoire Flash de taille 16MB en plus d'un lecteur de mémoire externe de type microSD.

3. **Connecteurs d'extension :** La PYNQ-Z1 dispose de connecteurs d'extension compatibles avec le format Arduino et Pmod, permettant de connecter une variété de périphériques et d'extensions.
4. **Connectivité :** La carte comprend des interfaces de connectivité telles que des ports USB, Ethernet, HDMI, et des connecteurs pour UART, JTAG, etc.
5. **Boutons et LED :** Des boutons et des LED sont présents sur la carte pour permettre une interaction simple avec le système.



A B

Figure II.14. Face avant (A) et face arrière (B) de la carte PYNQ-Z1.

III.3.4 Concept général du PYNQ :

Le framework de développement d'une application sur circuit PYNQ est constitué par trois couches montrées sur la figure II.15.

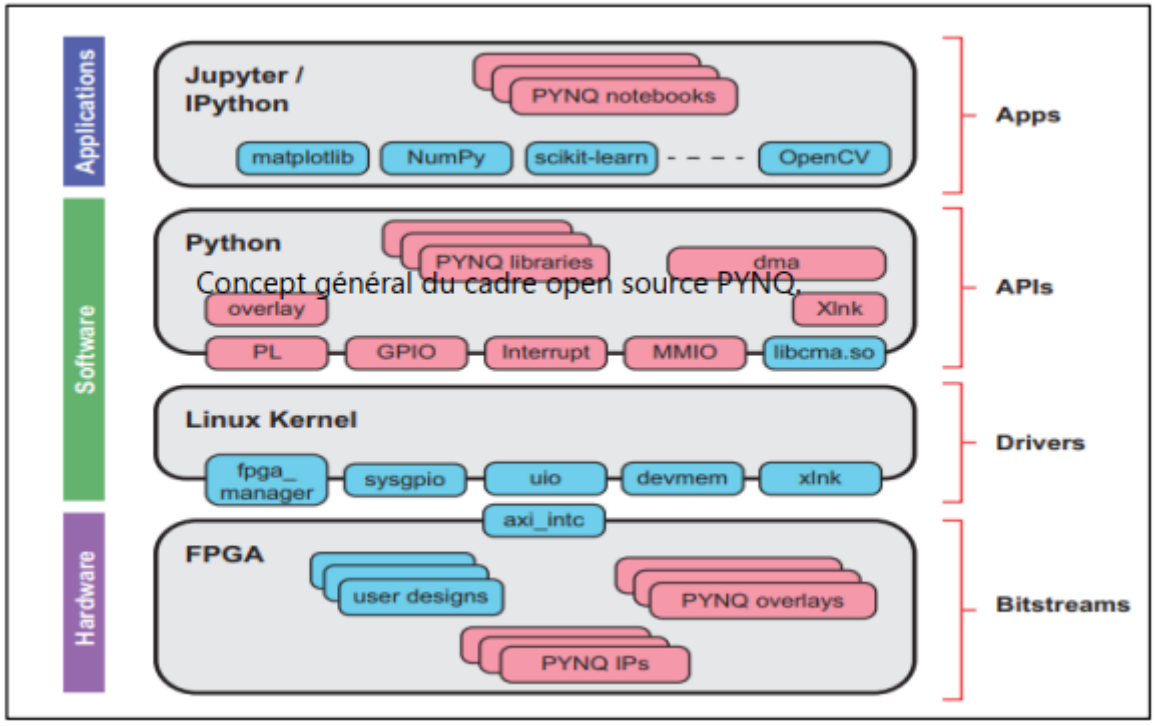
III.3.4.à Couche supérieure (Applications) :

La couche supérieure du PYNQ est facilitée par un ou plusieurs notebooks Jupyter.PYNQ intègre l'infrastructure de bloc-notes Jupyter pour exécuter directement le noyau sur le processeur ARM du périphérique Zynq. Dans un notebook Jupyter PYNQ, le développeur doit créer ses propres fonctionnalités personnalisées en écrivant son propre code Python et en le réutilisant de manière sélective via de nombreuses bibliothèques Python open source disponibles, y compris des packages tels que NumPy, Matplotlib, OpenCV.

III.3.4.b Couches intermédiaires (Software) :

Les couches intermédiaires du PYNQ se composent du logiciel Python, du système d'exploitation et des pilotes logiciels de bas niveau. Dans la couche intermédiaire supérieure, le framework PYNQ comprend des bibliothèques Python et des API pour interagir avec divers éléments des systèmes basés sur Zynq. Par exemple, il existe des API Python pour télécharger des superpositions (fichiers bitstream) sur le PL, pour communiquer avec les ressources GPIO et gérer les interruptions générées dans le PL. Il existe également des API Python pour les transferts mappés en mémoire (MMIO) et les transferts DMA.[21]

Dans la couche intermédiaire inférieure, le framework PYNQ comprend un système d'exploitation basé sur Linux, des chargeurs de démarrage pour lancer le démarrage du système et un serveur Web pour héberger les ordinateurs portables Jupyter. Elle comprend également un ensemble de pilotes pour interagir avec les éléments du système matériel Zynq.



Concept général du cadre open source PYNQ.

Figure II .15 : Concept général du cadre open source PYNQ.

III.3.4.c Couche inférieure (Hardware) :

La couche inférieure du PYNQ représente une conception de système matériel, sur lequel il sera créé sur Vivado à l'aide d'IP Integrator et des outils de conception associés, puis générée dans un fichier bitstream (*.bit). Le fichier bitstream est transféré sur la carte mémoire insérée dans la carte cible. Le processus de programmation du système matériel sur le PL peut alors être lancé directement à partir d'un bloc-notes Jupyter susmentionné sur la couche supérieure de tel concept (fonctionnant sur le PS).[21]

En tenant compte de tous ces composants réutilisables, on peut définir le terme ‘bibliothèques matérielles’. Il s'agit d'un terme générique faisant référence à l'ensemble de IP et de superpositions disponibles dans le cadre du PYNQ pour une réutilisation flexible. Les détails des conceptions matérielles et leurs fonctionnalités peuvent être résumés en Python via une interface de programmation d'application (API) ou les bibliothèques matérielles peuvent être considérées comme similaires aux bibliothèques logicielles.

IV. Conclusion

Les réseaux de neurones de type CNN impliquent un calcul de convolution sur un nombre très élevé de matrices ce qui nécessite des circuits de calcul très puissants et un espace mémoire important. Les circuits FPGA et les SoC de type ZYNQ offrent une solution pour l’implémentation efficace des réseaux CNN sur une plateforme à la fois reconfigurable et reprogrammable. Dans ce contexte, ce chapitre a été consacréà une étude approfondie des circuits logiques programmables FPGA et des circuits ZYNQ, mettant en lumière leur architecture et leur fonctionnement. Nous avons exploré l'évolution des circuits électroniques, l'émergence des FPGAs et leur importance dans la conception de systèmes numériques. Les FPGAs et les PYNQ se sont révélés être des outils puissants et flexibles, permettant la réalisation de circuits complexes et la validation de conceptions avant leur production. La famille Zynq et le cadre open source Pynq ont été présentés comme des innovations majeures, facilitant l'utilisation des FPGAs dans le domaine des systèmes embarqués et ouvrant de nouvelles perspectives pour les développeurs.

Il existe plusieurs cartes de développement utilisant le framework PYNQ comme PYNQ-Z1, PYNQ-Z2, PYNQ-ZU2 et PYNQ-ZU3EG. Toutes ces cartes offrent un environnement de développement complet combinant la puissance des processeurs ARM avec la flexibilité des FPGA pour une large gamme d'applications, y compris l'IA, le traitement de signal, la vision par ordinateur.

Chapitre 3 :

Implémentations, optimisation et résultats

I. Introduction

La classification d'images joue un rôle crucial dans la collecte, l'analyse et l'utilisation des données visuelles dans plusieurs domaines de notre vie quotidienne. La technique de base la plus utilisée du concept du Deep Learning dans le domaine de classification est le réseau de neurones convolutifs (CNN).

Dans ce chapitre nous allons représenter la partie pratique de notre travail en commençant par la programmation d'un réseau CNN pour la classification des digits manuscrits. Ensuite, nous expliquons les différentes étapes de conception du réseau de classification à l'aide du logiciel System Generator version 2018. Finalement nous allons présenter les résultats d'implémentation de réseaux CNN sur la carte PYNQ-Z1 en utilisant l'outil d'implémentation Vivado 2018.

II logiciels utilisés

II.1-Matlab

Le nom Matlab vient d'abréviation de « Matrix (Mat) LaBoratory (Lab), c'est une plateforme de programmation développé par MathWorks et conçue spécifiquement pour les ingénieurs et les scientifiques afin d'analyser et de concevoir des systèmes et des produits qui transforment notre monde. L'outil Matlab permet de :

- Analyse des données.
- Développement des algorithmes.
- Création des modèles et des applications.
- Manipulations matricielles.
- Traçage des fonctions et des données.
- Mise en œuvre d'algorithme.
- Création d'interfaces d'utilisateurs.
- Création d'interfaçage avec des programmes écrits dans d'autres langages [22].

II.2-Simulink

Simulink est un outil de modélisation et de simulation qui permet de créer des modèles de systèmes complexes en utilisant des schémas-blocs. Il est souvent utilisé pour la modélisation de systèmes physiques, la simulation de comportements dynamiques, et la vérification de systèmes de contrôle. Les deux outils Matlab et Simulink sont souvent utilisés ensemble pour concevoir et simuler des systèmes complexes, notamment dans les domaines de l'électronique, de l'automatisation industrielle, et de la mécanique

II.3-Python

Python est un langage de programmation à haut niveau. Sa philosophie de conception met l'accent sur la lisibilité du code avec l'utilisation d'une indentation significative. Python est dynamiquement typé et recueilli. Il prend en charge plusieurs paradigmes de programmation, y compris la programmation structurée (surtout procédurale), orientée objet et fonctionnelle. Il est souvent décrit comme un langage comprenant des paquages en raison de sa vaste bibliothèque standard. Python est constamment classé comme l'un des langages de programmation les plus populaires.



Les avantages du langage Python sont nombreux, ce qui explique sa popularité croissante parmi les développeurs. Parmi ces avantages on peut citer :

- **Facilité d'apprentissage et de lecture** : Python est réputé pour sa syntaxe simple et facile à comprendre, ce qui en fait un excellent choix pour les débutants en programmation.
- **Polyvalence** : Python peut être utilisé dans une grande variété de domaines, y compris le développement web, l'analyse de données, l'apprentissage automatique, l'automatisation de tâches, et bien plus encore.
- **Une grande et active communauté** : Python bénéficie d'une grande communauté d'utilisateurs et de développeurs dynamique et engagée, qui contribue à son évolution constante et à la création d'une multitude de bibliothèques et de frameworks.
- **Vaste collection de bibliothèques** : Python dispose d'une vaste collection de bibliothèques open source qui facilitent le développement d'applications en fournissant des solutions prêtes à l'emploi pour de nombreux problèmes courants.
- **Grande portabilité** : Python est pris en charge par de nombreuses plates-formes, y compris Windows, macOS et Linux, ce qui le rend facilement portable d'un système à l'autre.

➢ **Orientation objet :** Python prend en charge la programmation orientée objet, ce qui permet une structuration modulaire et une réutilisation facile du code.

Python est l'un des langages les plus populaires pour le développement d'applications de deep learning, principalement en raison de :

➢ **Bibliothèques spécialisées** : Python dispose de plusieurs bibliothèques spécialisées dans le deep learning, notamment TensorFlow, PyTorch et Keras. Ces bibliothèques fournissent des outils puissants pour la construction, l'entraînement et le déploiement de réseaux neuronaux.

➢ **Bibliothèques de traitement de données** : Python propose également de nombreuses bibliothèques pour le traitement et la manipulation des données, telles que NumPy, Pandas et SciPy, ce qui facilite la préparation des données avant de les utiliser dans des modèles de deep learning.

➢ **Intégration avec d'autres technologies** : Python peut être facilement intégré avec d'autres technologies couramment utilisées dans le domaine du deep learning, telles que les bases de données, les frameworks web et les outils de visualisation. Python est un langage de programmation interprété, multiparadigme et multiplateforme. Il est conçu pour être lisible et favorise la programmation impérative structurée, fonctionnelle et orientée objet. Python est dynamiquement typé et utilise une gestion automatique de la mémoire par ramasse-miettes. Il est souvent décrit comme un langage "batteries included" en raison de sa bibliothèque standard exhaustive. [23]

II.4- Xilinx System generator (XSG)

Xilinx System Generator est un outil de conception logicielle qui permet aux concepteurs de systèmes embarqués et de traitement de signal de concevoir, simuler et implémenter des systèmes complexes et performants sur des circuits FPGA (Field-Programmable Gate Array) de Xilinx. Ce logiciel agit comme un plugin ou une extension pour Simulink. Pour utiliser Xilinx System Generator, les utilisateurs doivent créer un sous-système contenant les blocs Xilinx et les blocs Simulink nécessaires, puis configurer les paramètres de simulation et de génération de code HDL. La figure III.1 donne un aperçu d'une fenêtre de développement de l'outil XSG.

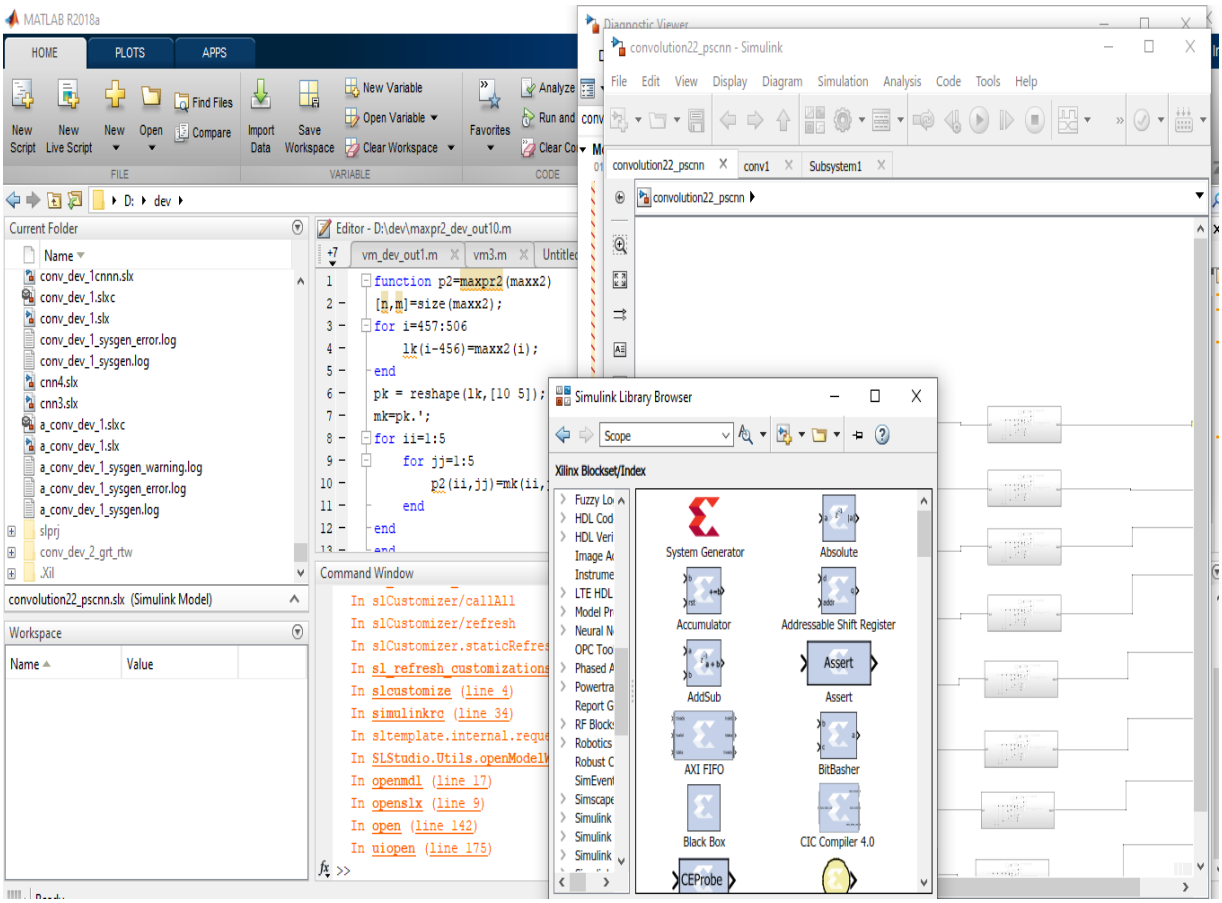


Figure III.1 Fenêtre de développement de l'outil XSG.

Les fonctionnalités principales de XSG sont les suivantes :

- Conception de designs de FPGA : Xilinx System Generator permet aux utilisateurs de créer des designs de FPGA en utilisant des blocs graphiques et de les valider par simulation dans Simulink.
- Génération de code HDL : Le code généré peut être synthétisé en un FPGA Xilinx.
- Intégration avec Vivado : Les designs générés peuvent être facilement importés dans un projet Vivado pour une implémentation finale.
- Support pour les FPGA Xilinx : Xilinx System Generator est spécifiquement conçu pour les FPGA Xilinx et génère des designs optimisés pour ces appareils. [24]

II.5 logiciels vivado



Vivado Design Suite est une suite logicielle produite par la société technologique américaine Xilinx et qui offre plusieurs façons d'accomplir les tâches impliquées dans la conception, la mise en œuvre et la vérification des appareils Xilinx. À cette fin, Xilinx Vivado vise à synthétiser et analyser des conceptions en langage de description matérielle (HDL) afin de fournir des modules supplémentaires pour le développement des systèmes sur puce et des outils de synthèse de haut niveau. Les fonctionnalités d'analyse et la vérification de conception incluent la programmation, le débogage, la simulation logique, la planification des E/S, la visualisation de la logique de conception, la modification des résultats de mise en œuvre

III. Application de reconnaissance de digits manuscrit

III.1 Explication de l'application

La reconnaissance de chiffres manuscrits est une tâche essentielle dans le domaine de l'intelligence artificielle et de la vision par ordinateur. Elle a des applications variées, allant de la lecture automatique des chèques bancaires à la reconnaissance d'écriture sur les formulaires.

Dans son cadre le plus général, l'application fonctionne en utilisant la vision par ordinateur et le traitement d'image pour d'abord localiser et isoler les caractères individuels dans une image. Une fois les chiffres isolés, ils sont prétraités pour améliorer la qualité de l'image en vue d'une meilleure reconnaissance. Cela peut inclure le redimensionnement des images, la normalisation des contrastes, la binarisation (conversion en images en noir et blanc) et le centrage des chiffres. Après le prétraitement, les images sont alimentées dans un réseau de neurones convolutifs (CNN), un type d'intelligence artificielle qui s'inspire du système visuel humain et est particulièrement efficace pour la reconnaissance d'image. Le réseau va "apprendre" à reconnaître les chiffres manuscrits à travers un processus d'apprentissage supervisé sur une grande base de données de chiffres manuscrits étiquetés, comme MNIST.

III.2 La base de données MNIST

La base de données MNISTpour « Modified ou Mixed National Institute of Standards and Technology »est une collection élargie de la base de données NIST originale, conçue pour servir de point de référence pour tester et évaluer les techniques de reconnaissance d'images. Elle a été rendue populaire par Yann LeCun et Corinna Cortes. MNIST est une collection

incontournable de chiffres manuscrits utilisée dans le monde de l'apprentissage automatique et de la vision par ordinateur pour le développement et l'évaluation de modèles de reconnaissance d'images. Elle comprend 70 000 images en niveaux de gris, divisées en un ensemble d'apprentissage de 60 000 images et un ensemble de test de 10 000 images, chacune normalisée à une taille de 28x28 pixels. Les images proviennent de deux sources différentes, offrant une variété de styles d'écriture qui contribue à la robustesse des modèles entraînés avec cette base. Bien que considérée comme un point de départ basique par rapport à des bases de données plus modernes et complexes, MNIST reste largement appréciée pour son équilibre entre simplicité et défi, servant de benchmark initial pour les innovations en reconnaissance d'images. La figure III.2 donne des échantillons choisis aléatoirement de la base MNIST.

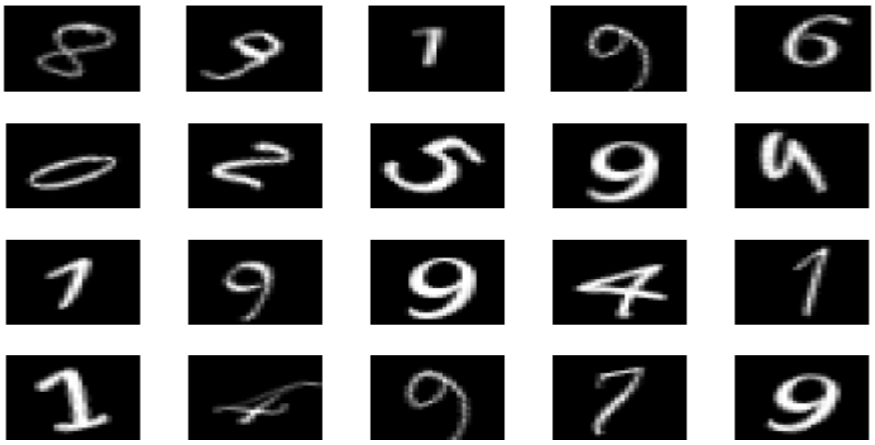


Figure III.2 : Exemple d'mage de chiffres de la base de données MNIST.

III.3 Architecture du réseau CNN

Pour notre application, nous avons choisi une architecture de réseau neuronal convolutif (CNN) composée de trois couches de convolution, trois activations de type Relu, deux couches de maxpooling, un couche FC et finalement une activation Sofmax comme la montre la figure III.3.

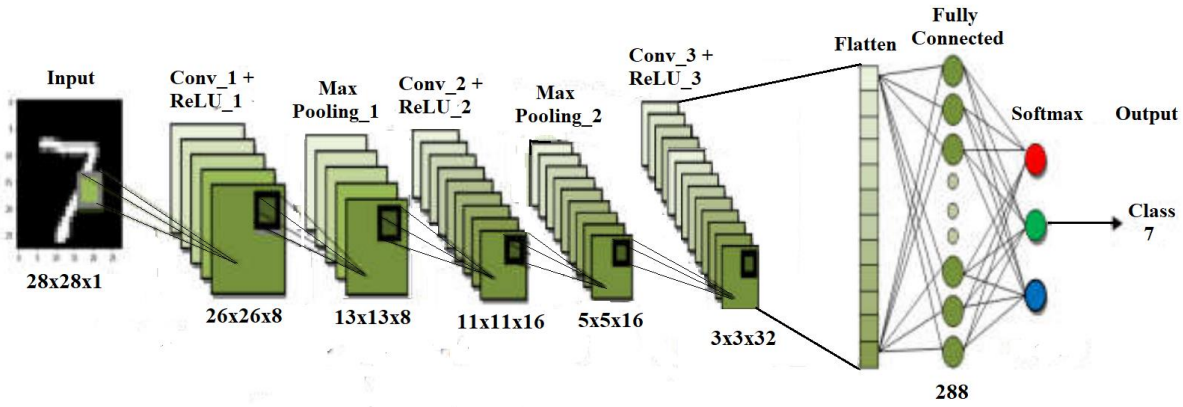


Figure III.3 Architectures du réseau CNN choisie pour la reconnaissance de digits manuscrits.

Le tableau suivant résume les différents paramètres de chaque couche du réseau :

Tableau III.1 caractéristiques des différentes couches du réseau choisi.

Couche	**Nombre d'entrées et dimensions**	**Nombre de sorties et dimensions**	**Dimension du kernel**	**Nombre de paramètres**
Convolution1	(28,28,1)	(26,26,8)	(3 ,3)	80
Relu 1	(26,26,8)	(26,26,8)	N/A	0
Maxpooling 1	(26,26,8)	(13,13,8)	(2,2)	0
Convolution2	(13,13,8)	(11,11,16)	(3 ,3)	1168
Relu 2	(11,11,16)	(11,11,16)	N/A	0
Maxpooling 2	(11,11,16)	(5,5,16)	(2,2)	0
Convolution3	(5,5,16)	(3,3,16)	(3 ,3)	4640
Relu 3	(3,3,32)	(3,3,16)	N/A	0
FC	288	10	N/A	2890
Softmax	10	10	N/A	0

III.4 Programmation, apprentissage et évaluation du réseau

III.4.1 En utilisant matlab

Cette étape est réalisée en utilisant la version 2018 du logiciel *Matlab* qui contient un toolbox très riche en fonctions spécialement conçu pour les réseaux CNN. Le programme peut être divisé en quatre partis qui sont :

- Préparation des données d'apprentissage et de test.
- Détermination des différentes couches du réseau.
- Apprentissage du réseau.
- Evaluation du réseau.

a) Préparation des données d'apprentissage et de test :

On commence par la génération d'un datastore des images de la totalité de la base de données.

L'utilisation de Datastore permet de gérer une collection de fichiers d'images, où chaque image individuelle est chargée dans la mémoire, mais la collection entière d'images n'est pas chargée nécessairement dans la mémoire en même temps. La génération du datastore est assurée par la fonction *imageDatastore* de *Matlab*.

Le Data store est ensuite divisé en deux parties, la première partie définie les images utilisées pour l'apprentissage et la deuxième partie définie les images de test ou d'évaluation. Cette opération est réalisée à l'aide de la fonction *splitEachLabel.*

b) **Détermination des différentes couches du réseau :**

Pour déterminer l'architecture d'un réseau CNN, il faut spécifier les différentes couches avec leurs caractéristiques. En *Matlab* la spécification des couches commence par le mot clé « *layers*= » puis en spécifie les différentes couches et leurs caractéristiques entre deux crochets « [ ] ». Le tableau III.2 résume l'ensemble des fonctions *Matlab* utilisées pour la détermination des différentes couches d'un réseau CNN.

Tableau III.2* : *Explication des fonctions Matlab utilisées pour la détermination des différentes couches du réseau CNN.

Nom de la fonction	**Type de la couche**	**Les paramètres de base**
ImageInputLayer	Couche d'entrée du réseau	InputSize : les dimensions de l'image d'entrée. Normalization : type de normalisation sur l'image s'il existe. Name : le nom de la couche d'entrée. Exemple : imageInputLayer ([100 100 1], 'Name', 'Input','Normalization', 'none')
Convolution2dLayer	Convolution deux dimensions	FilterSize : Hauteur et largeur des filtres, spécifiées sous forme de vecteur [h w], si le filtre est carré on spécifie un seul entier. NumFilters : le nombre de groupes de filtres. Ce paramètre détermine le nombre de canaux dans la sortie de la couche. Stride : c'est le pas de déplacement oriental et vertical du filtre. Name : le nom de la couche Exemple : Convolution2dLayer (3,8,'Name', 'Conv_1')
BatchNormalizationLayer	Normalisation par lots	Name : le nom de la couche Exemple: batchNormalizationLayer ('Name', 'BN_1')
ReluLayer	Fonction d'activation de type ReLu	Name : le nom de la couche Exemple : ReluLayer('Name','Relu_1')
MaxPooling2dLayer	Fonction de sous échantillonna	PoolSize : Dimensions des régions de pooling sous la forme d'un vecteur de deux entiers positifs [h w], ou un scalaire si la région est carrée.

	ge (pooling) de type Max	Stride : pas de déplacement horizontal et vertical de la région de pooling. Il est spécifié sous la forme d'un vecteur de deux entiers positifs [a b] ou scalaire si le pas est le même horizontalement et verticalement. Name : le nom qu'on donne à la couche Exemple: maxPooling2dLayer(2,'Stride',2,'Name','MaxPool_1')
FullyConnectedLayer	Couche entièrement connectée	OutputSize : un entier spécifiant la taille de la sortie de la couche. Name : chaine de caractères spécifiant le nom de la couche. Exemple : FullyConnectedLayer (5, 'Name', 'FC')
SoftmaxLayer	Fonction d'activation de type soft max utilisée généralement après la dernière couche FC	Name : chaine de caractères spécifiant le nom de la couche. Exemple : SoftmaxLayer('Name','SoftMax');
ClassificationLayer	Couche permettant d'attribuer une classe (classification)	Name : chaine de caractères spécifiant le nom de la couche. Exemple : ClassificationLayer ('Name', 'Output Classification') ;

c) **Apprentissage du réseau :**

Avons de lancer la tâche d'apprentissage du réseau il faut d'abord déterminer les options d'apprentissage. Pour déterminer les options d'apprentissage sous Matlab on utilise la fonction trainingOptions.

Après le choix des options d'apprentissage, on lance l'opération d'apprentissage comme suivant : net=trainNetwork (TrainImages, layers, options);

Après l'étape d'apprentissage, on obtient un réseau entrainé et près à être utilisée pour la classification. La figures III.4 et 3.4 montrent les graphes obtenus et affichés durant la phase d'apprentissage du réseau CNN. Le premier graphe représente l'évolution de l'accuracy en fonction de nombre d'itérations et le deuxième graphe donne l'évolution de la fonction perte (loss) en fonction du nombre d'itération. On remarque que l'accuracy de validation atteint la valeur de 97.71%.

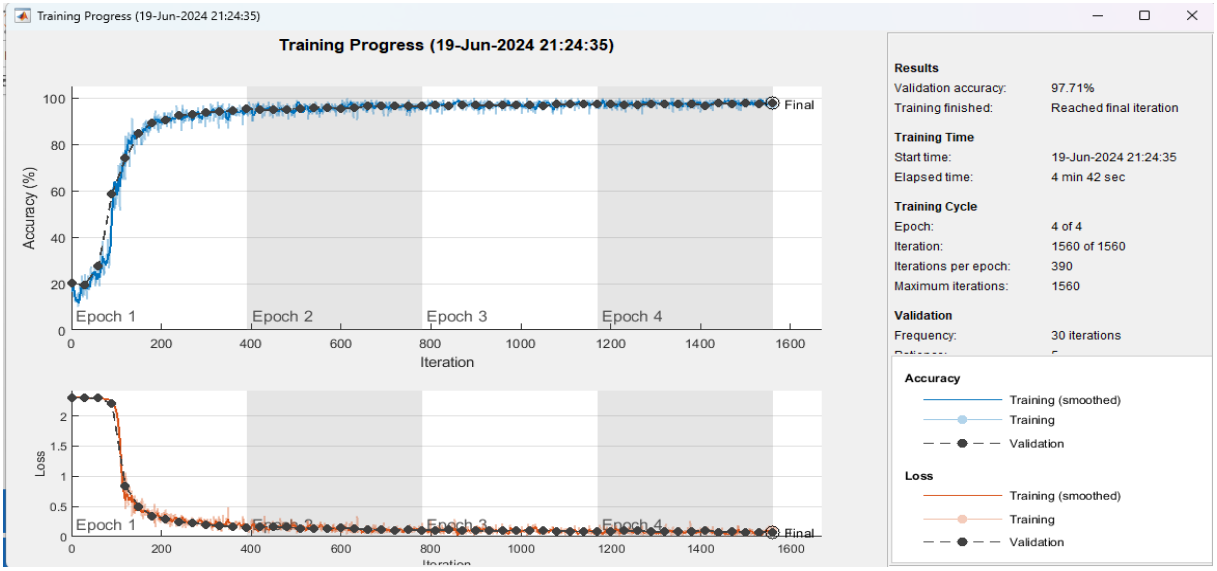


Figure III.4 Graphes affichés durant la phase d'apprentissage du réseau CNN.

d) **Evaluation du réseau :**

On utilise la partie test de la base de données pour calculer le taux de la classification correcte du réseau (justesse ou accuracy). On injecte toutes les images de test l'une après l'autre à l'entrée du réseau qui va les classifier et on compare le résultat de classification avec les valeurs correctes des classes puis on calcule le taux de la classification correcte. Pour notre application on constate que l'accuracy atteint la valeur de 97.71% qui est la même que celle de l'accuracy de validation.

e) **Sauvegarde du réseau** : lorsque on arrive au meilleur résultat de classification, on peut sauvegarder les paramètres du réseau dans un fichier de format spécifique afin de pouvoir l'utiliser pour faire l'inférence sans avoir besoin de faire l'apprentissage à nouveau. Pour cela il suffît de charger le réseau à partir du fichier de sauvegarde. L'opération est réalisée à l'aide des deux fonctions save et load : save (filename, 'net') ; et loadednet= load('filename.mat');

III.4.2 En utilisant PYTHON

Les étapes de développement de l'application de classification sous l'environnement JUPYTER et en utilisant le langage de programmation Python sont :

a) Importation des librairies :

Dans la plupart des langages de programmation, l'importation d'une bibliothèque est une étape essentielle pour utiliser les fonctionnalités qu'elle propose. Pour cette application nous avons besoins des bibliothèques suivantes :

```
import tensorflow as tf
import matplotlib.pyplot as plt
import numpy as np
from keras.models import Sequential
from keras.layers import Dense, Dropout, Activation, MaxPooling2D, Conv2D, Flatten
```

b) Chargement de la base de donnéesMNIST :

Puisque la base de données MNIST est intégré à la bibliothèque keras, on peut la charger directement comme suivant :

mnist=tf. keras. datasets. mnist

(x_train, y_train), (x_test, y_test) =mnist. load_data ()

Cela permet de charger les matrices correspondantes aux images d'apprentissage dans x_train, leurs classes dans y_train, les matrices des images de test dans x_test et leurs classes dans y_test.

La répartition des images est comme suivant :

- Nombre d'image d'apprentissage=6000*10=60000.
- Nombre d'images de test=1000*10=10000.
- Nombre total : 7000*10=70000.

Ensuite la troisième dimension de chaque image est spécifiée pour pouvoir l'injecter à la première couche du réseau CNN.

c) Spécification de l'architecture du réseau CNN :

Le segment suivant du programme donne la manière de spécifier la structure du réseau sous Python et en utilisant la bibliothèque Keras.

```
model=Sequential()
model.add(Conv2D(8, (3,3),input_shape=x_trainr.shape[1:]))
model.add(Activation("relu"))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Conv2D(16, (3,3),input_shape=x_trainr.shape[1:]))
model.add(Activation("relu"))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Conv2D(32, (3,3),input_shape=x_trainr.shape[1:]))
model.add(Activation("relu"))
model.add(Flatten())
model.add(Dense(10))
model.add(Activation("Softmax"))
```

Le tableau III.3 résume l'architecture du réseau après l'étape de spécification de sa structure.

d) Apprentissage du réseau

Avant l'apprentissage du réseau CNN,on commence d'abord par le choix des options d'apprentissage en suite en lance le processus d'apprentissage comme suivant :

```
from tensorflow.keras.optimizers import SGD
sgd_optimizer = SGD(learning_rate=0.001)
model.compile(loss="sparse_categorical_crossentropy", optimizer=sgd_optimizer, metrics=['accuracy'])
```

```
history=model.fit(x_trainr, y_train, epochs=5, validation_split=0.3)
```

Tableau III.3 Résumé de l'architecture du réseau utilisé pour la classification de digits manuscrits

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 26, 26, 8)	80
activation (Activation)	(None, 26, 26, 8)	0
max_pooling2d (MaxPooling2D)	(None, 13, 13, 8)	0
conv2d_1 (Conv2D)	(None, 11, 11, 16)	1168
activation_1 (Activation)	(None, 11, 11, 16)	0
max_pooling2d_1 (MaxPooling2D)	(None, 5, 5, 16)	0
conv2d_2 (Conv2D)	(None, 3, 3, 32)	4640
activation_2 (Activation)	(None, 3, 3, 32)	0
flatten (Flatten)	(None, 288)	0
dense (Dense)	(None, 10)	2890
activation_3 (Activation)	(None, 10)	0

Total params: 8,778
Trainable params: 8,778
Non-trainable params: 0

La figure suivante résume l'évolution du processus d'apprentissage. On remarque que le réseau peut assurer un taux d'accuracy de 96.04% pour les données de validation et de 96.43% pour les données d'apprentissage.

```
Epoch 1/5
1313/1313 [==============================] - 159s 120ms/step - loss: 0.6700 - accuracy: 0.8264 - val_loss: 0.2717 - val_accuracy: 0.9177
Epoch 2/5
1313/1313 [==============================] - 143s 109ms/step - loss: 0.2274 - accuracy: 0.9307 - val_loss: 0.2001 - val_accuracy: 0.9378
Epoch 3/5
1313/1313 [==============================] - 142s 108ms/step - loss: 0.1681 - accuracy: 0.9493 - val_loss: 0.1658 - val_accuracy: 0.9495
Epoch 4/5
1313/1313 [==============================] - 139s 106ms/step - loss: 0.1371 - accuracy: 0.9577 - val_loss: 0.1444 - val_accuracy: 0.9565
Epoch 5/5
1313/1313 [==============================] - 138s 105ms/step - loss: 0.1177 - accuracy: 0.9643 - val_loss: 0.1359 - val_accuracy: 0.9604
```

d) Evaluation :

Après l'étape d'apprentissage on calcule le taux d'accuracy sur l'ensemble de test comme suivant :

```
loss, accuracy = model.evaluate(x_testr, y_test)
print('Test accuracy:', accuracy)
```

```
313/313 [==============================] - 1s 3ms/step - loss: 0.0750 - accuracy: 0.9772
Test accuracy: 0.9771999716758728
```

On remarque que le taux accuracy de la partie test obtenu en utilisant python est exactement le même que celui obtenu en utilsant Matlab (97.71%) .

e) **Sauvegarde du réseau** :

Pour sauvegarder un modèle et pour charger un modèle déjà sauvegardé on précède comme suivant :

```
!mkdir -p saved model
model.save('saved_model/digits1.h5')
```

```
from tensorflow.keras.models import load_model
```

```
loaded_model=load_model('saved_model/digits1.h5')
```

IV. Conception des circuits de calcul sous System Generator

Cette étape est pratiquement le cœur de notre travail dans laquelle nous avons fait la conception des différents circuits qui vont effectuer les différentes opérations du réseau CNN. Ces circuits vont être ensuite convertis en RTL (registre transfert Level) et implémentés sur le la carte Pynq-z1 cible.

Pour la conception des circuits de calcul de la convolution, de ReLU et de Max pooling nous avons utilisé le logiciel System Générateur et pour faire la mise en forme des données après chaque étape de calcul nous avons utilisé le logiciel Matlab.

La figure III.5 donne le schéma bloc global des différents circuits que nous avons conçu en utilisant Matlab et System Generator.

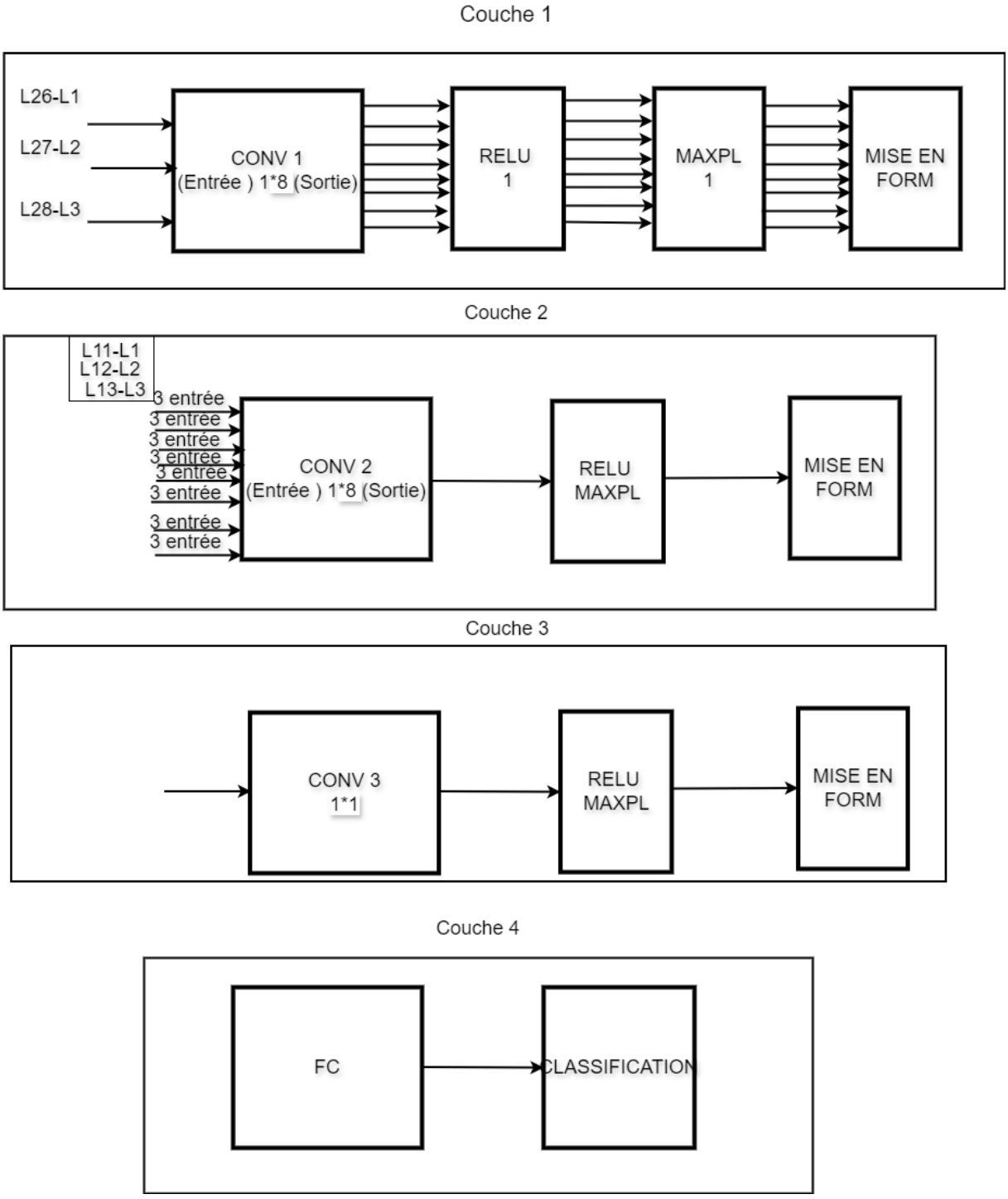


Figure III.5 : Schéma synoptique de l'architecture globale du réseau CNN.

Le diagramme est divisé en quatre sections principales étiquetées "Couche 1", "Couche 2", "Couche 3" et "Couche 4".

- **La couche 1** comprend le circuit de conception de la première convolution (CONV 1) avec une entrée et huit sorties, suivie d'une fonction d'activation RELU 1) à huit entrées et huit sorties et un circuit de maxpooling MAXPL 1, et un bloc de mise en forme des données.
- **La couche 2** est composée d'un circuit de calcul de convolution (CONV 2) avec huit entrées et une seule sortie suivie d'un circuit de ReLU et MAXPL chacun à une seule entrée et une seule sortie et finalement d'un bloc de mise en forme.
- **La couche 3** contient un circuit de convolution à entrées séries et sorties séries suit un (CONV 3), en plus des circuits de ReLU et MAXPL et un bloc de mise en forme de données.
- **La couche 4** est formée d'un circuit de calcul de la couche entièrement connectée à une entrée et dix sorties suivi du circuit de classification.

IV.1 Circuit de calcul de la convolution 1

Le circuit proposé permet de calculer d'une façon parallèle les 8 sorties de maps de caractéristiques de la première couche de convolution ligne après ligne. Afin de minimiser le nombre de cycle de retard nous avons proposé la structure des entrées montrées dans la figure III.6.

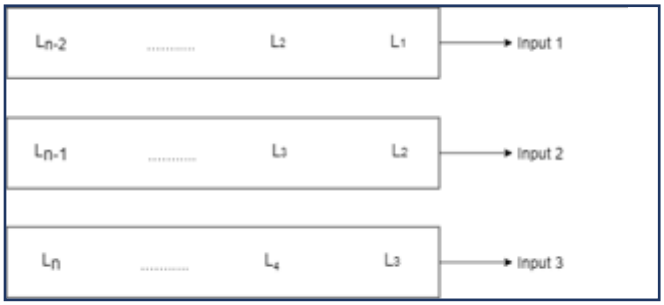


Figure III.6 structures des entrées de la première couche de convolution.

Le circuit de calcul est composé de huit branche identiques parallèles, chaque branche permet le calcul de l'une des huit matrices de convolution de sortie comme illustré dans la figure III.7.

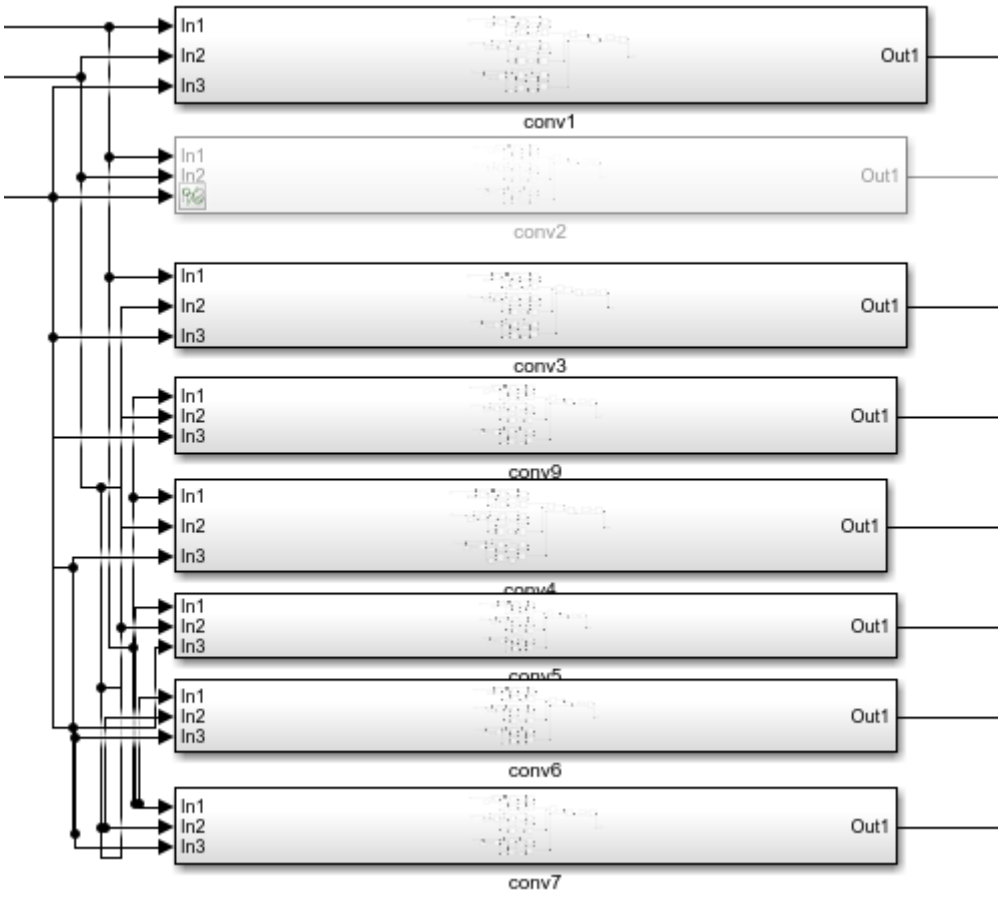


Figure III 7 : Schéma du circuit de calcul de la première convolution

Chacune des huit branches contient : 9 multiplieurs de nombres réels, 9 additionneurs de nombres réels, 6 éléments de retard et 10 générateurs de constantes pour sauvegarder les neuf paramètres d'un filtre de dimension 3×3 et un seul biais. La figure III.8 donne la structure détaillée d'une branche de calcul et la figure III.9 donne le schéma de conception à l'aide du logiciel System Generator.

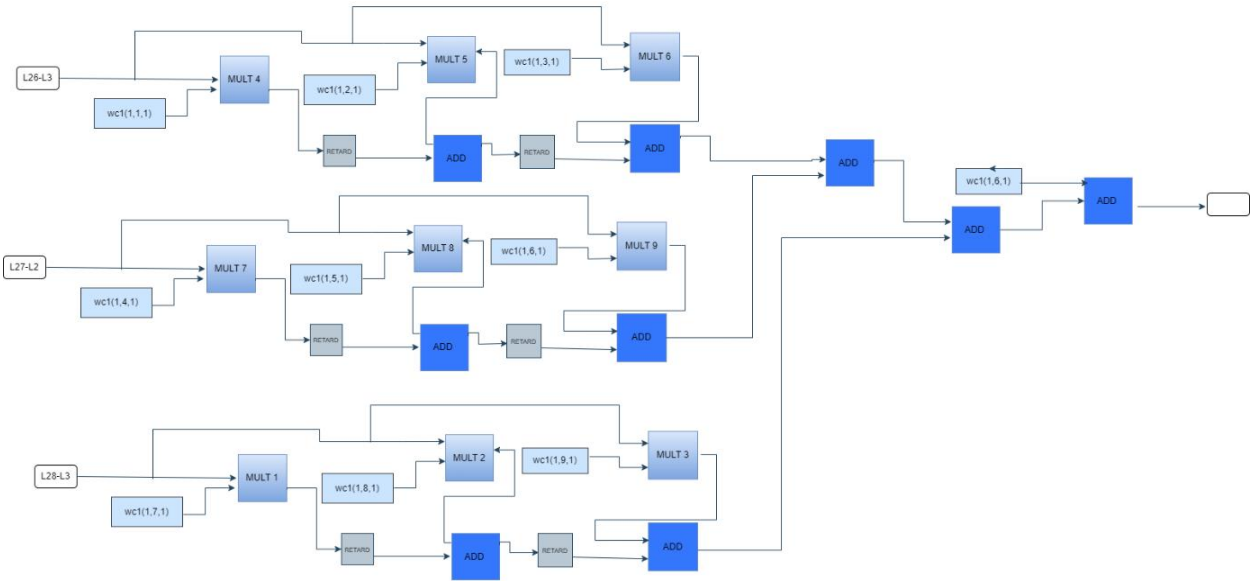


Figure III 8 : Architecture détaillée de l'une des huit branches du circuit de calcul de la première convolution.

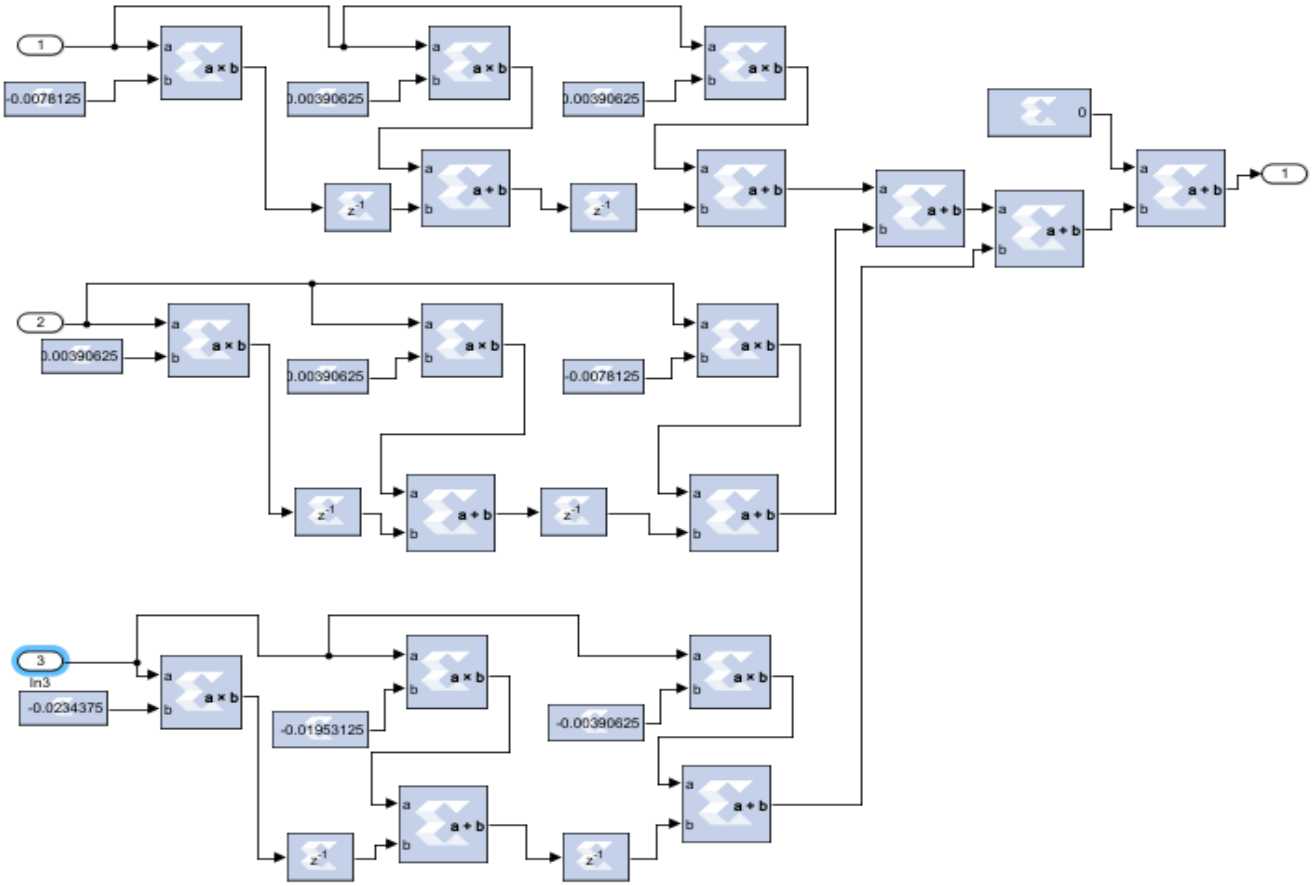


Figure III 9 : Schéma de conception d'une branche du circuit de calcul de la première convolution.

IV.2. Circuit de calcul ReLU 1 et maxpoling 1

Le circuit de calcul de ReLu est très simple, il est composé de huit branches identiques dont chaque branche contient un multiplexeur, un extracteur du bit de signe et un élément de mémorisation.

Le circuit de calcul du max pooling est formé de huit branches, la figure III.10 donne le schéma de conception de l'une des huit branches. Chaque branche est composée d'un élément de retard de 28 cycles horloge, d'un élément de retard d'un seul cycle, d'un élément de sous échantillonnage et de deux éléments de calcul de maximum. L'élément de calcul de maximum permet de calculer le maximum de deux entrées, il est composé d'un comparateur et d'un multiplexeur.

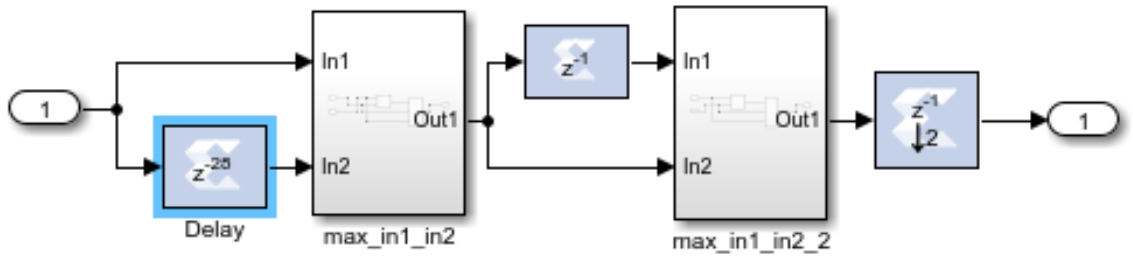


Figure III.10 Structure de l'une des huit branches du circuit de calcul du max pooling 1.

IV.3 Circuit de calcul de la convolution 2

Le circuit de calcul pour la deuxième convolution permet de prendre les huit maps de caractéristiques d'entrées en parallèle et de calculer les 16 maps de caractéristiques de sorties en série l'une après l'autre et ligne après ligne. Il est composé de 8 huit branches dont chaque branche prend en charge l'une des huit entrées. Afin de permettre le calcul série des sorties les données d'entrées sont restructurées selon le schéma de la figure III.11.

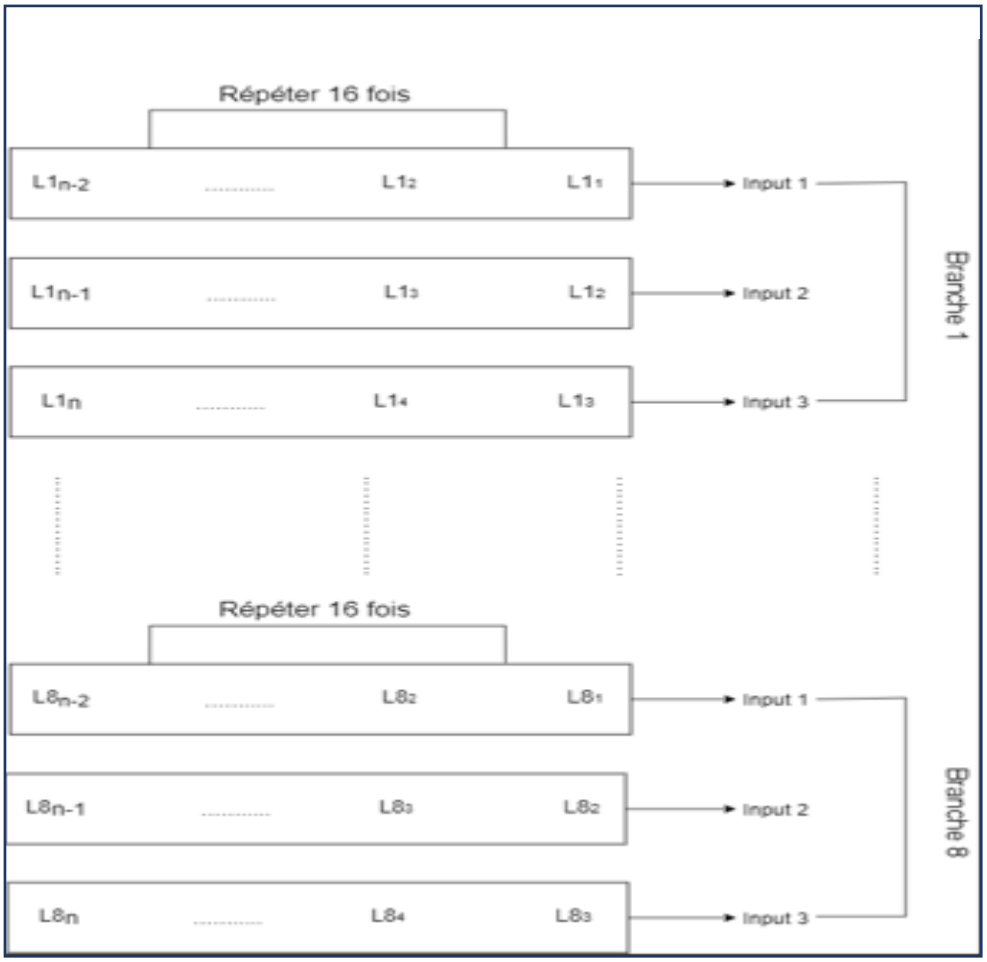


Figure III.11 structures des entrées de la deuxième couche de convolution.

Les sorties des huit branches sont additionnées à l'aide de 7 additionneurs pour former une convolution unique. Cette dernière est additionnée à un biais pour former une map de caractéristiques. Le schéma de conception du circuit de calcul de la deuxième convolution est illustré dans la figure III.12.

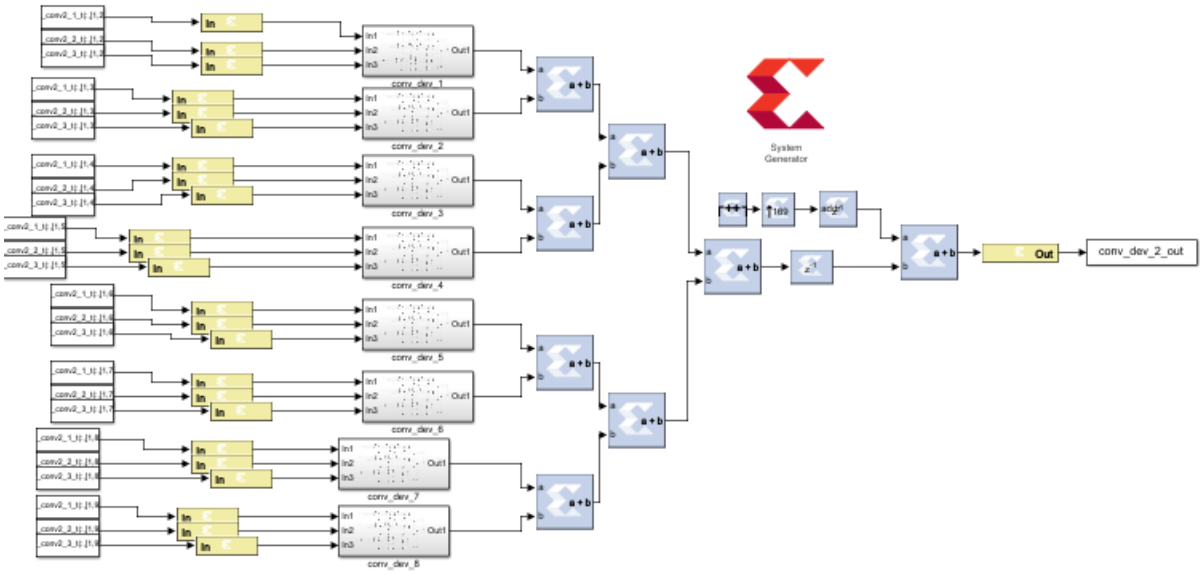


Figure III.12 circuit de conception du circuit de calcul de la convolution 2.

IV. 4 Circuit de calcul série de ReLU 2 et Max pooling 2

Le circuit de calcul de ReLU 2 possède uniquement une seule branche car les sorties sont en série. Le circuit de calcul de Maxpooling 2 lui aussi possède une seule branche identique à celle donnée dans la figure III.10.

IV.5 Circuit de calcul série de la convolution 3

Le circuit de calcul de la troisième convolution prend les 16 matrices données en série et calcul les 32 matrices de sorties en série. Pour prendre les entrées en série et faire le calcul des sorties en série nous avons proposé la structure des données d'entrée selon la figure III.13.

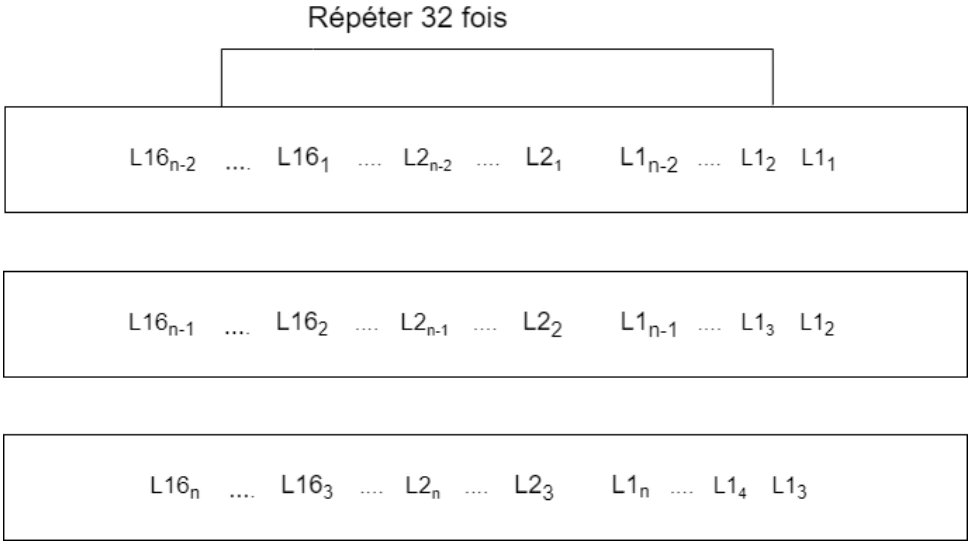


Figure III.13 structures des entrées de la troisième couche de convolution.

Puisque les entrées sont injectées en série et le calcul se fait en série, le circuit de calcul de la troisième couche de convolution possède uniquement une seule branche de calcul comme le monte la figure III.14.

L'utilisation de circuit de calcul série permet de réduire significativement le nombre d'éléments de calcul et de mémorisation en comparaison avec les circuits de calcul parallèle surtout pour la couche Conv 3. Pour un circuit parallèle en entrée et en sortie on doit 9×16×32 multiplieurs alors que le circuit de calcul série n'utilise que 9 multiplieurs. Il est important de noter que le circuit PYNQ-Z1 possède uniquement 220 DSP48 pour la multiplication.

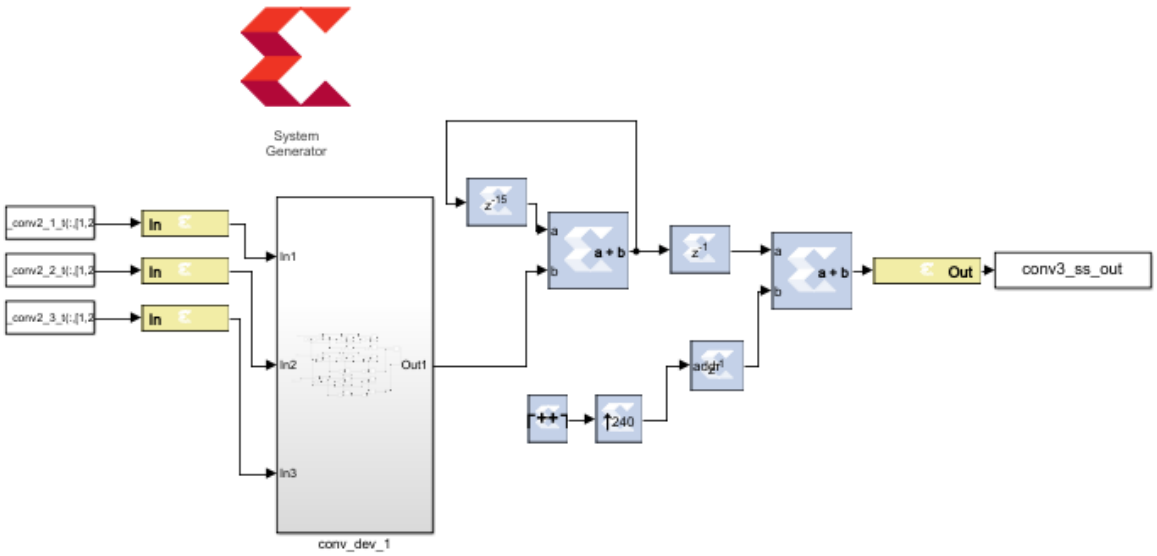


Figure III.14 : Schéma de conception du circuit de calcul de la convolution 3.

Les circuit de calcul série de ReLU 3 et de Maxpooling 3 sont identiques à ceux des deux circuits Relu2 et Maxpooling 2 respectivement.

IV.6 Circuit de calcul de la couche FC et classification

Le schéma de conception est composé de deux partie comme montré dans la figure III.15. La première partie concerne la couche FC du réseau et permet de calculer le score de classification correspondant à chacune des dix classes possible en utilisant dix branches identiques. La deuxième partie permet de donner le résultat de classification ou très précisément les numéros de la calasse ayant le score maximal et la valeur du score maximal.

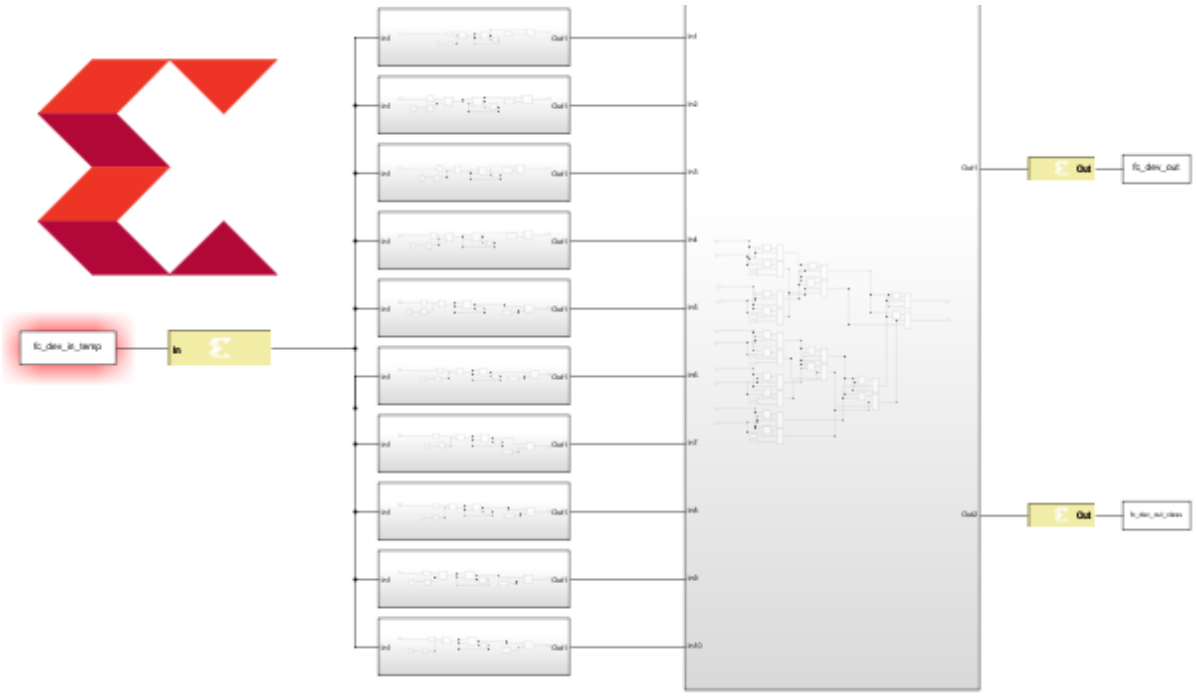


Figure III.15 : Schéma de conception du circuit de la couche FC et la couche de classification.

V. Résultats de l'implémentation

V.1 Etapes d'implémentation

Après vérification du bon fonctionnement de chacun des circuits de conception expliqués dans le paragraphe précédent, nous avons généré leurs model HDL « Hardware Discription Language ». Ensuite nous avons passés à l'environnent Vivado pour faire l'implémentation des différents circuits en utilisant la représentation en virgule fixe et aussi en virgule flottante.

Durant la phase d'implémentation on commence d'abord par réaliser la synthèse du circuit. La figure III.16 est une capture de la fenêtre Vivado après la fin de la synthèse de l'un des circuits de calcul.

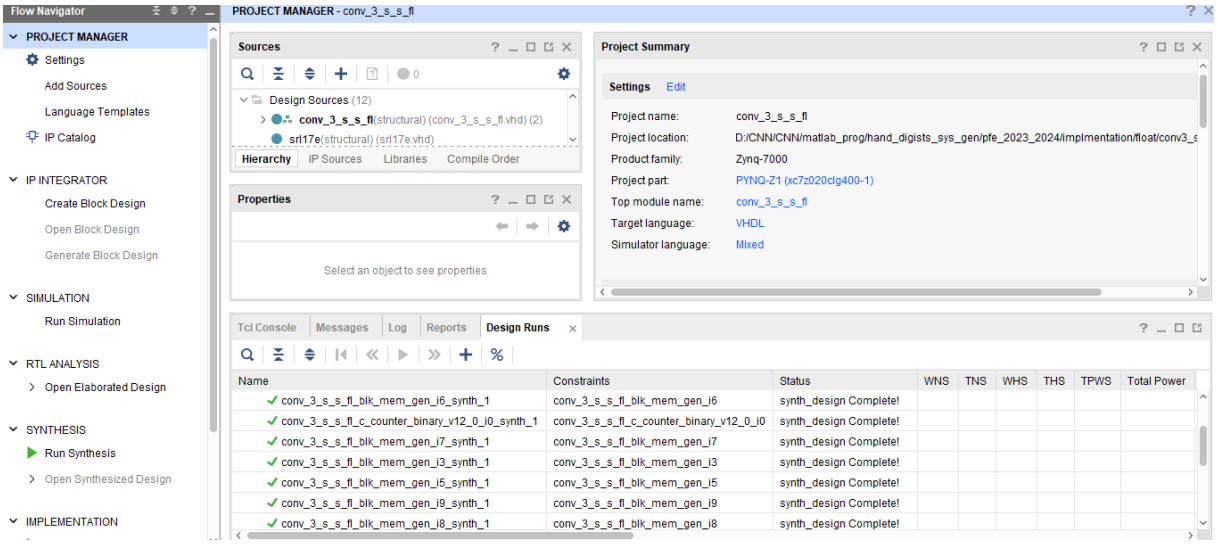


Figure III.16 : capture de la fenêtre Vivado après l'étape de synthèse de l'un des circuits de calcul.

Une foi l'opération de synthèse est achevée sans erreurs on précède à l'étape d'implémentation pour générer les résultats d'implémentation. La figure III.17 est une capture de la fenêtre Vivado après l'étape d'implémentation.

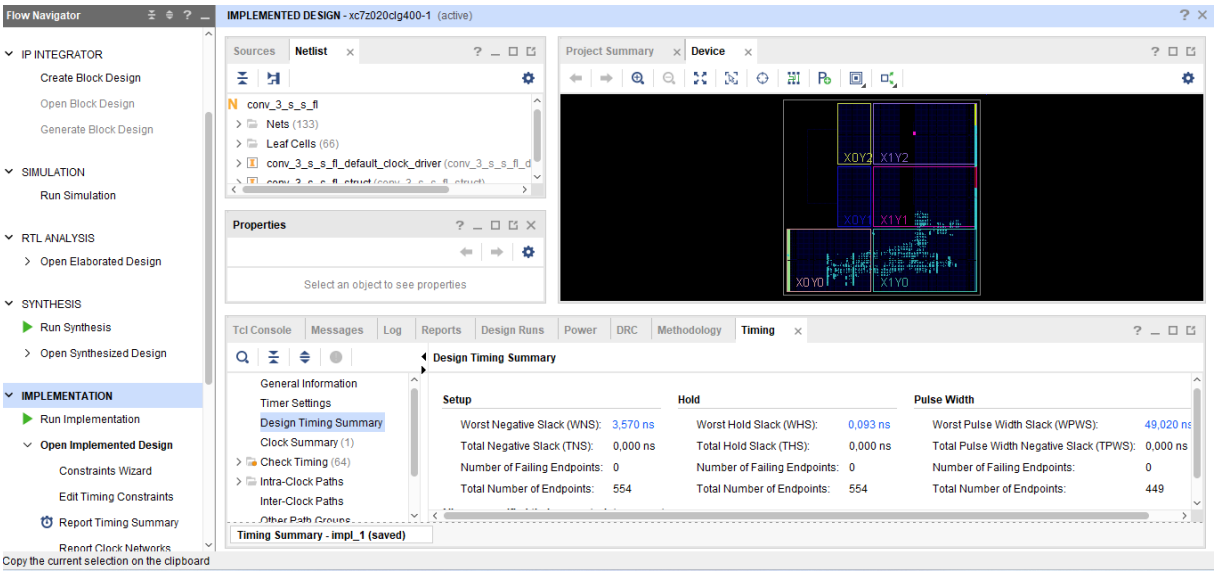


Figure III.17 : Fenêtre Vivado à la fin de l'étape d'implémentation.

Après la fin de l'étape d'implémentation on peut connaitre la consommation en ressources et en puissance de chacun des circuits de calcul.

V.2 Résultats d'implémentation en virgule flottante :

La virgule flottante est une méthode utilisée pour représenter un très vaste intervalle de nombres réels dans les systèmes informatiques, permettant de conserver une précision relative élevée sur une large gamme de grandeurs. Ce système de représentation est particulièrement utile dans des domaines comme les calculs scientifiques et l'ingénierie, où l'on doit manipuler à la fois des nombres très grands et très petits.

Le tableau III.4 résume les ressources de la carte PYNQ-Z1 et la puissance consommées par les différents circuits en utilisant la représentation en virgule flottante.

Le Tableau III.4 : Les ressources et la puissance dynamique consommées par les différents circuits de calcul implémenté sur ZYNQ XC7Z020-1CLG400C en utilisant la représentation en virgule flottante.

	Slice registers		**Slice LUT**		**FF**		**DSP**		**BRAM**		**POWER(W)**			
											static		Dynamic	
	Utilisation	Pourcentage	Utilisation	Pourcentage	Utilisation	Pourcentage	Utilisation	Pourcentage	Utilisation	Pourcentage	Utilisation	Pourcentage	Utilisation	Pourcentage
Conv1	**1536**	**1.44%**	**29 728**	**55.88 %**	**1000**	**0.13%**	**216**	**98.18**	**00**	**0.00%**	**0.104**	**78%**	**0.030**	**22%**
Relu1 maxpl1	**132**	**0.12%**	**198**	**0.37%**	**70**	**53200**	**00**	**00%**	**00**	**0.00%**	**0.104**	**99%**	**0.001**	**1%**
Conv2	**3043**	**2.86%**	**30893**	**58.06 %**	**1091**	**2.05%**	**216**	**98.18 %**	**36.5**	**26.07 %**	**0.108**	**72%**	**0.042**	**28%**
Relu2 maxpl2	**132**	**0.12%**	**183**	**0.34%**	**50**	**0.09%**	**00**	**0%**	**0**	**140**	**0,104**	**99%**	**0.001**	**1%**
Conv3	**396**	**0.37%**	**4223**	**7.94%**	**160**	**0.30%**	**27**	**12.27 %**	**5**	**3.57%**	**0.105**	**95%**	**0.006**	**5%**
Relu 3	**0**	**0%**	**16**	**0.03%**	**0**	**0%**	**0**	**0%**	**0**	**140**	**0.104**	**99%**	**0.001**	**1%**
FC class	**730**	**0.686 %**	**8447**	**15.88 %**	**18**	**0.034 %**	**30**	**13.64 %**	**5**	**3.57%**	**0.105**	**76%**	**0.032**	**24%**

A partir du tableau on peut tirer les conclusions suivantes :

- Le circuit de calcul de convolution 2 consomme plus de ressources en comparaison avec tous les autres circuits.
- L'utilisation des circuits de calcul série (cas de conv3) réduit significativement les ressources utilisées et la puissance consommée.
- Les couches de convolution nécessitent plus de ressources FPGA et plus puissance par rapport au couches ReLU, Max pooling et FC.
- Les ressources disponibles sur le circuit ne sont pas suffisantes pour implémenter la totalité du réseau ce qui nécessite l'utilisation de la virgule fixe pour la représentation des données.

V.3 Résultats d'implémentation en virgule fixe

La représentation en virgule fixe est particulièrement utile dans les systèmes où les ressources sont limitées (comme dans le matériel embarqué), ou dans les applications où le temps de traitement doit être très prévisible et cohérent, puisque les calculs en virgule fixe prennent généralement un temps d'exécution constant.

Dans notre implémentation et après études des valeurs obtenues dans les différentes couche de convolution nous avons choisi le format 16.8 (16 bits dont 8 pour la partie fractionnaire).

Le Tableau III.5 montre les ressources FPGA utilisées par les circuits de calcul et la puissance consommée par les circuits en cas de la représentation des données en virgule fixe du réseau CNN proposés.

A partir des résultats du tableau III.5 et en les comparant avec les résultats du tableau III.4 pn peut constater que :

- L'utilisation du format de représentation en virgule fixe améliore l'efficacité d'implémentation d'une façon très significative en comparaison avec le format en virgule flottante. Par exemple pour la couche conv2 le nombre de Slice registers passe de **3043 à 1475**, le nombre Slice Luts passe de **30893** à 1235 et le nombre de DSP48E passe de **216** à **72.**
- Les ressources disponibles sur le circuit ZYNQ XC7Z020-1CLG400C sont suffisantes pour l'implémentation de la totalité du réseau malgré qu'ils soient limitée en comparaison avec d'autres circuits ZYNQ.

- La puissance dynamique consommée par le circuit est très limitée ce qui offre la possibilité d'alimenter le circuit par une batterie pour les applications embarquées.

Tableau III.5: Ressources et *puissance consommées par les différentes couches de calcul avec les données en virgule fixe.*

	Slice registers		**Slice LUT**		**FF**		**DSP**		**BRAM**		**POWER(W)**			
											Static		Dynamic	
	Utilisation	Pourcentage	Utilisation	Pourcentage	Utilisation	Pourcentage	Utilisation	Pourcentage	Utilisation	Pourcentage	Utilisation	Pourcentage	Utilisation	Pourcentage
Conv1	**768**	**0.722%**	**1024**	**1.924%**	**348**	**0.654%**	**72**	**32.73%**	**00**	**0%**	**0.104**	**99%**	**0.001**	**1%**
Relu1 maxpl1	**63**	**0.06%**	**68**	**0.10%**	**36**	**0.05%**	**00**	**0%**	**00**	**140**	**0.104**	**99%**	**0.001**	**1%**
Conv2	**1475**	**1.38%**	**1235**	**2.32%**	**483**	**0.91%**	**72**	**32.73**	**36.5**	**2.14%**	**0.105**	**99%**	**0.001**	**1%**
Relu2 maxpl2	**68**	**0.06%**	**56**	**0.10%**	**27**	**0.05%**	**00**	**0%**	**0**	**0%**	**0.104**	**99%**	**0.001**	**1%**
Conv3	**220**	**0.20%**	**187**	**0.35%**	**91**	**0.17%**	**9**	**4.09%**	**5**	**3.57%**	**0.105**	**98%**	**0.002**	**2%**
Relu 3	**0**	**0%**	**8**	**0.01%**	**0**	**0%**	**0**	**0%**	**0**	**0%**	**0.104**	**99%**	**0.001**	**1%**
FC et class	**410**	**0.39%**	**466**	**0.88%**	**170**	**0.32%**	**10**	**4.55%**	**5**	**3.57%**	**0.105**	**93%**	**0.007**	**7%**

VI. Conclusion

Dans ce chapitre, nous avons détaillé les différentes étapes de la réalisation de notre projet. Nous avons débuté par la conception, l'apprentissage et la validation du réseau convolutif destiné à la reconnaissance automatique de chiffres manuscrits. Pour choisir l'architecture du réseau, nous avons recherché un équilibre entre simplicité et précision. Ainsi, nous avons opté pour un réseau comportant trois couches de convolution, trois couches de ReLu, deux couches de Maxpooling et une couche entièrement connectée. Le réseau sélectionné atteint une précision de près de 97,91%.

Dans la deuxième étape nous avons fait la conception des circuits qui réalisent les différentes opérations pour les diverses couches. Vue la limitation de la carte PYNQ-Z1 en ressources de la partie PL nous avons choisi une architecture qui limite la consommation de ressources. Durant cette étape nous avons constaté que les circuits de calcul de convolution sont les plus compliqués.

Dans la dernière étape de notre projet nous avons fait l'implémentation des circuits développés dans l'étape précédente sur la partie PL du circuit PYNQ Z1 en utilisant le logiciel Vivado. Les rapports d'utilisation de ressources montrent que les circuits de convolution sont les plus exigent en termes de ressources consommées. Pour limiter ses ressources nous avons utilisé la représentation en virgule fixe au lieu de la virgule flottante.

Conclusion générale

Conclusion générale

La classification d'images est une tâche essentielle dans diverses applications de la vision par ordinateur, telles que la reconnaissance d'objets, la détection de visages et la surveillance vidéo. Elle consiste à classer les images dans des catégories prédéfinies en fonction de leurs caractéristiques visuelles. Vue leur capacité exceptionnelle pour la classification des images, les réseaux de neurones convolutifs (CNN) ont été largement utilisés avec succès pour cette tâche. Cela grâce à leurs caractéristiques distinctives d'extraction automatique des caractéristiques discriminantes, de profondeur de couche, de capacité de généralisation et d'utilisation de données massives.

Pour réaliser une inférence efficace d'un modèle CNN, la plateforme d'exécution doit posséder des ressources de calcul et de mémorisation adéquates. Une des solutions les plus efficaces pour implémenter des réseaux CNN consiste à utiliser une plateforme combinant des ressources logiques reconfigurables de type FPGA et des capacités de traitement programmable de type ARM. Dans ce contexte, les circuits Zynq offrent une puissante combinaison de traitement logiciel et matériel sur une seule puce, les rendant particulièrement adaptés aux applications nécessitant à la fois des performances élevées et une grande flexibilité.

Dans ce cadre de travail, nous avons fait une implémentation d'un système de classification de digits manuscrits à base de réseau CNNsur la carte PYNQ-Z1 qui est basée sur un circuit ZYNQ.

Dans la partie théorique, nous avons étudié la classification d'images en nous concentrant sur les réseaux CNN. Nous avons approfondi notre compréhension de leur fonctionnement, de leur architecture et de leur utilisation dans le domaine de l'intelligence artificielle. Par ailleurs, nous avons exploré les circuits Zynq, incluant les composants programmables (FPGA) et les processeurs ARM.

Dans la partie pratique nous avons détailléles étapes nécessaires à la mise en œuvre des connaissances théoriques acquises. Ces étapes la sélection de l'architecture du réseau CNN, la programmation et l'apprentissage et le processus de conception et d'implémentation d'un réseau de reconnaissance de chiffres manuscrits, de l'architecture initiale à l'implémentation matérielle, en passant par les outils logiciels et les considérations de représentation des données.

L’évaluation du réseau CNN programmé en Matlab et en Python a confirmé sa capacité à atteindre un taux de justesse (Accuracy) de classification de 97.91. Ce taux est très satisfaisant vue la simplicité de l’architecture qui contient trois couches de convolution deux couches de maxpooling et une seule couche entièrement connectée.

Pour l’implémentation du réseau CNN sur la carte PYNQ-Z1, nous avons opter pour une architecture qui minimise les ressources utilisées par l’utilisation hybride de circuit de calcul parallèle et série. En outre, et pour minimiser d’avantage les ressources nous avons utilisé la représentation des données en virgule fixe au lieu de la virgule flottante.

Les résultats d’implémentation montrent que le réseau ne consomme que 0.799% de sliceregisters, 19.786% de slice LUT, 0.432% de flip-flop, 0.475% de DSPet 31,752% de BRAM du circuit ZYNQ XC7Z020-1CLG400C. Les architectures proposés permettent aussi de réduire la puissance dynamique consommée par le circuit.

Les résultats obtenus sont prometteurs et mettent en lumière l'importance de la convergence entre l'apprentissage profond et les systèmes embarqués. En intégrant des réseaux CNN sur des plateformes matérielles comme les circuits Zynq, nous ouvrons la voie à des applications innovantes et performantes dans des domaines tels que la vision par ordinateur et l'analyse d'images en temps réel.

Malgré les succès rencontrés, certaines limites ont été identifiées. Parmi celles-ci, la complexité de la conception et de la vérification du bon fonctionnement des différentes couche. ainsi que les contraintes de ressources limitées pour les circuits à cout raisonnable tel que le PYNQ-Z1. Cela nécessite une attention continue pour optimiser les performances tout en limitant l’utilisation de ressources et la consommation de puissance.

Plusieurs améliorations peuvent être considérées comme perspectives de notre travail, entre autres on peut citer :

- Proposition d’une architecture d’implémentation de la partie de mise en forme des données sur la partie PS du circuit.
- Implémentation de réseaux CNN plus profond sur circuit FPGA et Zynq.

Références

[1] Platon, « La République, livres IV-VII », Traduction par Émile Chambry. Les Belles Lettres, 1933 (Tome VII, p. 202-286).

[2] Rafael C.Gonzalez & Richard E.Woods , «Digital Image Processing», Pearson Education Inc, 2008.

[3] Mokri M. Z., « Classification des images avec les réseaux de neurones convolutionnels » , mémoire de Master, université de Tlemcen, 2017.

[4] MEZZOUG, **K.** « traitement et analyse des images numériques cours pour : master génie informatique », 2019/2020.

[5] M. Sandeli, « Traitement d'images par des approches bio-inspirées application à la détection et à la segmentation d'images », mémoire de magistère, *Université de Constantine 2*, 2014.

[6] R.C. Gonzalez et Woods. « Digital Image Processing - 2d edition ». Addison Wesley 2002.

[7] A. Medjaoui, « Les segmentations des images par contours actifs : application à l'analyse des images satellitaires à haute résolution », mémoire de Master, *Université Abou Bakr Belkaid–Tlemcen*, 2012.

[8] T. Mohamed, D. Djafar, « La détection des contours de scènes sémantiques avec plusieurs niveaux d'abstraction pour la recherche d'images », mémoire de Master, Université Kasdi Merbah Ouargla, 2013.

[9] T Guyer. « Cours de traitement d'images ». Université de Chambéry**.**

[10] J.P. Cocquerez et S. Philipp. « Analyse d'images : filtrage et segmentation ». Masson 1995.

[11] M. J. Chichi, « Algorithmes de traitement des images par un modèle de réseaux de neurones (comparaison avec les filtres de base) », mémoire de Master, *Université Abou Bakr Belkaid Tlemcen*, 2011.

[12] EL HABIB, M. et CHAHBOUB, M. R. « Implémentation efficace d'un réseau de neurones convolutif (CNN) sur circuit FPGA » mémoire de Master, université de Médea, 2*022.*

[13] HEATON, Jeff. Ian goodfellow, yoshua bengio, and aaron Courville « Deep learning » The mit press, 2016.

[14] Bouhedadja, M.A., *«* Utilisation de l'apprentissage profond pour la classification des mauvaises herbes dans le bleuet nain », Mémoire de maitrise, Université du Québec à Trois-Rivières, 2021.

[15] Myriam Emilion, « Les algorithmes de Deep Learning », Directive Marketing, 2024.

[16] RUSSELL, Stuart J. et NORVIG, Peter. « *Artificial intelligence: a modern approach*», Pearson, 2016.

[17] SÉROT, Jocelyn « Les circuits FPGA et le langage VHDL, une intro pour les programmeurs et par l'exemple » Editions Ellipses, 2019.

[18] Xilinx «Fiche technique du Zynq 7000», 2018,

[19] Xilinx, «Zynq-7000 All Programmable SoC Technical Reference Manual UG585 (v1.7) » , February 11, 2014.

[20] Louise H. Crockett , Ross A. Elliot, Martin A. Enderwitz, Robert W. Stewart « The Zynq Book Embedded Processing with the ARM® Cortex®-A9 on the Xilinx® Zynq®-7000 All Programmable SoC » University of Strathclyde, Glasgow, Scotland, UK 2014.

[**21**] **Louise H. Crockett, David Northcote, Craig Ramsay, Fraser D. Robinson, Robert W**. **Stewart,** « Exploring Zynq® MPSoC With PYNQ and Machine Learning Applications », University of Strathclyde, Glasgow, Scotland, UK, April 2019.

[22] QUARTERONI, Alfio Maria, SACCO, Riccardo, et SALERI, Fausto, « Méthodes numériques pour le calcul scientifique: programmes en MATLAB » , Springer Science & Business Media, 2000.

[23] LUTZ, Mark. « Programming python » O'Reilly Media, Inc., 2010.

[24] Xilinx-AMD « Vivado Design Suite Reference Guide Model-Based DSP Design Using System Generator » 2020.