REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITE YAHIA FARES DE MEDEA

Faculté de **T**echnologie

Département du **G**énie **E**lectrique

Mémoire de fin d'études de Master

Filière : Télécommunications

Spécialité : Systèmes Des Télécommunications

Implémentation sur circuit FPGA d'un systéme de classification d'armes

---

Présentés par :

1.Chenibet Mohamed Belkhir

2.Mellati Ishak

Proposés et Dirigés par :

Dr. DALI Mohammed

Promotion 2022-2023

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.

Nous souhaitons exprimer notre profonde gratitude, notre reconnaissance et nos remerciements les plus sincères à notre promoteur, Monsieur. « Dr Dali Mohammed» pour ses orientations éclairées, sa modestie, ainsi que pour ses encouragements constants durant toute la progression de ce projet. Nous lui sommes extrêmement reconnaissants pour son soutien précieux.

Nous voudrions remercier sincèrement les membres du jury qui ont accepté de juger notre travail.

En dernier lieu, il est important de rendre hommage à tous les enseignants qui ont joué un rôle essentiel tout au long de notre parcours éducatif. Nous souhaitons exprimer notre reconnaissance sincère envers nos précieux parents, qui ont toujours été présents pour nous soutenir et nous encourager tout au long de notre chemin.

MOHAMED / ISHAK

Dédicace

Je dédie ce mémoire

À mes chers parents, ma mère et mon père, pour leur patience, leur amour inconditionnel et leurs encouragements .

À mes frères et ma soeurs

À tous mes amis

À mon binôme Ishak

À tous mes collègues du Master spécialisé-systèmes de télécommunications

Et enfin à tous ceux qui ont contribué de près ou de loin à la réalisation de ce projet

MOHAMED

Dédicace

Je dédie ce mémoire

À mes très chers parents, dont le sacrifice, l'amour, la patience et le soutien ont été inestimables,

À mes frères,

À mon binôme Mohamed,

À mes amis et mes collègues d'études,

Je souhaite adresser mes remerciements à toutes les personnes qui ont contribué, de près ou de loin, pour que ce projet soit possible.

Ishak

ملخص:

في هذا العمل قمنا بتثبيت نظام تصنيف لخمسة أنواع من الأسلحة بشكل فعال مبني على أساس استعمال الشبكة العصبونية التلافيفية CNN على دائرة من النوع FPGA. تعد شبكات CNN نماذج قوية جدًا لتقنية التعلم العميق (Deep Learning: DL) المستخدمة في مجال تصنيف وكشف الأشكال في الصور ومقاطع الفيديو. تم تصميم نظام التصنيف باستخدام برنامج System Generator وتم تنفيذه على Vertex-7 FPGA باستخدام برنامج Vivado.

تم تصميم الطبقات المختلفة للشبكة باستخدام دوائر الحوسبة المتوازية ثم دوائر الحوسبة التسلسلية لتقليل موارد FPGA اللازمة للتنفيذ. ولقد أثبت النظام المقترح كفاءته وقدم نتائج تصنيف مرضية للغاية.

الكلمات المفتاحية:

- الذكاء الاصطناعي IA، التعلم العميق DL، CNN، FPGA،Vertex-7 ، Xilinx، System Generator، Vivado.

Abstract

In the present work, we have performed an efficient implementation of weapons classification system based on convolutional neural network CNN on Field Programmable Gate Array circuit FPGA. CNNs are very powerful models of the deep learning technique DL used in the field of classification and detection of objects in images and videos. The classification system was designed using System Generator software and implemented on a Xilinx Vertex-7 FPGA using Vivado software.

The different layers of the network have been designed using parallel computing circuits and then serial computing circuits to reduce the FPGA resources necessary for the implementation.

The proposed system has proven its efficiency and has given very satisfactory classification results.

Keywords: Artificial Intelligence (AI), Deep Learning (DL), CNN, FPGA, Vertex-7, System Generator, Vivado, Weapons classification.

Résumé

Dans le présent travail, nous avons implémenté efficacement un système de classification de cinq types d'armes à base de réseau de neurones convolutif CNN (Convolutional neural networks) sur un circuit de type FPGA (FieldProgrammable Gate Array). Les réseaux CNN sont des modèles très puissants de la technique de l'apprentissage profond (Deep learning : DL) utilisés dans le domaine de la classification et de la détection des objets dans les images et les vidéos. Le système de classification a été conçu en utilisant le logiciel System Generator et implimenté sur un circuit FPGA Vertex-7 de Xilinx en utilisant le logiciel Vivado.

Les différentes couches du réseau ont été conçues en utilisant des circuits de calcul parallèle et ensuite des circuits de calcul série pour réduire les ressources FPGA nécessaires à l'implémentation.

Le système proposé a prouvé son efficacité et a donné des résultats de classification très satisfaisants.

Mots clés : Intelligence artificielle (IA), apprentissage en profondeur (DL), CNN, classification d'armes, FPGA, Virtex-7, System Generator , Vivado.

Sommaire

Liste des figures ..... **I**
Liste des tableaux ..... **III**
Liste des abréviations ..... **IV**
Introduction générale **6**
Chapitre I : **8**
De l'Intelligence Artificielle aux réseaux CNN **8**
I.1. Introduction : **9**
I.2.Intelligence Artificielle : **9**
I.2.1 . Domaines d'applications de l'Intelligence Artificielle : **9**
I.3 -Machine Learning : **10**
I.3.1-Types d'apprentissage automatique : **10**
I.3.1.1-Apprentissage Supervisé : **10**
I.3.1.2 .Apprentissage semi-supervisé : **10**
I.3.1.3-Apprentissage Non Supervisé : **11**
I.3.1.4-Apprentissage par Renforcement : 11
I.4.1.Exemples de techniques du Deep Learning : **12**
I.4.2 -Domaines d'application de l'apprentissage profonde : **14**
I.5- réseaux de neurones artificiels : **15**
***I.5.1*-Architecture des réseaux de neurones artificiels :** **16**
I.6-Réseaux de neurones convolutif CNN : **16**
I.6.1 - Architecture des réseaux CNN : **16**
I.6.1.1 -Couche d'entrée : 17
I.6.1.2 - Couches de convolution : 17
I.6.1.3 -Batch Normalization (Normalisation par lots) : 18
I.6.1.4-Les fonctions d'activations : 19
I.6.1.5-Couche sous-échantillonnage (Pooling) : 20
I.6.1.6-Couches complètement connectées (Fully Connected FC) : 21
I.6.1.7. La fonction softmax : 22
I.6.2-Apprentissage d'un réseau CNN : **22**
I.7.Conclusion : **23**
Chapitre II **24**
Les circuits logiques programmables FPGA. **24**
II.1- Introduction : **25**
II.2-Présentation et généralités sur les FPGA : **25**
II.2.1-Définition : **25**

II.2.2- Architecture matérielle des circuits FPGA : **25**
II.2.2.1- Les blocs logiques configurables (CLB) : 26
II.2.2.2. Les Blocs IOB (Input Output Bloc) : 26
II.2.2.3-Blocs d'interconnexion : 27
II.2.2.4-Les blocs à usage spécifique : 28
II.2.2.4.2-Les blocs DSP 28
II.2.3-Les Différents Types d'FPGA : **29**
II.2.3.1- Les FPGA à Anti Fusibles : 29
II.2.3.2- Les FPGA de types SRAM : 29
II.2.3.3- Les FPGA à base de Flash : 30
II.3- Les Familles et caractéristiques des FPGA : **30**
II.3.1 - La famille FPGA Virtex-7 : **30**
II.3.1.1.- Caractéristiques de La famille FPGA Virtex-7 : 31
II.3.2- La famille FPGA Zynq : **32**
II.3.2.1-Caractéristiques de la famille FPGA Zynq : 33
II.3.3- La famille UltraScale FPGA : **33**
II.3.3.1- Les Caractéristiques de La famille UltraScale FPGA : 34
II.4- Les Défis de la mise en œuvre de CNN sur les circuits FPGA : **35**
II.5. Environnement de travail : **35**
II.5. 1-Vivado Design Suite : **35**
II.5.2. System Générateur (XSG) : **36**
II.6-Conclusion : **37**
Chapitre III **38**
Résultats de simulation et d'implémentations **38**
III.1- Introduction : **39**
III.2- Prétraitement et collecte de données **39**
III.2.1 Collecte des images **39**
III.2.2- Redimensionnement : **40**
III.2. 3-Conversion des images en niveaux de gris et en couleurs RVB : **41**
III. 3 Conception et évaluation du réseau CNN : **42**
III.3.1 Architecture du réseau CNN proposé : **42**
III.3.2 Conception et évaluation du réseau : **43**
III.4- Conception des circuits de calcul pour implémentation sur FPGA : **48**
III.4. 1 Circuits de calcul parallèle : **48**
***III*.4.1.1 Circuit de calcul parallèle de la convolution :** 48

III.4.1.2 - Circuit de calcul parallèle de la normalisation (BN) et d'activation et ReLU 1 et Max pooling 1 : 51
III.4.1.c Le circuit de calcul parallèle de MaxPooling : 52
III.4.1.4 - Le circuit de calcul parallèle de la couche complètement connectées et de classification : 53
III.4.2 Circuits de calcul série : 53
III.4.2.1 Circuit de calcul série de la convolution : 54
III.4.2.2 - Circuit de calcul série de la normalisation (BN) et d'activation et ReLU 1 et Max pooling 1 55
III.4.2.3 Le circuit de calcul série de MaxPooling : 55
III.5 – Résultats d'implémentation des circuits sur FPGA : 56
III.5.1 Implémentation des circuits de calcul parallèle. 56
III.5.2 Implémentation des circuits de calcul série avec représentation des données en virgule flottante : 58
III.5.3 Implémentation des circuits de calcul série avec représentation des données en virgule fixe : 59
Conclusion générale 63
Bibliographie 65

Liste des figures

Figure I.1 : Relation entre l'intelligence artificielle, l'apprentissage automatique, l'apprentissage en profondeur et CNN.

Figure I.2 : Schéma synoptique de l'apprentissage supervisé.

Figure I.3 : Principe de l'apprentissage non supervisé.

Figure I.4 : Transformation d'un réseau ANN conventionnel en réseau RNN.

Figure I.5 : Principe de fonctionnement d'un réseau RNN.

Figure I.6 : Principe de fonctionnement d'un réseau GAN.

Figure I.7 : Structure de l'unité de base d'un réseau de neurones.

Figure I.8 : Architecture d'un Réseau Neurone Artificiel.

Figure I.9 : Architecture générale de réseau CNN.

Figure I.10 : Exemple de calcul de la convolution.

Figure I.11 : Exemple sur le pas de convolution. (stride).

Figure I.12 : Les graphes des fonctions d'activation**.**

Figure I .13: Les types de Couche pooling avec un filtre 2x2 et un pas de 2.

Figure I.14 : Couche entièrement connectée (Fully Connected layer).

Figure II.1 : Architecture interne d'un FPGA.

Figure II.2 : Configuration de base d'une cellule CLB d'un FPGA.

Figure II.3 : Structure d'un IOB de Xilinx sortie.

Figure II.4 : Distribution des blocs d'entrée /sortie (IOB) en banques.

Figure II.5 : Interconnexion interne d'un circuit FPGA.

Figure II.6 *:* Architecture simplifiée d'un bloc DSP48E**.**

Figure II.7 *:* Cellule de mémoire statique.

Figure II.8 : Architecture interne de la famille Virtex -7 de Xilinx.

Figure II.9 : Présentation du SoC Zynq-7000.

Figure II.10 : Illustration de l'architecture Xilinx UltraScale

Figure II.11 : L'environnement de visualisation de l'IDE de Vivado.

Figure III.1 : Exemples d'images de chaque classe de la base de données

Figure III.2 Illustration de la déformation introduite par le redimensionnent.

Figure III.3 : Exemples d'images de la base de données avant et après leur conversion en niveaux de gris.

Figure III.4 :l'architecture du réseau CNN proposé.

Figure III.5 : justesse et erreur de classification pour un réseau CNN sans couche de normalisation.

Figure III. 6 : justesse et erreur de classification pour un réseau CNN avec couche de normalisation.

Figure III.7 : Schéma fonctionnel du circuit de calcul parallèle correspondant à une couche de convolution.

Figure III.8 : Schéma détaillé d'une unité de convolution élémentaire conv_i_j.

Figure III.9 : schéma global du circuit de calcul parallèle de la deuxième couche de convolution.

Figure III.10 : Schéma fonctionnel du circuit de calcul parallèle de normalisation BN et d'activation ReLu.

Figure III.11 : Schéma fonctionnel du circuit de calcul parallèle de MaxPooling.

Figure III.12 : La structure du circuit de calcul de la couche complètement connectées

Figure III.13 : Schéma fonctionnel du circuit de calcul série correspondant à une couche de convolution.

Figure III.14 : Schéma fonctionnel du circuit de calcul série de normalisation BN et d'activation ReLu.

Figure III.15 : Schéma fonctionnel du circuit de calcul série de MaxPooling.

Figure III.16 : Représentation de la donnée en virgule flottante.

Figure III.17 : Représentation des données en virgule fixe.

Liste des tableaux

Tableau II.1 : Ressources disponibles pour La famille FPGA Virtex-7**.**

Tableau II.2 : Ressources disponibles pour La famille FPGA Zynq (Zynq -7000).

Tableau II.3 : Ressources disponibles pour La famille UltraScale FPGA.

Tableau III.1 : Nombre d'images par chaque classe de la base de données utilisées pour la classification de cinq types d'armes.

Tableau III.2 : Nombre d'entrées et nombre de sorties pour les couches de convolution du réseau proposé.

Tableau III.3 : Les fonctions Matlab utilisées pour la détermination des différentes couches du réseau CNN.

Tableau III .4 : Résumé des résultats de classification obtenu pour les différentes combinaisons de paramètres d'évaluation.

Tableau III.5 : Caractéristiques des circuits de calcul parallèle des différentes couches de convolution.

Tableau III.6 : Caractéristiques des circuits de calcul parallèle des différentes couches de normalisation BN.

Tableau III.7 : Caractéristiques des circuits de calcul parallèle des différentes couches de ReLu.

Tableau III.8 : Caractéristiques des circuits de calcul parallèle des différentes couches de MaxPooling.

Tableau III.9 : Caractéristiques des circuits de calcul série des différentes couches de convolution.

Tableau III.10 : Caractéristiques des circuits de calcul série des différentes couches de normalisation BN et d'activation ReLu.

Tableau III.11 : Caractéristiques des circuits de calcul série des différentes couches de MaxPooling.

Tableau III.12 : Ressources du circuit Virtex 7 XC7VX485T-2FFG1761 et puissance consommées par les différentes couches du circuit parallèle.

Tableau III.13 : Ressources du circuit Virtex 7 XC7VX485T-2FFG1761 consommées par différant couches de calcule en série (virgule flottant).

Tableau III.14 : Ressources du circuit Virtex 7 XC7VX485T-2FFG1761 et puissance consommées par les différentes couches de calcul en série avec les données en virgule fixe.

Liste des abréviations

AI	Artificielle Intelligence
AE	Auto Encoder
BRAM	Block Random Access Memory
BLE	Basic Logical Elements
CNN	Convolutional Neural Network
CPU	Central Processing Unit.
CONV	Convolutional Layer
CMOS	Complementary Metal Oxide Semiconductor
CLB	Configurable Logic Block
CPLD	Complex Logic Programmable Device
DL	Deep Learning
DSP	Digital Signal Processing
DCM	Digital Clock Manager
EPROM	Erasable programmable read only memory
FPGA	Field Programmable Gate Array
FC	Fully-connect Layer
FF	Flip flop
GPU	Graphics Processing Unit
IOB	Input Output Block
JTAG	Joint Test Action Group
LUT	Look-up Table
LSTM	Long Short-Term Memory
ML	Machine learning.
MUX	Multiplexer
NN	Neural Network
PAL	Programmable Array Logic
RNN	Recurrent Neural Networks
RBM	Restricted Boltzmann Machine
ReLU	Unités Rectifié linéaires

ReLU6	Rectified Linear Unit 6
RAM	Random Access Memory
ROM	Read Only Mémory
SOM	Self-organized map
SRAM	Static Random Access Memory
VGG	Visual Geometry Group
VHDL	Very high-speed Integrated Circuits Hardware Description Language
XSG	Xilinx System Generat

Introduction générale

Nous vivons dans l'air de l'intelligence artificielle où plusieurs taches qui ont été confiées uniquement à l'intelligence de l'humain sont réalisées facilement par la machine et parfois d'une façon meilleure. Le terme Intelligence Artificielle IA désigne le développement de systèmes informatiques qui peuvent exécuter des tâches qui nécessitent généralement l'intelligence humaine, comme la perception visuelle, la reconnaissance de la parole, la prise de décisions et la traduction du langage.
L'augmentation de la puissance de calcul et l'utilisation massive des appareils et systèmes électroniques de traitement (ordinateur de bureau, laptop, tablette, smartphone) ont permis de booster l'utilisation des techniques de l'intelligence artificielle dans le domaine de la vision par ordinateur.

Le Machine Learning (ML) est un sous-domaine de l'IA qui implique le développement d'algorithmes qui peuvent apprendre des données sans être explicitement programmés. Les algorithmes ML sont conçus pour améliorer leurs performances sur une tâche spécifique car ils sont exposés à plus de données.

Le Deep Learning (DL) est un type de ML qui utilise les réseaux neuronaux artificiels (ANN) avec plusieurs couches pour apprendre les représentations hiérarchiques des données. Les algorithmes DL sont particulièrement efficaces pour traiter de grands ensembles de données complexes et ont connu un succès remarquable dans un large éventail d'applications, y compris la classification des images, la reconnaissance de la parole, le traitement du langage naturel et les jeux.

Dans le contexte de la classification des armes, l'IA, le ML et le DL peuvent être utilisés pour analyser des images ou des vidéos d'armes et identifier automatiquement leur type et leurs caractéristiques. Les réseaux de neurones convolutifs CNN peuvent être utilisés pour extraire des caractéristiques des images et les classer en fonction de ces caractéristiques. Cette approche peut être utilisée pour une gamme d'armes, y compris les armes à feu, les couteaux, les explosifs et les pistolets.
Pour entrainer un réseau CNN à la classification des armes, un grand ensemble de données d'images ou de vidéos de différents types d'armes est nécessaire. Cet ensemble de données peut être étiqueté manuellement avec le bon type d'arme, et le CNN peut être formé en utilisant l'apprentissage supervisé pour classer les nouvelles images ou vidéos en fonction des caractéristiques apprises.
Les réseaux neuronaux CNN utilisent des couches de convolution pour extraire des caractéristiques des images, puis des couches entièrement connectées pour classer les images en fonction de ces caractéristiques.
L'implémentation des réseaux CNN sur des systèmes embarqués est un défi majeur en raison des contraintes matérielles strictes telles que les ressources de calcul limitées, la bande passante mémoire et la mémoire disponible sur puce. Les CNN nécessitent généralement des millions de paramètres et des milliards d'opérations pour traiter une seule image en entrée. Pour surmonter ces obstacles, les FPGA (Field Programmable Gate Array) sont devenus une solution alternative attrayante en raison de leurs performances relativement élevées, de leur flexibilité, de leur efficacité énergétique et de leur cycle de développement rapide par rapport aux CPU et aux GPU. Cependant, les FPGA sont limités en ressources, ce qui nécessite de trouver une correspondance appropriée entre le modèle de CNN et le modèle d'implémentation.

Dans ce contexte, l'objectif de notre projet de fin d'étude est de faire une implémentation efficace sur circuit FPGA d'un système de classification de cinq types d'armes en utilisant les réseaux CNN.

Ce présent document est divisé en trois chapitres :

Le premier chapitre est consacré à la présentation des concepts de base et des notions importantes qui sont en relation avec l'Intelligence Artificielle, le Machine Learning et le Deep Learning (définition, architectures, types ...etc.). Une étude détaillée sur les réseaux CNN qui est la base de notre projet est fournie dans la dernière partie de ce chapitre.

Le deuxième chapitre est réservé à la présentation des circuits logiques programmables FPGA et l'environnement matérielle et logicielle de leurs développements. Nous commençons par donner les éléments de base nécessaires à la compréhension du fonctionnement des circuits FPGA (définition, l'architecture interne des circuits FPGAs, principaux composants constituant un circuit FPGA, familles ...). Ensuite, présentons les logiciels de base utilisés pour le développement de notre application sur FPGA. A la fin du deuxième chapitre nous abordons les défis de la mise en œuvre des réseaux CNN sur les circuits FPGA.

Dans le troisième chapitre, nous présentons les différentes phases aboutissants à l'implémentation du réseau CNN de classification d'armes sur un circuit FPGA. En premier lieu, nous expliquons l'étape de préparation de la base de données, ensuite l'étape de choix de l'architecture convenable du réseau CNN qui permet d'assurer un bon compromis entre simplicité et justesse de classification. Ensuite, nous présentons les différentes conceptions des circuits de calcul des différentes couches du réseau CNN. Dans le dernier volet du troisième chapitre nous présentons les résultats d'implémentation des différents circuits de calcul sur le circuit FPGA de type Virtex 7.

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

Chapitre I :

De l’Intelligence Artificielle aux réseaux CNN

I.1 Introduction :

Ce chapitre intitulé "De l'Intelligence Artificielle à réseaux CNN" propose une exploration des domaines de l'intelligence artificielle comme l'apprentissage automatique (Machine Learning), l'apprentissage en profondeur (Deep Learning) avec une description détaillée sur les différents types de DL. L'accent est mis sur les réseaux de neurones, en particulier les réseaux de neurones à convolutions CNN, avec une analyse approfondie de chaque couche et de ses paramètres dans l'architecture des réseaux CNN. Le chapitre explore également le rôle des réseaux CNN dans l'analyse de données visuelles complexes pour diverses applications, y compris la classification d'images.

I.2 Intelligence Artificielle :

L'IA est un processus qui vise à imiter l'intelligence humaine en utilisant des algorithmes exécutés dans un environnement informatique dynamique. Son objectif est de permettre aux ordinateurs de penser et d'agir de manière similaire à un être humain. Pour cela, une application d'IA nécessite trois composantes essentielles : un système informatique puissant, une grande quantité de données et des algorithmes avancés. [1]

Comme indiqué par la figure I.1, l'IA est un domaine complet qui englobe plusieurs sous-domaines tels que le calcul évolutif, les systèmes experts, l'IA symbolique, l'apprentissage automatique, l'apprentissage en profondeur, l'apprentissage par renforcement, et bien d'autres encore. [2]

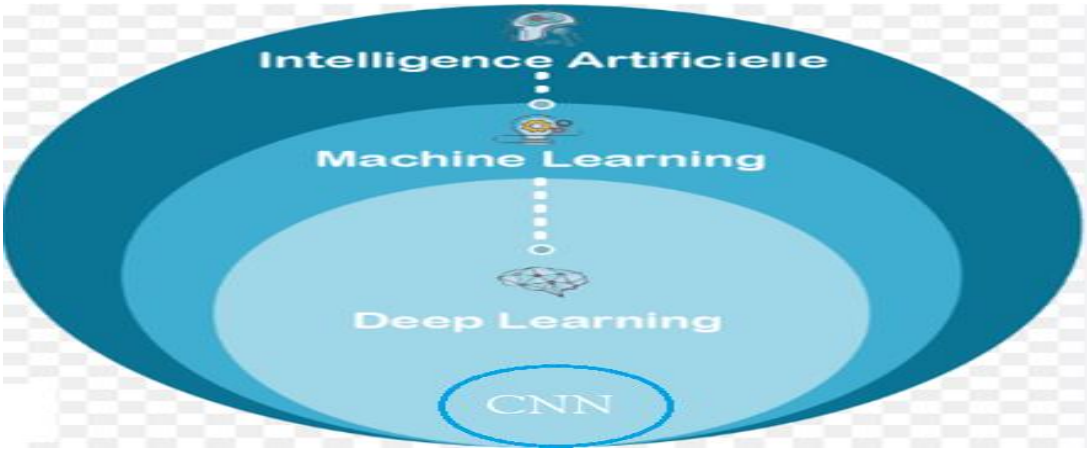


Figure I.1 : *Relation entre l'intelligence artificielle, l'apprentissage automatique, l'apprentissage en profondeur et les réseaux CNN.*

I.2.1 Domaines d'applications de l'Intelligence Artificielle :

L'intelligence artificielle a connu un développement parallèle avec le calcul numérique plutôt que de se positionner en concurrence avec lui. De ce fait, elle s'intéresse aux domaines d'application considérés comme complexes ou même insolubles par le calcul numérique. Parmi ces domaines, on peut citer notamment : le calcul formel, l'analyse de texte et d'images dans le domaine de la vision, la génération de plans dans la robotique, la perception, l'interprétation, la décision et l'action dans les machines autonomes, la traduction, la compréhension et la synthèse dans le langage naturel, la démonstration de théorèmes, les jeux, ainsi que la représentation des connaissances dans diverses disciplines telles que le droit, la médecine. [3]

I.3 Machine Learning :

L'apprentissage automatique est un sous-ensemble de l'intelligence artificielle (IA). Il vise à apprendre aux ordinateurs à apprendre des données et à s'améliorer avec l'expérience - au lieu d'être explicitement programmés pour le faire. Chaque fois qu'un moteur de recherche Web comme Google est utilisé pour rechercher sur Internet, l'une des raisons qui lui permet de fonctionner si bien est l'utilisation d'un algorithme d'apprentissage qui a appris à classer les pages Web. Les algorithmes du machine learning sont utilisés à diverses fins comme l'exploration de données, le traitement d'images, l'analyse prédictive, etc. pour n'en nommer que quelques-unes. Le principal avantage de l'apprentissage automatique est que, une fois qu'un algorithme apprend quoi faire avec les données, il peut faire son travail automatiquement. [4]

I.3.1 Types d'apprentissage automatique :

I.3.1.1 Apprentissage Supervisé :

L'apprentissage supervisé, qui est une tâche d'apprentissage automatique consistant à apprendre une fonction reliant une entrée à une sortie à partir d'un ensemble de données étiquetées. Les algorithmes d'apprentissage supervisé nécessitent une assistance externe et l'ensemble de données d'entrée est divisé en ensemble de données de d'apprentissage « training » et d'essai « testing ». Les algorithmes apprennent des modèles à partir des données d'apprentissage et les appliquent aux données d'essai pour la prédiction ou la classification. Le schéma synoptique des algorithmes d'apprentissage automatique supervisés est donné à la figure ci-dessous. [4].

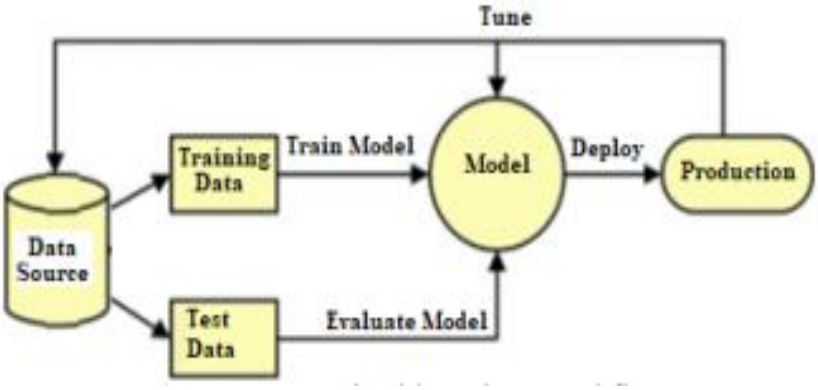


Figure I.2 : *Schéma synoptique de l'apprentissage supervisé [4]*

I.3.1.2 .Apprentissage semi-supervisé :

Ce type d'apprentissage forme un modèle d'apprentissage sur un ensemble de données partiellement annoté contenant des données annotées et beaucoup de données non annotées. L'idée est de faire correspondre les annotations en fonction de la similarité entre les données annotées et non annotées.

L'apprentissage semi-supervisé est ainsi classé entre l'apprentissage supervisé utilisant uniquement des données annotées et l'apprentissage non supervisé utilisant uniquement des données non annotées. La combinaison de ces deux ensembles de données peut grandement améliorer les résultats sans recourir à des interventions d'annotation manuelles fastidieuses (coûteuses et chronophages).

I.3.1.3 Apprentissage Non Supervisé :

Dans ce type d'apprentissage, on dispose d'une base de données non étiquetées et on cherche à les grouper en classes appelées cluster. Chaque cluster est constitué d'objets qui présentent une forte similitude entre eux (similarité intra-classes) et une très faible similitude avec les objets des autres clusters (similarité interclasses). La machine essaie de structurer et de trier les données saisies selon certaines caractéristiques. Par exemple, une machine pourrait apprendre que des pièces de différentes couleurs peuvent être triées selon la couleur [5].

La figure I.3 illustre le principe de l'apprentissage non supervisé.

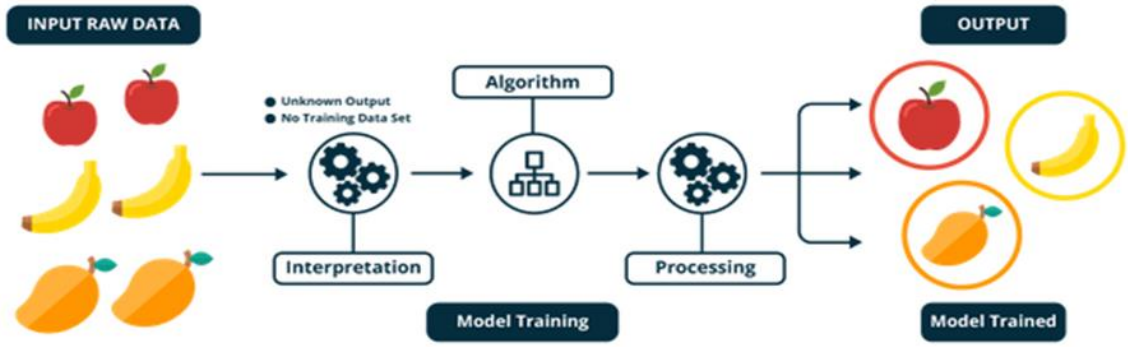


Figure I.3 : *principe de l'apprentissage non supervisé*

I.3.1.4-Apprentissage par Renforcement :

Cette méthode d'apprentissage vise à utiliser les observations des interactions avec l'environnement pour exécuter des actions qui maximisent la récompense ou minimisent les risques pour créer des programmes intelligents (également appelés agents). [4]

I.4 Le Deep Learning :

L'apprentissage profond ou Deep Learning DL est une technique d'apprentissage automatique qui comprend trois types d'apprentissage : supervisé, semi-supervisé et non supervisé. Elle utilise des réseaux de neurones artificiels avec des fonctions d'activation pour produire des sorties non linéaires. Cette méthode est inspirée de la structure neuronale du cerveau humain. Le Deep Learning permet aux machines de s'entraîner à effectuer des tâches que les humains peuvent effectuer intuitivement et presque automatiquement, mais sont par ailleurs très difficiles pour les ordinateurs.

Le mot Deep signifie qu'entre la première couche ou couche d'entrée et la dernière couche ou couche de sortie, nous avons un ensemble de couches cachées. Dans le Deep Learning, un réseau peut consommer une grande quantité de données d'entrée, puis les traiter à travers plusieurs couches grâce auxquelles nous pouvons apprendre des caractéristiques complexes des données.

L'apprentissage profond est de plus en plus populaire et a été utilisé pour des applications telles que la reconnaissance d'images, la récupération d'images, les moteurs de recherche et le traitement du langage naturel. Ses avantages comprennent la capacité à apprendre des caractéristiques abstraites, une complexité de formation réduite, une précision prometteuse, une gestion de grands ensembles de données. [6]

I.4.1 Exemples de techniques du Deep Learning :

I.4.1.1 Réseaux Neuronaux Récurrents (RNN) :

Le réseau de neurones récurrent (RNN) est une classe d'apprentissage en profondeur basée sur les travaux de David Rumelhart en 1986. Les RNN sont connus pour leur capacité à traiter et à obtenir des informations à partir de données séquentielles. L'analyse vidéo, le sous-titrage d'images, le traitement du langage naturel (NLP) et l'analyse musicale sont des exemples types de l'application des réseaux RNN. Contrairement aux réseaux de neurones standards qui supposent l'indépendance entre les points de données, les RNN capturent activement les dépendances séquentielles et temporelles entre les données.

Les réseaux RNN fonctionne sur le principe de sauvegarder l'état actuel des couches cachées et de la renvoyer à l'entrée afin de prédire les sorties.

La figure I.4 montre comment est transformé un réseau conventionnel en réseau RNN. Les nœuds des différentes couches cachées du réseau neuronal sont compressés pour former une seule couche du réseau RNN.

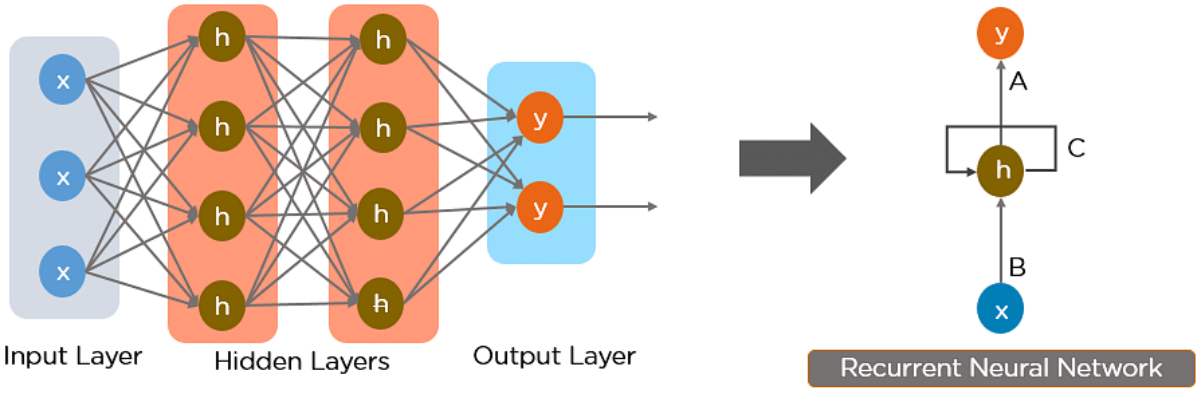


Figure I.4 : *Transformation d'un réseau ANN conventionnel en réseau RNN.*

"x" est la couche d'entrée, "h" est la couche cachée et "y" est la couche de sortie. A, B et C sont les paramètres de réseau utilisés pour améliorer la sortie du modèle.

A tout instant t, l'état courant du réseau est obtenu par une combinaison de l'entrée actuelle x(t) et de l'état précédent du réseau h(t-1) selon l'équation :

$$h(t) = f(h(t-1), x(t)) \qquad \textbf{I.1}$$

Où f est une fonction qui dépend de deux paramètres A et B.

La sortie actuelle du réseau y(t) est déterminée en fonction de l'état actuel du réseau h(t) et du paramètre A comme le montre la figure I.5 [7], [8].

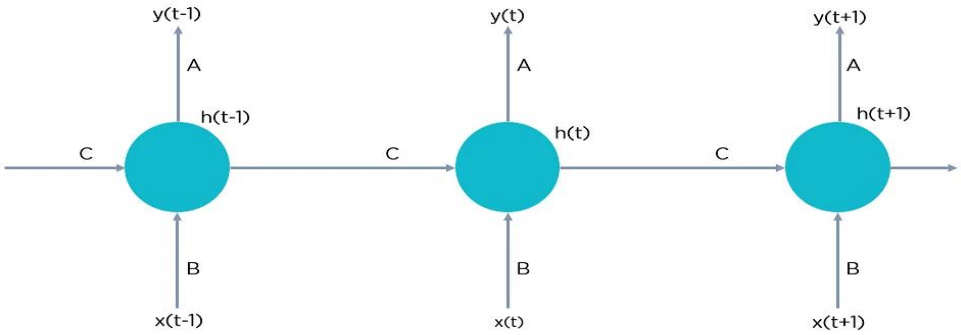


Figure I.5 : *principe de fonctionnement d'un réseau RNN.*

I.4.1.2 Réseaux Adversarial Génératifs (GAN) :

Les réseaux GAN ont été introduits pour la première fois par Ian Goodfellow et d'autres de l'Université de Montréal en 2014. Les GAN sont capables d'imiter n'importe quelle distribution de données dans n'importe quel domaine : images, musique, discours et prose. Les GAN sont un exemple de réseau qui utilise l'apprentissage non supervisé pour faire apprendre deux modèles en parallèle. Un aspect clé des GAN est la façon dont ils utilisent un nombre de paramètres nettement inférieur à la normale par rapport à la quantité de données sur lequel le réseau est entraîné. Le réseau est obligé de représenter efficacement les données d'entraînement, ce qui le rend plus efficace pour générer des données similaires aux données d'entraînement.

Un réseau GAN est composé d'un discriminateur D, et d'un générateur G, qui fonctionnent en parallèle selon la structure de la figure I.6. L'objectif du générateur est de pouvoir créer une fausse sortie qui ressemble à une sortie réelle, le générateur s'entraînant à travers ses interactions avec le discriminateur et non à partir d'un contenu réel. L'objectif du générateur est de produire une sortie si proche de la réalité qu'elle confond le discriminateur en ce qu'il est capable de différencier les fausses données des vraies données.

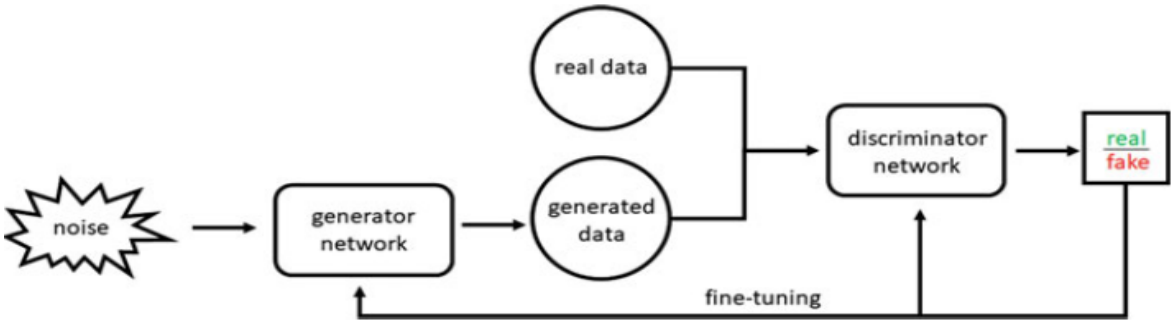


Figure I.6 : *Principe de fonctionnement d'un réseau GAN*

Il y a trois étapes dans la procédure de fonctionnement d'un réseau GAN qui sont :

- Tout d'abord, le générateur prend des nombres aléatoires et renvoie une image. Cette image générée est introduite dans le discriminateur avec un flux d'images extraites de l'ensemble de données réel.

- Deuxièmement, le discriminateur prend à la fois des images réelles et fausses et renvoie des probabilités ; une sortie proche de 0 étant la les données du générateur sont fausses et une sortie proche de 1 indiquant que les données sont réelles.

- Troisièmement, le réseau discriminateur fournit une rétroaction au générateur afin de l'entraîner et d'améliorer sa sortie.

c. Réseaux Neuronaux Convolutifs (CNN) :

Une architecture CNN typique comprend généralement des couches alternées de convolution et de mise en commun, suivies d'une ou plusieurs couches entièrement connectées à la fin. Dans certains cas, une couche entièrement connectée est remplacée par une couche de mise en commun de la moyenne globale. En plus des différentes fonctions de mappage, différentes unités de régulation telles que la normalisation et le dropout des lots sont également incorporées pour optimiser les performances du CNN. La disposition des composants du CNN joue un rôle fondamental dans la conception de nouvelles architectures et l'obtention de meilleures performances. Cette section aborde brièvement le rôle de ces composants dans une architecture CNN.

d. LSTM networks :

Les réseaux LSTM (Long Short-Term Memory) sont une variante des réseaux de neurones récurrents (RNN) qui sont conçus pour modéliser les dépendances à long terme dans les données séquentielles. Ils sont composés de blocs de mémoire qui contiennent des cellules de mémoire avec des auto-connexions, ainsi que des portes qui contrôlent le flux d'information. Les LSTM ont été particulièrement efficaces dans la modélisation de séquences de données telles que la parole, le texte et la musique. [9].

I.4.2 Domaines d'application de l'apprentissage profonde :

L'apprentissage profond est appliqué dans divers domaines tels que l'informatique, la robotique, la bio-informatique et l'éducation assistée par ordinateur et l'intelligence artificielle. L'éventail d'applications du Deep Learning est très large on peut citer à titre d'exemple :

- ***La reconnaissance faciale*** : Un algorithme de Deep Learning apprend à détecter sur une photo les caractéristiques du visage tels que : les yeux, la bouche, le nez...etc. Et cela en fournissant à l'algorithme un ensemble d'images du visage qui vont être utilisées pour l'entraînement, et à force de les entraîner, il sera capable de détecter un visage sur une image.

- ***La détection d'objets :*** Les algorithmes de détection d'objets sont désormais capables d'identifier un objet ou une personne dans une image qui contient plusieurs éléments (images complexes) jusqu'au niveau du pixel.

- ***La Traduction automatique*** : La traduction automatique a fait de grands progrès grâce à l'utilisation des réseaux de neurones. Cependant, ce type d'IA nécessite la traduction humaine de grandes quantités de contenu à l'avance. Il s'agit d'un apprentissage supervisé, où la machine fait une supposition et obtient ensuite la bonne réponse d'un humain, ce qui lui permet d'ajuster son traitement en conséquence. Cette méthode est efficace pour les langues largement parlées, comme l'anglais ou le français, pour lesquelles il existe de nombreux documents parallèles.

- ***La classification d'images*** : Reconnaître des images en fonction de ce qu'elles représentent (par exemple des voitures, des personnes) est une tâche difficile pour l'ordinateur, qui doit d'abord effectuer une analyse pour extraire des caractéristiques de chaque image. Avec le deep learning, les

informations d'entrée sont analysées couche par couche, le résultat d'analyse d'une couche est envoyé à la couche suivante. Les informations ainsi collectées sont cartographiées dans un algorithme flexible, grâce auquel l'ordinateur peut déterminer en quelques opérations si l'image appartient à la catégorie d'une voiture ou d'une personne. [10], [11]

I.5 Réseaux de neurones artificiels :

Les réseaux de neurones artificiels ont été proposés comme modèles Mathématiques de la fonction physiologique des neurones. Le modèle le plus simple de l'unité de base du réseau est un perceptron linéaire. C'est une cellule dans le réseau de neurones artificiel, chaque neurone calcul sa sortie comme une somme pondérée des entrées, décalée d'un biais et suivie d'une fonction d'activation. Un réseau de neurones est composé de plusieurs perceptrons fortement liées par des liens avec des pondérations différents.

La structure de l'unité de base d'un réseau de neurones est illustrée dans la figure ci-dessous :

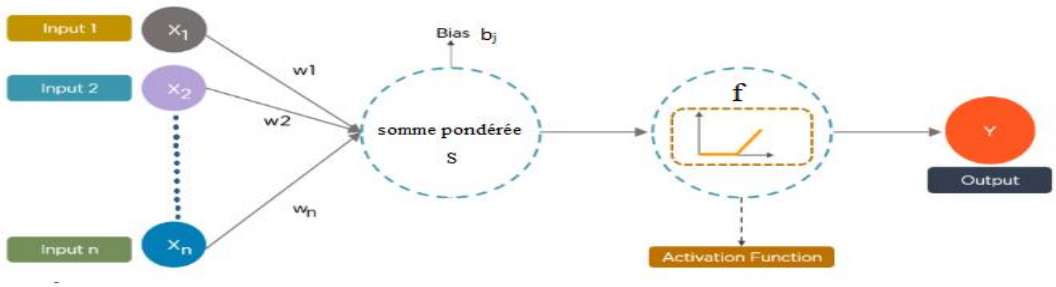


Figure I.7 : *Structure de l'unité de base d'un réseau de neurones.*

La structure du réseau comprend les éléments suivants [12] :

1- *N* entrées notées x_1, x_2, ...x_n: ces entrées peuvent être issue soit de l'entrée totale du réseau (si le neurone se trouve dans la couche d'entrée) soit des sorties d'autres neurones des couches précédentes.

2- Un vecteur de poids notés : W_{j1}, W_{j2}, ... W_{jn} et d'une valeur de biais notée b_j : les poids et le biais sont utilisés pour calculer la somme pondérée et biaisée des entrées.

3-Une fonction d'activation : cette fonction est utilisée pour calculer la sortie finale du neurone.

On distingue deux phases :

La somme pondérée et biaisée des entrées est donnée par :

$$s=\sum_{i=1}^{n}(W_i.X_i + bj) \qquad \textbf{I.2}$$

La sortie du neurone est donnée par :

$$Y = f(\sum_{i=1}^{r}(W_i.X_i + bj)). \qquad \textbf{I.3}$$

Où *f* est la fonction d'activation du réseau.

I.5.1 Architecture des réseaux de neurones artificiels :

Un réseau de neurone est constitué d'une couche d'entrée, d'une couche de sortie et d'une ou plusieurs couches intermédiaires appelées couches cachées.

-***Couche d'entrée*** : La couche d'entrée est la première couche du réseau neuronal à recevoir des données d'entrée. Le nombre de neurones dans la couche d'entrée est égal au nombre de caractéristiques dans les données d'entrée.

-***Couches cachées*** : Les couches cachées sont les couches intermédiaires entre les couches d'entrée et de sortie. Ils sont appelés cachés parce que leurs valeurs ne sont pas observées dans les données d'entrée ou de sortie. Le nombre de couches cachées et le nombre de neurones dans chaque couche cachée est un paramètre d'hyperlien qui doit être ajusté.

-***Couche de sortie*** : La couche de sortie est la dernière couche du réseau neuronal, qui produit la sortie. Le nombre de neurones dans la couche de sortie dépend du problème. Par exemple, si le problème est un problème de classification binaire, il y aura un neurone dans la couche de sortie, et si le problème est un problème de classification multi-catégories, il y aura plusieurs neurones dans la couche de sortie [13]

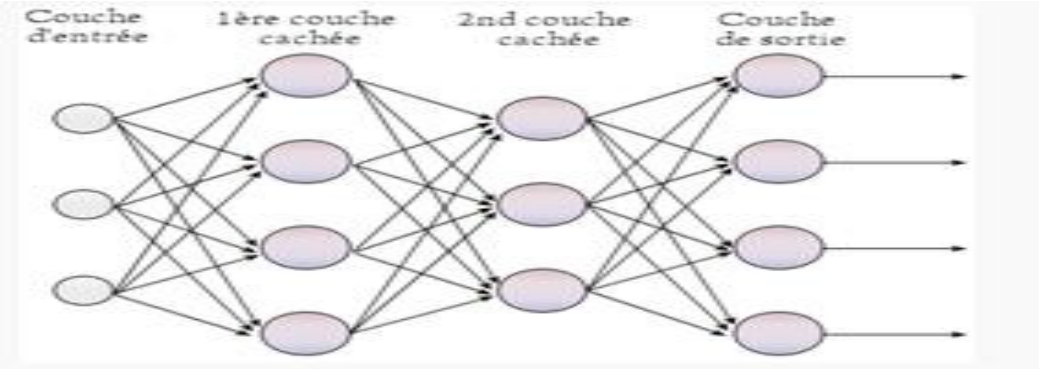


Figure 1.8 : *Architecture d'un Réseau de Neurones Artificiel*

I.6 Réseaux de neurones convolutif CNN :

Nous allons maintenant nous concentrer sur le concept et les différents modules couramment utilisés dans les réseaux CNN et les fonctions d'activation.

Les réseaux de neurones **convolutionnels** (CNN) sont un type particulier de réseau de neurones artificiels qui sont principalement utilisés pour le traitement de l'image et de la vision par ordinateur. Les couches de convolution, qui utilisent des opérations de convolution, sont la clé du fonctionnement des CNN, permettant de détecter des motifs et des caractéristiques dans les images. Les CNN ont été utilisés avec succès dans une variété d'applications, notamment la reconnaissance d'images, la classification d'images et la segmentation d'images. [14]

I.6.1 Architecture des réseaux CNN :

La figure I.9 représente l'architecture de base d'un réseau CNN qui est composée de la succession de plusieurs couches. Chaque couche du réseau effectue une opération bien déterminée. Les opérations réalisées par les différentes couches sont détaillées dans le prochain paragraphe.

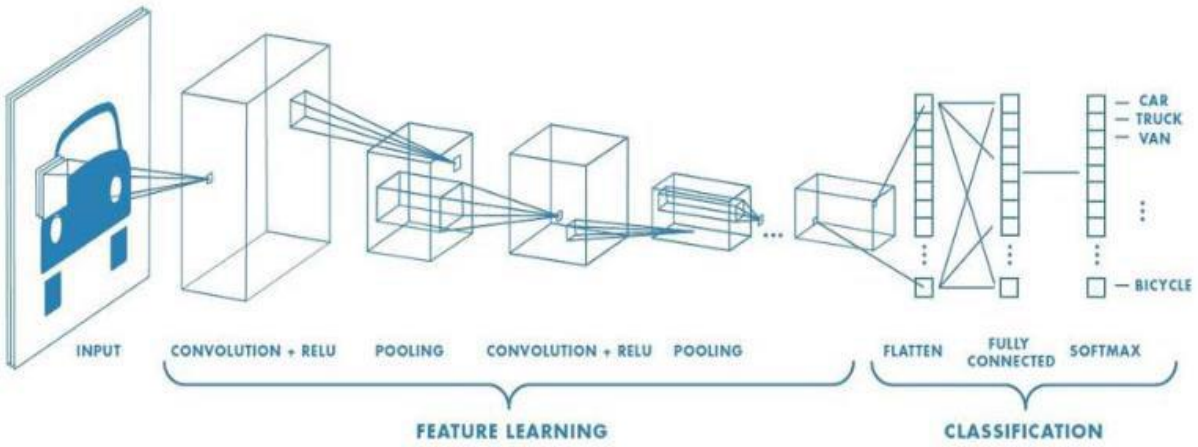


Figure I.9 : *Architecture générale de réseau CNN.*

I.6.1.1 Couche d'entrée :

Les données d'entrée sont les images à classer et chaque image est donnée par les valeurs de ces pixels dans une matrice de dimensions *[n m k],* où *n* est la hauteur de l'image, *m* est la largeur de l'image et *k* est le nombre de canaux d'image. Typiquement *k*=3 pour les images couleur RVB et *k=1* pour les images en niveaux de gris.

I.6.1.2 Couches de convolution :

La première couche des réseaux de neurones convolutifs est presque toujours constituée d'une couche de convolution, qui est l'élément essentiel de ces réseaux. Sa fonction principale est de détecter la présence d'un ensemble de caractéristiques dans les images d'entrée en effectuant un filtrage par convolution. Ce processus implique de "faire glisser" un filtre (appelé noyau ou Kernel) représentant la caractéristique sur l'image et de calculer le produit de convolution entre le noyau et chaque partie de l'image balayée. [15]

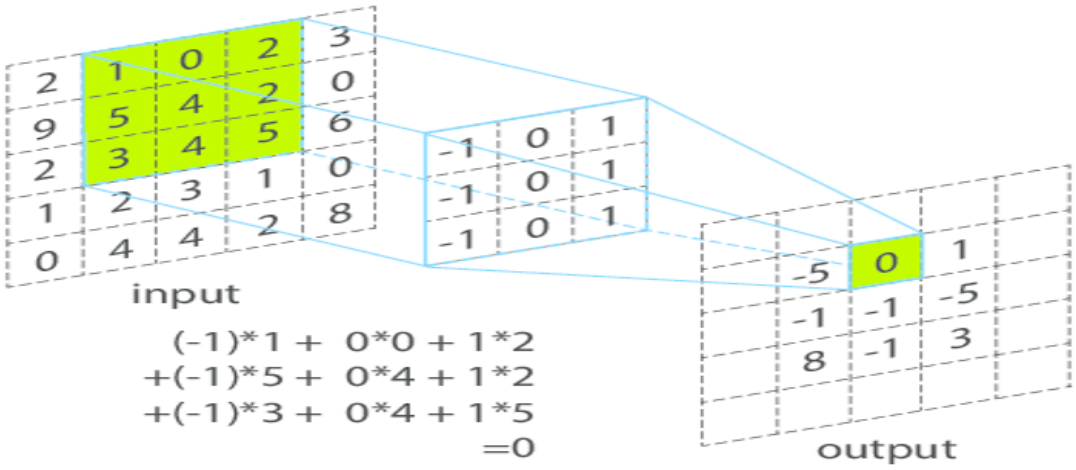


Figure I.10 : *Exemple de calcul de la convolution.*

Le Kernel est un réseau de valeurs ou de nombres séparés, dont chacun est appelé poids du noyau. Les valeurs de poids sont déterminées par le processus d'apprentissage. Dans la plupart des applications, le Kernel est une matrice carrée : une matrice 3x3 est courante, mais parfois une matrice 5x5 ou 7x7 est utilisée.

La couche de convolution est caractérisée par trois paramètres importants [15] :

-La profondeur de la couche ou le nombre d'entrées : qui correspond au nombre de noyaux de convolution utilisés pour calculer la même carte de caractéristiques de sortie.

-Le pas de convolution (stride) : qui est la distance à laquelle le filtre se déplace d'une position à la suivante. Par exemple, si le pas est de 1, le noyau de convolution sera déplacé d'un seul pixel vers la droite pour chaque mouvement horizontal du noyau et d'un seul pixel vers le bas pour chaque mouvement vertical du noyau lors du calcul d'une matrice de sortie. Le stride contrôle le chevauchement au niveau des cartes de caractéristiques de sortie

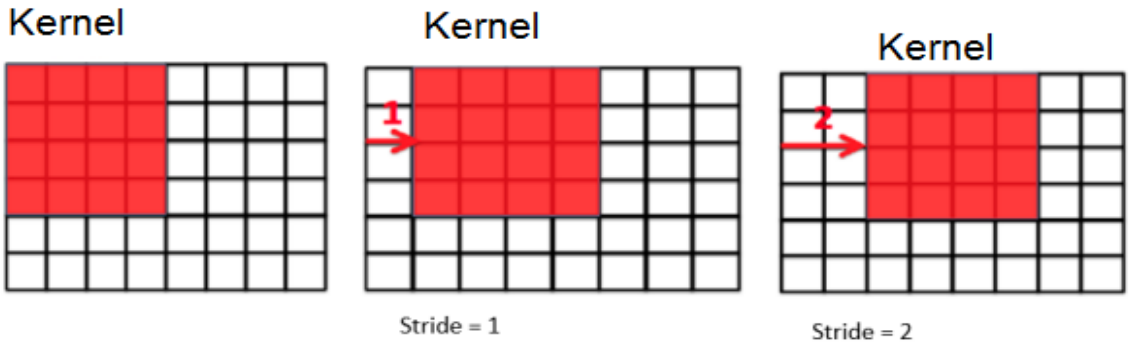


Figure I.11 : *Exemple sur le pas de convolution (stride).*

-le nombre de sortie : Correspond au nombre de maps de caractéristiques calculées par la couche de convolution. Ce nombre est égale au nombre de groupes de filtres de la couche de convolution.

I.6.1.3 Batch Normalization (Normalisation par lots) :

La Batch Normalization (normalisation par lots) est une technique de normalisation des entrées d'un réseau de neurones artificiels qui a été proposée en 2015 par Ioffe et Szegedy et qui permet d'accélérer l'apprentissage. Elle consiste à normaliser les activations d'une couche en soustrayant la moyenne et en divisant par l'écart type des activations dans un mini-lot d'exemples d'entraînement. De plus, la normalisation par lots agit comme une régularisation et peut réduire le besoin d'autres formes de régularisation, telles que la désactivation aléatoire et le lissage des poids [16]

L'équation de la batch normalization peut être exprimée comme suit :

$$\varphi = \frac{x_i - \mu_B}{\sqrt{{\sigma_B}^2 + \epsilon}} * \gamma + \beta \qquad \textbf{I.4}$$

x_i: est l'activation d'un neurone donné dans un batch d'exemples d'entraînement.

μ_B: est la moyenne des activations du batch.

${\sigma_B}^2$: La variance est le carré de l'écart type (σ_B) et représente la dispersion des valeurs par rapport à la moyenne.

ϵ: est une petite valeur (généralement près de zéro) ajoutée pour la stabilité numérique.

(γ, β) : Sont des paramètres appris lors de l'entraînement du modèle. Ces paramètres permettent au modèle de décider de l'échelle et du décalage optimaux pour les activations normalisées.

I.6.1.4 Les fonctions d'activations :

Les couches de fonction d'activation sont utilisées pour traiter la sortie de chaque couche de convolution dans un réseau de neurones convolutifs. La fonction d'activation appliquée sur la carte de caractéristiques produite par la couche de convolution génère une carte d'activation en sortie. La couche d'activation accélère le processus d'apprentissage des cartes d'entités en introduisant une combinaison non linéaire d'entités. En général, Trois propriétés sont attendues de la fonction d'activation [17] :

-Non linéarité **:** est une caractéristique essentielle de la fonction d'activation, permettant au réseau de neurones de traiter efficacement des problèmes qui ne sont pas linéaires.

-Différentiable **:** La propriété de différentiabilité de la fonction d'activation signifie qu'elle a une dérivée continue du premier ordre. Cette propriété est souhaitable car elle permet l'utilisation de méthodes d'optimisation basées sur un gradient, telles que la retro-propagation. pour entraîner efficacement le réseau de neurones.

-Monotone : La propriété de monotonie de la fonction d'activation aide le réseau de neurones à converger plus facilement vers un modèle plus précis.

Les types de fonctions d'activation les plus couramment utilisés dans les réseaux CNN et d'autres réseaux de neurones profonds sont :

I.6.1.4.1 La fonction Sigmoïde :

La fonction d'activation reçoit des nombres réels en entrée et renvoie des valeurs limitées entre zéro et un. La courbe de la fonction sigmoïde a une forme en S et peut être définie mathématiquement par [18]:

$$f(x)_{\text{sigm}} = \frac{1}{1+e^{-x}} \qquad \textbf{\textit{I.5}}$$

I.6.1.4.2 La fonction Tanh :

La fonction Tanh (Tangente hyperbolique) est une fonction mathématique couramment utilisée dans les réseaux de neurones artificiels pour introduire une non-linéarité. Elle est similaire à la fonction sigmoïde, mais elle renvoie des valeurs bornées entre -1 et 1. La fonction Tanh est définie comme suit : [18]

$$f(x)_{\textbf{Tanh}} = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}} \qquad \textbf{\textit{I.6}}$$

I.6.1.4.3 La fonction ReLu :

C'est la fonction la plus couramment utilisée dans le contexte CNN. Elle convertit les valeurs entières de l'entrée en nombres positifs. L'utilisation de la fonction ReLu accélère le processus d'entrainement. Une charge de calcul plus faible est le principal avantage de ReLu par rapport aux autres. [17]

Le calcul du gradient est très simple (0 ou 1 selon le signe de x). Tous les éléments négatifs sont mis a 0, Donc, cette fonction est facilement implémentée dans le matériel. De plus, le calcul d'un ReLu est facile. Sa représentation mathématique est :

$$f(x)_{\mathbf{ReLu}} = \max(0, x) \qquad \textbf{I.7}$$

I.6.1.4.4 La Fonction ReLu6 :

La fonction ReLU6 (Rectified Linear Unit 6) est une fonction d'activation utilisée dans les réseaux de neurones artificiels. Elle est similaire à la fonction ReLU (Rectified Linear Unit), mais elle limite les valeurs en sortie à 6. La fonction ReLU6 est définie comme suit :

$$f(x)_{\mathbf{ReLU6}} = \min(\max(0, x), 6) \qquad \textbf{I.8}$$

La fonction ReLU6 renvoie une valeur de 0 lorsque l'entrée est négative et une valeur de 6 lorsque l'entrée est supérieure à 6. Elle présente une pente constante de 1 pour les valeurs positives de l'entrée, ce qui en fait une fonction d'activation efficace pour les réseaux de neurones profonds. La fonction ReLU6 est souvent utilisée dans les réseaux de neurones pour la classification d'images et la reconnaissance d'objets. [18]

I.6.1.4.5 La Fonction LeakyReLu :

La fonction LeakyReLU est définie comme suit :

$$\begin{cases} f(x) = x \text{ si } x > 0 \\ f(x) = ax \quad Ailleurs \end{cases} \qquad \textbf{I.9}$$

(Où a est une petite constante positive généralement 0,01)

Par rapport à la fonction standard ReLU, qui a un gradient de 0 pour toutes les entrées négatives, la fonction Leaky ReLU fournit un gradient non nul, qui peut aider à prévenir les "neurones morts" dans le réseau, où un neurone produit toujours 0 en raison d'un biais négatif. Cela peut améliorer les performances et la vitesse de formation du réseau.

Le choix de la constante "a" dans la fonction Leaky ReLU peut affecter les performances du réseau, et il est généralement choisi par l'expérimentation et l'accord.[19]

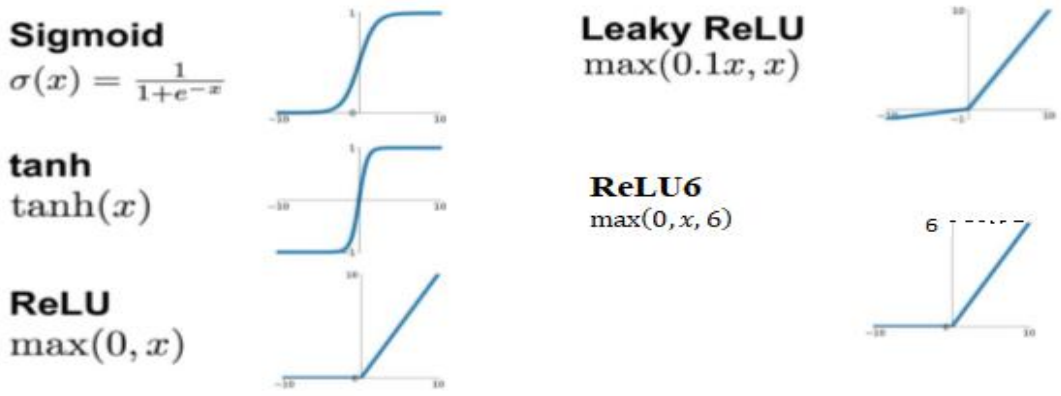


Figure I.12 : *Les graphes des fonctions d'activation.*

I.6.1.5 Couche sous-échantillonnage (Pooling) :

Une couche de sous-échantillonnage, ou couche de pooling, est une couche couramment utilisée dans les réseaux de neurones convolutionnel (CNN) pour réduire la taille spatiale des données en entrée et en préservant les informations les plus importantes. Cela permet de réduire le nombre de

paramètres à entraîner, de diminuer la sensibilité aux petites variations dans les données en entrée, et de générer des représentations plus compactes des données.[20]

Il existe deux types de Pooling :

-Le Max-Pooling : est le type de pooling le plus fréquemment utilisé. Il consiste à sélectionner le pixel ayant la valeur maximale dans le filtre, comme illustré dans la figure I.13.

- Le Mean-Pooling : est une fonctionnalité similaire au Max Pooling, mais elle fonctionne en extrayant des patchs à partir des cartes d'entités d'entrée, en calculant la valeur moyenne de chaque patch et en ignorant toutes les autres valeurs. En d'autres termes, cette technique consiste à diviser la carte d'entités en patchs, puis à calculer la moyenne de chaque patch pour obtenir une carte d'entités sous-échantillonnée [16] comme illustré dans la figure I.13.

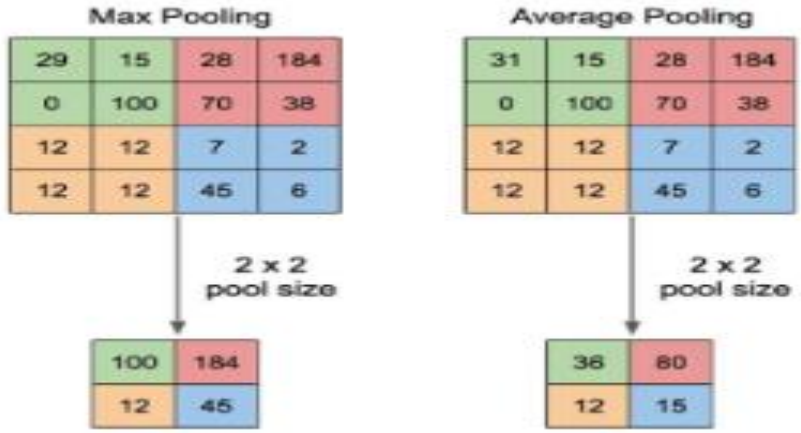


Figure I.13 *: les types de Couche pooling avec un filtre 2x2 et un pas de 2*

I.6.1.6 Couches complètement connectées (Fully Connected FC) :

En général, la couche complètement connectée (ou FC) est placée à la fin de chaque architecture de réseau de neurones convolutionnel (CNN). Chaque neurone dans cette couche est connecté à tous les neurones de la couche précédente, suivant le principe de base du réseau neuronal multicouche conventionnel. L'entrée de la couche FC provient généralement de la dernière couche de pooling ou de convolution.

Pour obtenir l'entrée de la première couche complètement connectée, on applique une transformation matricielle qui convertit toutes les matrices représentant la sortie de la couche précédente en vecteurs, puis on concatène tous les vecteurs pour obtenir un vecteur unique. Les poids de chaque couche complètement connectée sont représentés sous forme d'une matrice unique à deux dimensions, dont une dimension représente le nombre d'entrées de la couche et l'autre dimension représente le nombre de sorties de la couche. La sortie d'une couche complètement connectée est obtenue en faisant le produit matrice-vecteur de la matrice des poids et du vecteur des caractéristiques d'entrée, suivie de l'ajout d'un vecteur de biais.

Les sorties des couches de convolution et de pooling représentent des caractéristiques de haut niveau de l'image d'entrée. L'objectif de la couche complètement connectée est d'utiliser ces caractéristiques pour classer l'image d'entrée du réseau en différentes classes en fonction de la base de données d'apprentissage. Bien que la plupart des cartes de caractéristiques des couches de

convolution et de pooling puissent être utiles pour la tâche de classification, une combinaison de ces fonctionnalités peut encore améliorer les performances de classification. [21]

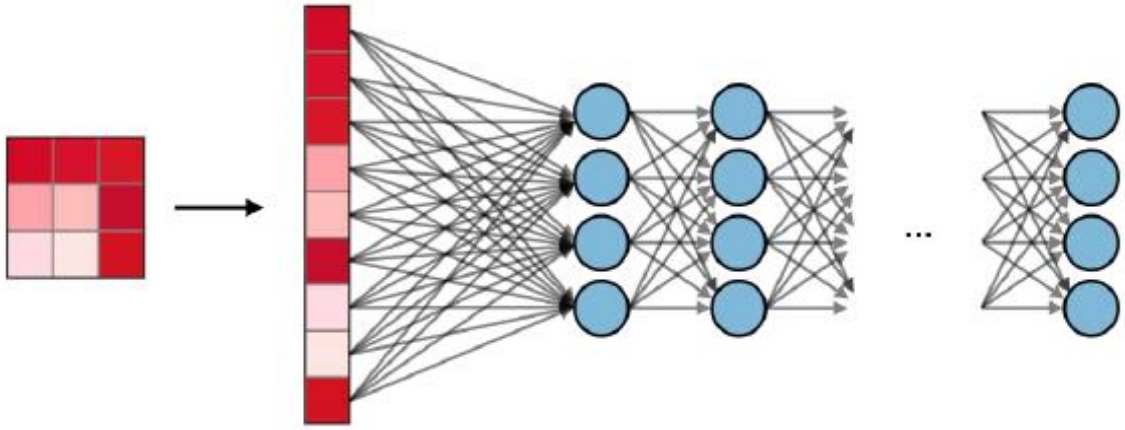


Figure 1.14 : *Couche entièrement connectée (FullyConnected layer).*

I.6.1.7 La fonction softmax :

La fonction softmax est une fonction d'activation souvent utilisée dans les réseaux de neurones pour produire des sorties normalisées et interprétables. Elle prend un vecteur de scores réels en entrée et produit un vecteur de probabilités normalisées en sortie. La fonction softmax est définie par l'équation suivante [18] :

$$p_i = \frac{e^{a_i}}{\sum e^{a_k}} \qquad \textbf{\textit{I.10}}$$

Où p_i est la probabilité de la classe i, a_i est le score de la classe i, et la somme est prise sur toutes les K classes. La fonction softmax transforme les scores d'entrée en une distribution de probabilité pour faciliter la tâche de classification. Elle est souvent utilisée comme dernière couche d'un réseau de neurones pour produire des sorties normalisées et interprétables.

I.6.2 Apprentissage d'un réseau CNN :

L'apprentissage de réseau CNN (Convolutional Neural Network) implique l'utilisation de grands ensembles de données pour entraîner le réseau à extraire des caractéristiques des images. Cette étape d'apprentissage est souvent supervisée, ce qui signifie que les images sont étiquetées avec les classes auxquelles elles appartiennent.

Une fois que le réseau a été entraîné, il peut être utilisé pour l'inférence, c'est-à-dire pour classifier de nouvelles images. Pendant l'inférence, le réseau prend une image en entrée et l'utilise pour prédire la classe à laquelle elle appartient.

Le processus d'apprentissage et d'inférence de réseau CNN est utilisé dans une variété d'applications, notamment la reconnaissance d'images, la détection d'objets, la segmentation d'images et la classification de vidéos. Il est particulièrement utile pour l'analyse de données visuelles complexes, telles que les images haute résolution ou les vidéos en mouvement.[22]

I.7 Conclusion :

Au cours de ce chapitre, nous avons présenté l'IA (intelligence artificielle) et ses sous domaines : ML (machine learning) et le DL (deep learning), avec ses différents types. Ensuite Nous avons examiné les réseaux neurones artificiels et ses sous domaines plus particulièrement les CNN, qui sont très efficaces dans la classification des images et la détection d'objets. Ce type de réseau est basé essentiellement sur l'opération de convolution qui permet d'extraire les caractéristiques significatives d'une image, afin de les utiliser pour une classification des images.

Chapitre II

Les circuits logiques programmables FPGA.

II.1- Introduction :

Au départ, l'apparition des FPGA (Field-Programmable Gate Arrays) s'est faite par l'intermédiaire de circuits logiques programmables de type PAL (Programmable Array Logic). Avec l'évolution de l'électronique, différentes familles de circuits programmables, tels que les CPLD (Complex Logic Programmable Device), les processeurs CPU et GPU.ont commencé à émerger. C'est en 1985 que la société XILINX a lancé les premiers circuits FPGA sur le marché, qui n'ont depuis cessé de se développer. Les FPGA sont passés de petites puces avec quelques milliers de blocs logiques à des systèmes sur puce comportant des milliards de transistors. L'avantage majeur des FPGA est leur capacité de reconfiguration rapide en quelques millisecondes, permettant ainsi une évolution des circuits pendant leur période d'utilisation. [23].

Grâce à ses caractéristiques remarquables telles que vitesse, parallélisme, capacités, coût, l'efficacité et les capacités reconfigurables dans notre projet nous préférions de l'utiliser car L'utilisation du FPGA pour la mise en œuvre de réseaux neuronaux offre une flexibilité dans les systèmes programmables.

II.2-Présentation et généralités sur les FPGA :

II.2.1 Définition :

Les FPGA sont des dispositifs électroniques contenant une matrice de circuits logiques reconfigurables qui sont connectés pour créer une implémentation matérielle d'une application logicielle spécifique. Contrairement aux processeurs, les FPGA utilisent un matériel dédié pour le traitement logique, n'ont pas de système d'exploitation et sont intrinsèquement parallèles, permettant à différents traitements de fonctionner simultanément sans concurrence pour les mêmes ressources. Les systèmes de contrôle basés sur FPGA peuvent appliquer une logique de verrouillage critique pour empêcher les erreurs humaines et offrent une reconfiguration flexible du circuit après le déploiement sur le terrain. Les FPGA offrent des performances et une fiabilité similaires à celles d'un circuit matériel dédié, tout en étant capables de remplacer des milliers de composants discrets grâce à la consolidation de millions de portes logiques dans une seule puce de circuit intégré. La puce FPGA contient des blocs logiques configurables (CLB) entourés d'une périphérie de blocs d'E/S, et les signaux sont acheminés dans la matrice FPGA par des commutateurs d'interconnexion programmables et des routes filaires [24].

II.2.2 Architecture matérielle des circuits FPGA :

Les circuits FPGA sont constitués d'une structure matricielle qui comprend deux types de blocs : les blocs d'entrées/sorties (IOB - Input Output Bloc) et les blocs logiques programmables (CLB - Configurable Logic Bloc), comme illustré dans la figure II.1. Le passage d'un bloc logique à un autre est effectué via un réseau d'interconnexions programmable [25].

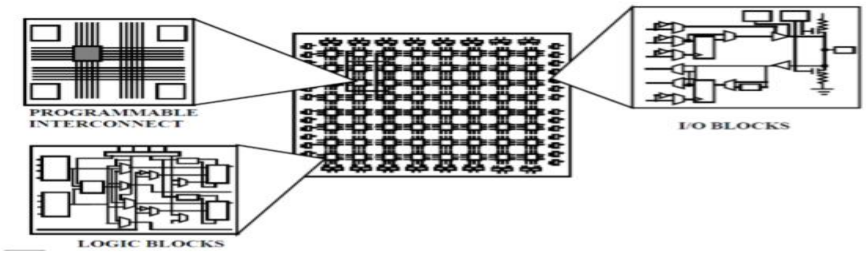


Figure II.1 : *Architecture interne d'un FPGA [25]*

II.2.2.1 Les blocs logiques configurables (CLB) :

Les blocs logiques configurables (CLB) sont les ressources logiques répétitives de base d'un FPGA. Lorsqu'ils sont reliés entre eux par les blocs de routage programmables, les CLB peuvent exécuter des fonctions logiques complexes, implémenter des fonctions de mémoire et synchroniser le code sur le FPGA. Les CLB contiennent de plus petits composants, y compris des bascules (FFs), des tables de recherche (LUTs) et des multiplexeurs (MUX). Un FF est la plus petite ressource de stockage du FPGA. Chaque FF dans un CLB est un registre binaire utilisé pour enregistrer les états logiques entre les cycles d'horloge sur un circuit FPGA. Un LUT stocke une liste prédéfinie de sorties pour chaque combinaison d'entrées. Les LUT fournissent un moyen rapide de récupérer la sortie d'une opération logique parce que les résultats possibles sont restaurés puis référencés plutôt que calculés. AMUX est un circuit qui sélectionne entre deux ou plusieurs entrées et retourne ensuite l'entrée sélectionnée. Anylogic peut être implémenté en utilisant la combinaison de FFs, LUTs et MUX. La configuration de base d'un bloc CLB est illustrée dans la figure II.3. [26]

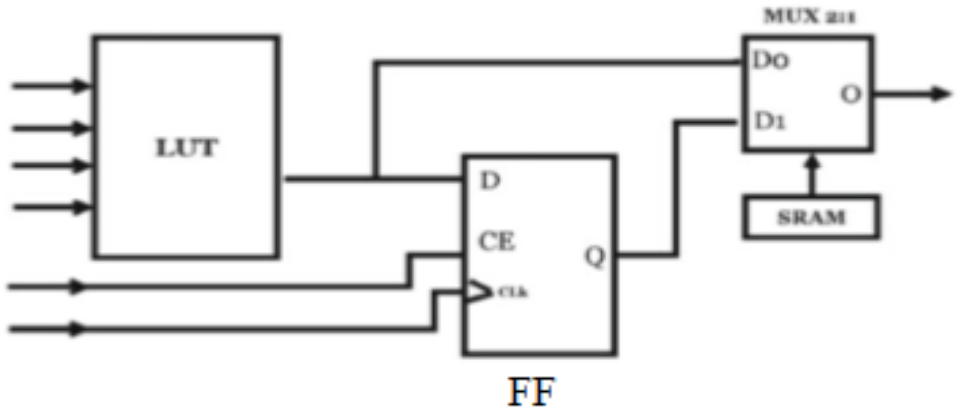


Figure II.2 : *Configuration de base d'une cellule CLB d'un FPGA. [27]*

II.2.2.2 Les Blocs IOB (Input Output Bloc) :

Les IOB (Input Output Bloc) sont des blocs d'entrées/sorties sur les circuits FPGA. Ils sont responsables de l'acheminement des signaux vers et depuis les broches du FPGA, ainsi que de l'adaptation de ces signaux aux niveaux de tension et de courant appropriés. Les IOB sont également capables de détecter les signaux d'entrée et de fournir des signaux de sortie correspondants pour la surveillance et le débogage des signaux. En outre, les IOB peuvent être configurés pour prendre en charge différents types d'interfaces de communication, telle que les bus série ou parallèle, les protocoles de communication Ethernet ou USB, ou les interfaces audio et vidéo. Les IOB sont souvent situés à l'extérieur de la matrice de blocs logiques configurables (CLB) sur les FPGA, de sorte qu'ils peuvent facilement communiquer avec les composants externes.[26]

Le schéma d'un IOB de Xilinx est illustré dans la figure II.3 et la figure II.4

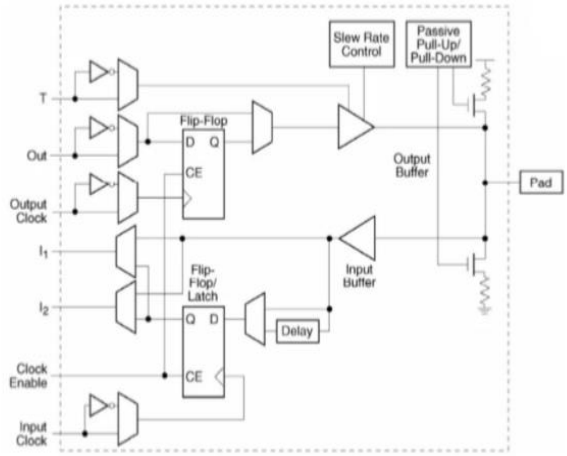


Figure II.3 : *structure d'un IOB de Xilinx sortie*

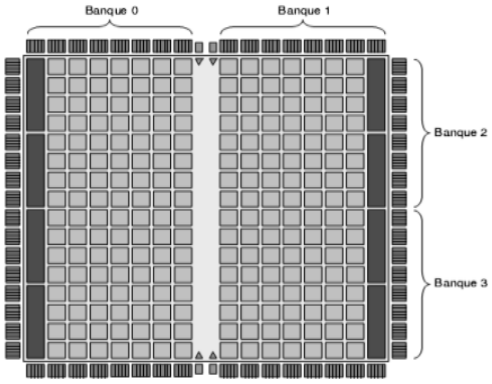


Figure II.4 : *Distribution des blocs (IOB) en banques [28]*

II.2.2.3 Blocs d'interconnexion :

Les blocs logiques configurables (CLB) et les blocs d'entrée/sortie (IOB) sont reliés entre eux par un réseau d'interconnexions programmable. Les connexions internes sont constituées de segments métallisés qui sont distribués horizontalement et verticalement entre les différents CLB. En parallèle, on trouve des matrices de routage programmables, réparties sur la totalité du circuit, qui permettent de connecter les différentes lignes à l'aide de transistors MOS dont l'état est contrôlé par des cellules de mémoire vive ou RAM (Random Access Memory).

Comme le montre la figure II.5, Xilinx propose trois types d'interconnexions différents pour répondre à cet objectif : les interconnexions à usage général, les interconnexions directes et les longues lignes.

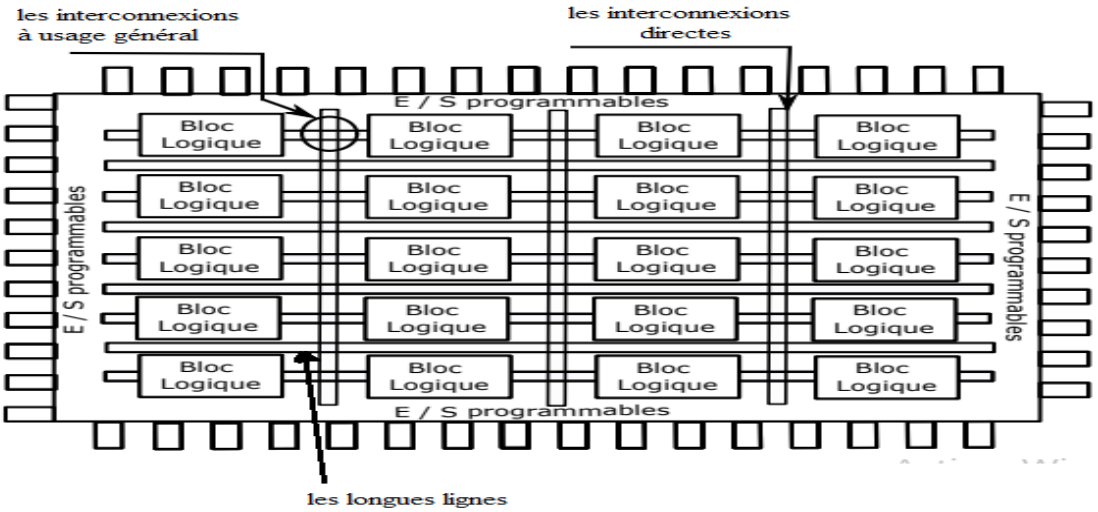


Figure II.5 : *Interconnexion interne d'un circuit FPGA.*

Les interconnexions à usage général sont utilisées pour relier un CLB à n'importe quel autre CLB à l'aide des matrices de routage, évitant ainsi que les signaux ne traversent les grandes lignes et ne soient affaiblis. Les interconnexions directes, quant à elles, relient les CLB voisins directement avec un maximum d'efficacité en termes de vitesse et d'occupation du circuit, sans passer par des matrices

d'interconnexion. Enfin, les longues lignes traversent le FPGA dans toute sa longueur et sa largeur, permettant de transmettre les signaux avec un minimum de retard entre les différents éléments dans le but d'assurer un synchronisme aussi parfait que possible. De plus, ces longues lignes permettent d'éviter la multiplicité des points d'interconnexion. [28]

II.2.2.4 Les blocs à usage spécifique :

Les FPGA disposent également de blocs à usage spécifique pour des fonctions particulières. Parmi eux, on peut citer :

II.2.2.4.1 Les blocs RAM (BRAM) :

Les blocs RAM sont des mémoires intégrées dans les circuits FPGA, conçues pour stocker de grandes quantités de données. Chaque bloc RAM peut être configuré en plusieurs sous-blocs de différentes tailles, mais le nombre de BRAM disponibles sur une puce est limité et varie selon le modèle de FPGA utilisé. Par conséquent, lors de la conception d'un circuit FPGA, le concepteur doit savoir combien de BRAM sont nécessaires pour sélectionner le modèle de FPGA approprié. Les BRAM sont utilisées pour diverses configurations, notamment la mémoire RAM à port unique, la mémoire RAM à double port et les registres FIFO.[18]

II.2.2.4.2 Les blocs DSP :

Les FPGA sont efficaces pour les applications de traitement numérique des signaux (DSP) car ils peuvent implémenter des algorithmes personnalisés et entièrement parallèles. Les applications DSP utilisent de nombreux multiplicateurs et accumulateurs binaires qui sont mieux implémentés dans les coupes DSP dédiées. Les blocs DSP combinent vitesse élevée et petite taille tout en conservant la flexibilité de conception du système. Les coupes DSP améliorent la vitesse et l'efficacité de nombreuses applications au-delà du traitement numérique du signal. La Figure II.6 montre l'architecture interne d'un bloc DSP48E de Xilinx. [29]

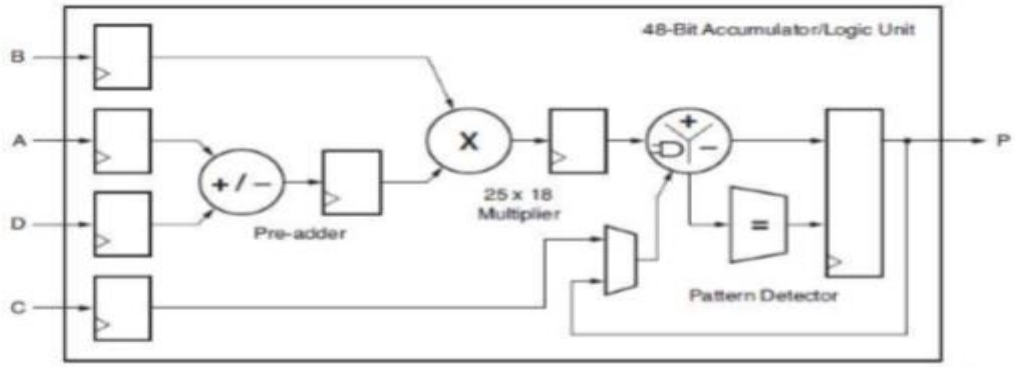


Figure II.6 : *architecture simplifiée d'un bloc DSP48E*

II.2.2.4.3- Les Processeurs Embarqués :

Dans de nombreuses applications, l'utilisation d'un processeur est nécessaire pour faciliter l'exécution de certaines tâches. Pour répondre à ce besoin, les fabricants de circuits FPGA ont inclus des processeurs embarqués dans leurs produits. En optant pour un circuit FPGA contenant un processeur embarqué (tel que le Virtex 5 de Xilinx), les développeurs peuvent simplifier considérablement le processus de conception et réduire l'utilisation des ressources FPGA ainsi que la consommation d'énergie. Xilinx propose deux familles de processeurs embarqués dans ses circuits FPGA : le picoBlaze et le microBlaze. [30]

II.2.2.4.4 Gestionnaire d'horloge numérique (DCM) :

Le Gestionnaire d'horloge numérique (DCM) est un composant électronique utilisé pour générer des signaux d'horloge numériques à des fréquences spécifiques dans les systèmes numériques. Il est souvent utilisé dans les conceptions de FPGA (Field Programmable Gate Array) pour permettre aux différents blocs fonctionnels de travailler ensemble en synchronisation, en ajustant la fréquence et le délai des signaux d'horloge pour répondre aux besoins spécifiques de chaque bloc. Le DCM peut également être utilisé pour effectuer des opérations telles que la multiplication de la fréquence d'horloge ou la division de la fréquence, ce qui peut être utile pour économiser de l'espace et de la puissance dans les conceptions électroniques. [25]

II.2.3 Les Différents Types d'FPGA :

Les FPGA ont différentes architectures et on distingue deux types principaux qui sont [26] :

- Les FPGA à anti fusibles
- Les FPGA basés sur SRAM
- Les FPGA à base de Flash

II.2.3.1 Les FPGA à Anti Fusibles :

Les FPGA anti-fusion sont des circuits programmables dont les configurations sont stockées dans des anti-fusibles, des composants qui peuvent être programmés une seule fois par l'utilisateur en appliquant une tension élevée qui crée une liaison permanente. Cette technologie présente des avantages tels que la sécurité, la fiabilité et la petite taille, mais elle est limitée par sa programmation unique. Les FPGA anti-fusion sont utilisés dans des applications critiques nécessitant une grande sécurité et une protection contre les attaques malveillantes.[31]

II.2.3.2 Les FPGA de types SRAM :

Les cellules de mémoire statique sont les cellules de base utilisées pour les FPGA basés sur SRAM. Ces dispositifs utilisent des cellules de mémoire statique qui sont divisées dans le FPGA pour fournir la reconfigurabilité. Un exemple d'une telle cellule de mémoire est montré à la *Figure II.7*. Dans un FPGA basé sur SRAM, les cellules SRAM sont principalement utilisées aux fins suivantes :

- Programmer l'interconnexion de routage de FPGA qui sont généralement dirigées par

Petits multiplexeurs.

- Programmer les blocs logiques configurables (CLB) utilisés pour mettre en œuvre les fonctions logiques.

La technologie de programmation basée sur la SRAM est devenue l'approche dominante pour les FPGA en raison de sa reprogrammabilité et de l'utilisation de la technologie de processus CMOS standard, ce qui conduit à une intégration accrue, vitesse plus élevée et consommation dynamique plus faible du nouveau processus avec une géométrie plus petite [32].

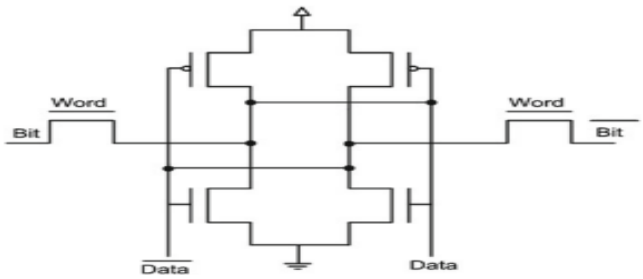


Figure II.7 : cellule de mémoire statique

II.2.3.3 Les FPGA à base de Flash :

Les FPGA basés sur Flash sont des circuits programmables non volatils et reprogrammables, offrant une faible consommation d'énergie et des temps de latence courts. Toutefois, la reconfiguration dynamique est déconseillée pour ces FPGA en raison des risques de destruction dus à des effets de rayonnement pendant le processus de reconfiguration. [31]

II.3 Les Familles et les caractéristiques des FPGA :

Il n'y a pas de normes spécifiques qui classifient les familles FPGA, mais plutôt des catégories basées sur les caractéristiques techniques des FPGA. Par exemple, les familles FPGA peuvent être classées en fonction du nombre de portes logiques équivalentes (LE), du nombre de blocs de mémoire intégrée (BRAM), de la capacité de traitement du signal numérique (DSP), de la connectivité Ethernet ou PCIe intégrée, etc. Chaque fabricant de FPGA peut avoir sa propre classification pour différencier les différentes familles de FPGA en fonction de ces critères. Par exemple, Xilinx classe ses familles FPGA en séries Virtex, Kintex et Artix, tandis que Altera (maintenant Intel) classe ses familles FPGA en séries Stratix, Arria et Cyclone.

II.3.1 La famille FPGA Virtex-7 :

La famille Virtex-7 de FPGA offre une capacité exceptionnelle et deux fois les performances de la série Virtex-6. Le plus grand appareil de la famille offre 2 millions de cellules logiques, soit 2,5 fois plus que tout autre FPGA, ainsi que des performances système exceptionnelles. La famille Virtex-7 est conçue pour répondre aux exigences des applications les plus exigeantes, offrant une haute bande passante d'E/S, un traitement à faible latence, un stockage sur puce pour les coefficients mis à jour dynamiquement et une prise en charge de la mémoire rapide hors puce. La famille prend en charge les fréquences critiques pour les applications sans fil, filaires, de radiodiffusion et aérospatiales et de défense, permettant aux clients de déplacer facilement leurs conceptions multi-puces à des implémentations à puce unique pour obtenir des avantages en termes de coût, de puissance et de performance.[33]

La figure II.8 donne une vue globale sur l'architecture interne de la famille Virtex-7.

La famille Virtex-7 de Xilinx comporte deux types de slices :

- Les slices de type **SLICE M** : ces slices sont optimisés pour les fonctions de traitement combinatoire et de mémoire. Elles disposent de deux blocs LUT (Look-Up Table) de 6 entrées chacun, ainsi que de deux Flip-Flops (FF) utilisables comme des registres ou des mémoires.

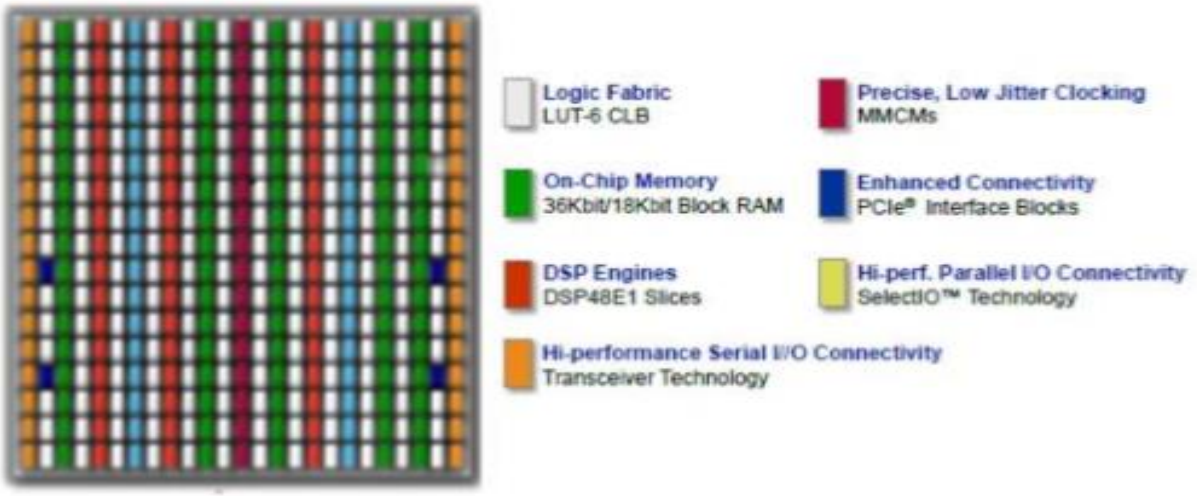


Figure II.8 *: Architecture interne de la famille Virtex 7 de Xilinx.*

- Les slices de type **SLICE L** : ces slices sont optimisés pour les fonctions de traitement séquentiel et de mémoire. Elles disposent d'un bloc LUT de 5 entrées, d'un bloc LUT de 6 entrées, et de deux Flip-Flops utilisables comme des registres ou des mémoires. Les SLICEL peuvent également être configurées en mode "distributed RAM", ce qui permet de créer des mémoires plus grandes et plus performantes que celles offertes par les SLICEM.

En général, les slices de type SLICEM sont utilisées pour les fonctions combinatoires et les petites mémoires, tandis que les slices de type SLICEL sont utilisées pour les fonctions séquentielles et les grandes mémoires. Cependant, chaque type de slice peut être configuré pour répondre à différents besoins de conception en utilisant les outils de conception de Xilinx.

Tableau II.1 : Ressources disponibles pour La famille FPGA Virtex-7.

	BLOCS CLB	**DSP**	**I/O**	**BLOCS BRAM**	**TRANSCEIVERS**
RESSOURCES	*Jusqu'à 2400*	*Jusqu'à 3600*	*Jusqu'à 1200*	*Jusqu'à 68 Mb*	*Jusqu'à 96 Gb/s*

II.3.1.1 Caractéristiques de La famille FPGA Virtex-7 :

La famille FPGA Virtex-7 est importante car elle offre une grande flexibilité, une haute performance et une grande capacité pour répondre aux besoins des applications les plus exigeantes. Parmi ses caractéristiques les plus importantes sont les suivantes [34] :

- **Taille :** La famille Virtex-7 va du plus 6,8 M de cellules logiques.
- **Technologie :** Les FPGA Virtex-7 sont construits selon un procédé de haute performance à faible puissance (HPL) de 28 nm.
- **Vitesse : peuvent** fonctionner jusqu'à 2,5 GHz.

- **Mémoire :** Chaque appareil Virtex-7 contient une mémoire de haute performance abondante, y compris Block RAM et UltraRAM.
- **Traitement de signal :** Chaque appareil Virtex-7 contient plusieurs blocs de traitement de signal numérique (DSP) haute performance avec jusqu'à 740 GMAC/s de performance de multiplication-accumulation.
- **Sécurité :** Les FPGA de Virtex-7 comprennent diverses caractéristiques de sécurité, comme le chiffrement AES-256 bits et la technologie IP sécurisée.

II.3.2 La famille FPGA Zynq :

La famille Zynq-7000 de Xilinx est construite sur une technologie de pointe de processus haute performance, basse consommation (HPL) de 28 nm et de High-k Gate (HKMG). Elle intègre un système de traitement (PS) ARM Cortex-A9 MPCore et une logique programmable (PL) Xilinx dans un seul appareil. Les processeurs ARM Cortex-A9 MPCore constituent le cœur du système PS et sont accompagnés d'une mémoire intégrée, des interfaces de mémoire externes et un riche ensemble de périphériques d'E/S. La gamme de périphériques de la gamme de SoC Zynq-7000 offre performances, puissance et facilité d'utilisation, généralement associées aux ASIC et ASSP, tout en offrant la flexibilité et l'évolutivité d'un FPGA. Bien que chaque périphérique de la famille Zynq-7000 contienne le même PS, les ressources PL et I / O varient d'un périphérique à l'autre. L'architecture Zynq-7000 permet la réalisation de fonctions système uniques et différenciées en mappant la logique et le logiciel personnalisés dans le PL et le PS respectivement. L'intégration du PS avec le PL offre des niveaux de performance inégalés par les solutions à deux puces, en raison de la bande passante d'E / S, du couplage libre et des budgets de puissance limités. [35]

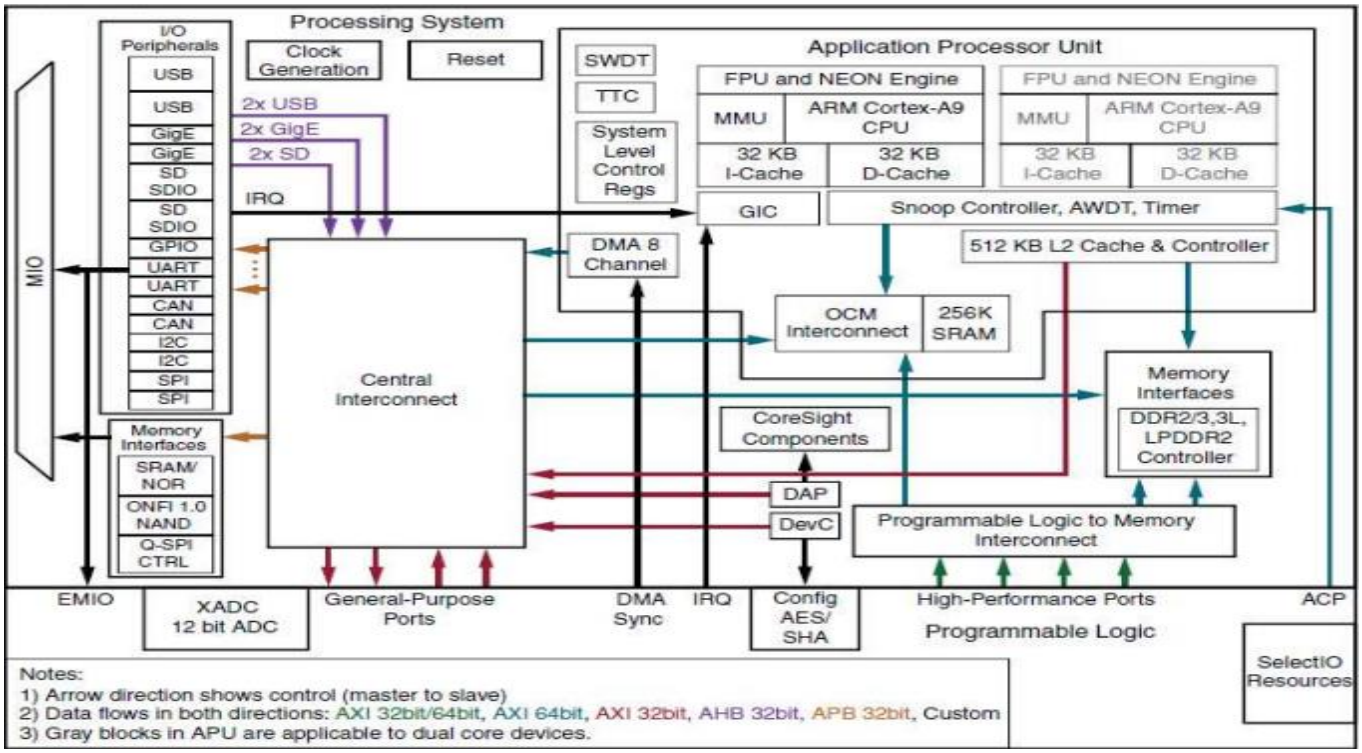


***Figure II.9** : Présentation du SoC Zynq-7000.*

Le tableau II.2 résume l'ensemble des ressources disponible de la famille Zynq -7000.

Tableau II.2 : *Ressources disponibles pour La famille FPGA Zynq (Zynq -7000).*

	BLOCS CLB	DSP	I/O	BLOCS BRAM	TRANSCEIVERS
RESSOURCES	Jusqu'à 53200	Jusqu'à 2200	Jusqu'à 1280	Jusqu'à 13.3 Mb	Jusqu'à 12.5Gb/s

II.3.2.1 Caractéristiques de la famille FPGA Zynq :

Voici quelques-unes des caractéristiques clés de la famille FPGA Zynq de Xilinx [35] :

- **Processeur ARM :** La famille de produits Zynq-7000 combine la puissance et la flexibilité des cœurs de processeurs ARM Cortex-A9 avec la haute performance de l'architecture FPGA programmable.

- **FPGA programmable :** La famille de produits Zynq-7000 est équipée d'un FPGA programmable qui permet aux concepteurs de personnaliser leurs conceptions de système en fonction de leurs besoins spécifiques.

-**Haute performance :** La famille de produits Zynq-7000 offre une haute performance grâce à la combinaison d'un processeur ARM Cortex-A9 avec un FPGA programmable de haute capacité.

-**Faible consommation d'énergie :** La famille de produits Zynq-7000 offre une faible consommation d'énergie grâce à la combinaison d'un processeur ARM Cortex-A9 à faible consommation d'énergie avec un FPGA programmable optimisé pour la consommation d'énergie.

-Haute connectivité : La famille de produits Zynq-7000 offre une haute connectivité avec des interfaces Ethernet, USB, CAN et autres, ainsi qu'un support pour les protocoles de communication sans fil tels que Wi-Fi et Bluetooth.

II.3.3 La famille UltraScale FPGA :

Les FPGA UltraScale de Xilinx sont des circuits intégrés programmables qui permettent aux utilisateurs de configurer la logique numérique pour répondre à des besoins spécifiques. Les FPGA UltraScale de Xilinx offrent une architecture de haute performance et de faible consommation d'énergie, dans un nœud FinFET de 16 nm. Ainsi qu'une grande flexibilité et une facilité de programmation. Ils sont constitués de blocs de logique programmable hétérogènes, tels que des LUTs (Look-Up Tables), des blocs de mémoire RAM, et des blocs de traitement de signaux numériques (DSPs). La structure des FPGA UltraScale est conçue pour faciliter la connexion entre ces différents blocs, permettant ainsi la création de circuits personnalisés pour des applications spécifiques, cela faisant d'UltraScale FPGA le choix idéal pour les systèmes de prochaine génération dans les communications filaires et sans fil, les centres de données, l'automobile, l'industrie et les applications militaires. [36]

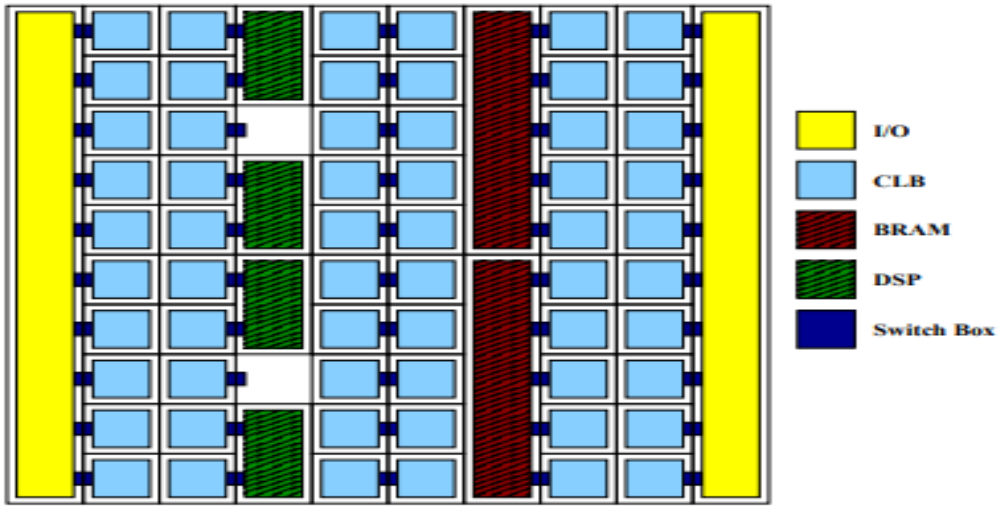


Figure II.10 *: Illustration de l'architecture Xilinx UltraScale*

Tableau II.3 : Ressources disponibles pour La famille UltraScale FPGA.

	BLOCS CLB	DSP	I/O	BLOCS BRAM	TRANSCEIVERS
RESSOURCES	*Jusqu'à 1920000*	*Jusqu'à 6840*	*Jusqu'à 4320*	*Jusqu'à 168Mb*	*Jusqu'à 32.75 Gb/s*

II.3.3.1 Les Caractéristiques de La famille UltraScale FPGA :

La famille UltraScale FPGA de Xilinx est la dernière génération de dispositifs programmables de la société, offrant des performances et une efficacité énergétique améliorées par rapport aux générations précédentes. Selon Xilinx, les appareils UltraScale peuvent offrir des performances allant jusqu'à 50 % supérieures à celles des appareils de la génération précédente, tout en consommant jusqu'à 45 % moins d'énergie. Parmi les autres caractéristiques clés de la famille UltraScale FPGA, on peut citer [36] :

- Une matrice FPGA améliorée, avec une architecture modulaire pour une meilleure utilisation des ressources et une plus grande flexibilité de conception.

- Des capacités d'interconnexion avancées, avec des connexions intégrées jusqu'à 28 Gbps pour une bande passante élevée et une latence réduite.

- Des blocs de traitement numérique (DSP) améliorés pour une puissance de traitement maximale des signaux.

- Des capacités de mémoire étendues, y compris des blocs de mémoire RAM ultra-rapides et des interfaces mémoire DDR4 pour une performance maximale.

- Une prise en charge complète des normes industrielles, notamment PCIe Gen3, HDMI 2.0 et 10/40/100G Ethernet.

II.4 Les Défis de la mise en œuvre de CNN sur les circuits FPGA :

La mise en œuvre de réseaux de neurones convolutionnels (CNN) sur des circuits intégrés programmables (FPGA) peut présenter plusieurs défis pour les chercheurs. Voici quelques-uns des principaux défis [37] :

-Contraintes de ressources : Les FPGA ont des ressources limitées en termes de mémoire et de puissance de traitement, ce qui peut rendre difficile la mise en œuvre de CNN complexes. Les chercheurs doivent donc concevoir des architectures optimisées qui utilisent efficacement les ressources disponibles.

- Complexité de la conception : La mise en œuvre de CNN sur FPGA peut être complexe en raison de la complexité des architectures de CNN et de la difficulté à concevoir des architectures FPGA efficaces pour répondre aux besoins des réseaux de neurones. Les chercheurs doivent donc être capables de concevoir des architectures FPGA qui prennent en compte les caractéristiques des CNN.

- Performances limitées : Les performances des FPGA pour la mise en œuvre de CNN peuvent être limitées par rapport aux autres technologies d'accélération des réseaux de neurones, telles que les GPU. Les chercheurs doivent donc trouver des moyens d'optimiser les performances des FPGA pour la mise en œuvre de CNN.

- Coûts élevés : Les FPGA peuvent être plus coûteux que d'autres technologies d'accélération des réseaux de neurones, ce qui peut rendre la mise en œuvre de CNN sur FPGA moins abordable. Les chercheurs doivent donc trouver des moyens de réduire les coûts de la mise en œuvre de CNN sur FPGA.

- Gestion de la mémoire : La gestion de la mémoire est un défi important lors de la mise en œuvre de CNN Par exemple, le modèle AlexNet a besoin de 250 Mo de mémoire pour stocker les poids basés sur une représentation en virgule flottante 32 bits. Ce problème est exacerbé par des modèles plus complexes, tels que le modèle VGG-16, qui requiert plus de 138 millions de poids.

la mise en œuvre de CNN sur des circuits FPGA peut être complexe et exigeante. Cependant, avec une planification appropriée et une expertise en conception FPGA, il est possible d'optimiser les performances, de minimiser les coûts.

II.5 Environnement de travail :

II.5.1 Vivado Design Suite :

La suite de conception Vivado® est conçue pour améliorer la productivité. Cette suite d'outils a été conçue pour accroître la productivité globale de la conception, de l'intégration et de la mise en œuvre de systèmes utilisant les dispositifs UltraScale™ et 7 séries, Versal® et Zynq® UltraScale+™ MPSoCs. Les appareils xilinxMD sont maintenant beaucoup plus gros et sont dotés d'une variété de nouvelles technologies, y compris la technologie d'interconnexion au silicium empilé (SSI), les interfaces d'E/S à haute vitesse jusqu'à 28 giga octets (Go), les microprocesseurs et les périphériques durcis, le signal mixte analogique et plus encore. Ces dispositifs plus grands et plus complexes créent des défis de conception multidimensionnelle, lorsqu'ils sont manipulés incorrectement, qui peuvent empêcher la réalisation d'un délai de commercialisation plus rapide et une productivité accrue. Avec la Vivado Design Suite, vous pouvez accélérer la mise en œuvre de la conception avec des outils de localisation et d'acheminement qui optimisent de manière analytique pour des paramètres de

conception multiples et simultanés, tels que la synchronisation, la congestion, la longueur totale du fil, l'utilisation et la puissance. La Vivado Design Suite offre des capacités d'analyse de conception à chaque étape de la conception. Cela permet de modifier la conception et la configuration des outils plus tôt dans les processus de conception où ils ont moins d'impact sur le calendrier global, réduisant ainsi les itérations de conception et accélérant la productivité.[38]

La Figure II.11 donne une vue globale de l'environnement de visualisation Vivado IDE.

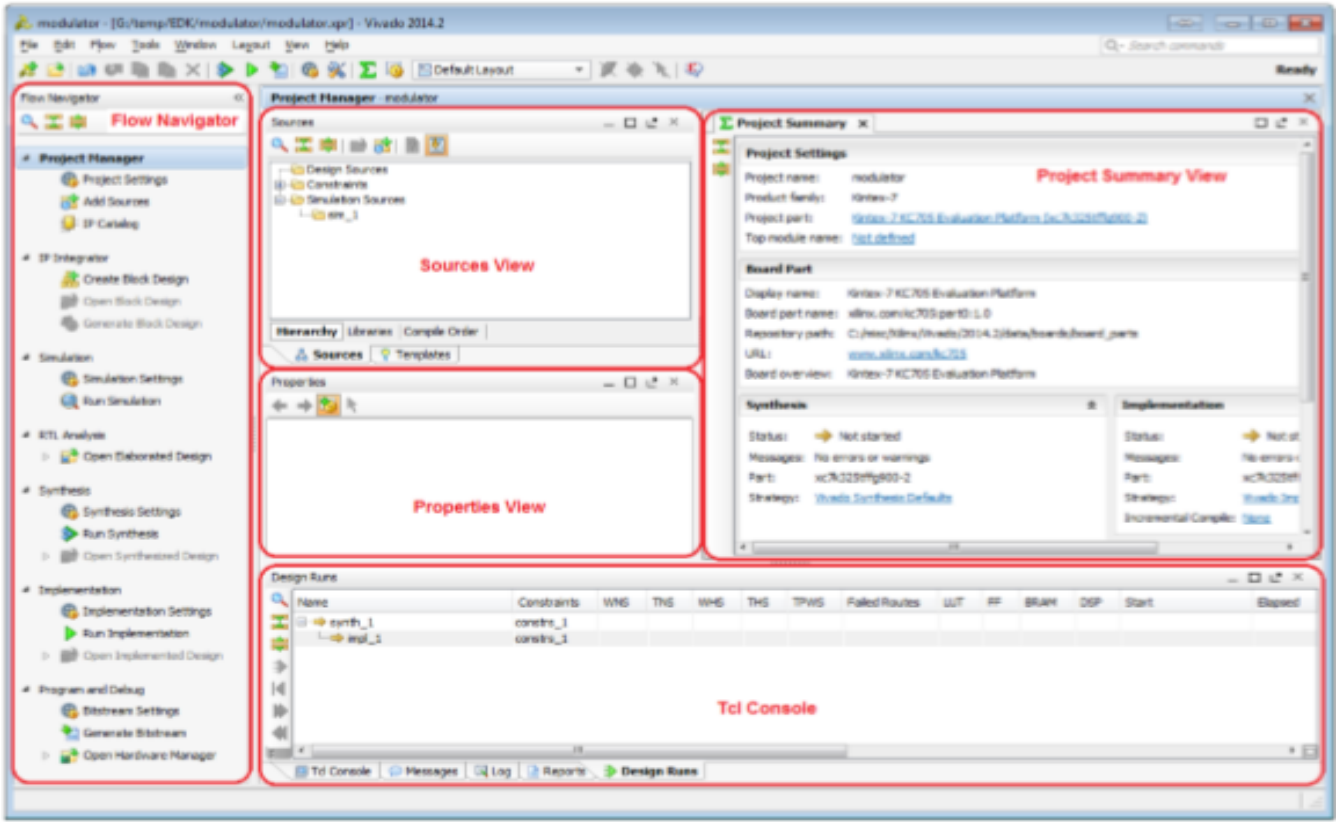


Figure II.11 *: L'environnement de visualisation de l'IDE de Vivado.*

II.5.2. System Générateur (XSG) :

Les langages de description de matériel modernes, tels que VHDL et VERILOG, simplifient le processus de développement de systèmes numériques. Toutefois, la mise en œuvre et la réalisation d'une conception sur une plateforme matérielle réelle demeurent difficiles pour les ingénieurs. Les méthodes de conception classiques sont inappropriées pour les systèmes numériques sophistiqués, allongeant ainsi les cycles de développement des projets. Par conséquent, il est essentiel d'avoir une plateforme de prototypage rapide permettant aux ingénieurs de se concentrer sur la conception du système plutôt que sur l'écriture et le débogage du code VHDL. C'est pourquoi XILINX a introduit la plateforme SYSTEM GENERATOR FOR DSP (XSG).

XSG n'est pas un programme en soi, mais plutôt une bibliothèque qui étend l'environnement SIMULINK de MATLAB. C'est l'intégration avec SIMULINK qui fait la force de XSG. Les ingénieurs peuvent facilement synthétiser leur conception vers un FPGA en utilisant les blocs fournis par XILINX, tout en utilisant SIMULINK pour modéliser et évaluer les performances de leur conception.

Mais, les blocs fournis par XSG ne permettent pas de concevoir toutes les fonctions souhaitées, ce qui limite la flexibilité du designer. [39]

II.6 Conclusion :

Ce chapitre a été consacré à la présentation des circuits logiques programmables FPGA et leur utilisation pour l'implémentation des réseaux CNN. Nous avons commencé par la présentation des différents blocs constituants l'architecture de base de ce type de circuits. En plus des trois éléments classiques qui sont les CLB, les IOB et les interconnexions, l'évolution des FPGA a permis l'intégration de bloc permettant une amélioration importante de performance comme les blocs DSP, le processeurs embarqués et les BRAM . Nous avons présenté aussi les caractéristiques de certaines familles récentes de circuits FPGA comme vertex_7, Zynq et Ultrascal. Nous avons mentionné certains défis de la mise en œuvre et d'implémentation des réseaux CNN sur circuits FPGA.

Enfin nous avons présenté les principaux logiciels qui facilitent la conception et l'implémentation des applications sur circuit FPGA. Pour la conception, nous avons utilisé le logiciel « Xilinx-System-Generator 2018» et pour l'implémentation nous avons utilisé le logiciel Vivado 2018.

Chapitre III

Résultats de simulation et d'implémentations

III.1- Introduction :

La détection et la classification des armes à partir d'une image ou d'une vidéo peut être utilisée pour la prévention rapides d'incidents potentiellement mortels tels que le terrorisme, les infractions pénales générales ou même la violence domestique. L'un des moyens de faire la classification d'une façon précise et efficace consiste à utiliser les réseaux de neurones convolutifs CNN.

Les réseaux CNN sont basés essentiellement sur l'opération de convolution qui est une opération très gourmande en nombre d'opérations arithmétiques. Il est donc nécessaire d'utiliser des circuits de calcul performants pour implémenter cette opération. Parmi les solutions envisagées pour ce type de problème, les circuits FPGA ont montré une très grande efficacité et souplesse grâce à leur reconfigurabilité et à leur capacité de calcul parallèle.

Dans ce chapitre nous allons représenter les résultats de simulation et les résultats d'implémentation sur circuit FPGA d'un réseau CNN destiné à faire la classification de cinq types d'armes qui sont : fusil d'assaut, fusil de chasse, pistolet et couteau et grenade

III.2 Prétraitement et collecte de données

III.2.1 Collecte des images

L'apprentissage d'un réseau de neurones de type CNN est une tâche critique qui détermine grandement ses performances. Pour faire réussir l'apprentissage il faut choisir judicieusement la base de données à utiliser pour cet objectif car un mauvais choix de la base de données peut mener à une dégradation des performances même si le réseau est bien conçu. Pour notre cas, l'un des problèmes que nous avons rencontrés est le manque de bases de données contenant plusieurs classes d'armes et avec un nombre suffisant d'images pour chaque classe.

Après recherche sur internet, nous avons trouvé une base de données brute qui contient sept classes d'armes, mais avec un nombre insuffisant d'images pour certaines classes [40]. Pour compléter la base de données, nous avons téléchargé d'autres images pour la classe grenade pour avoir une base de cinq classes. Le tableau III.1 donne le nombre d'images pour chacune des cinq classes.

Tableau* III.*1 : *Nombre d'images par chaque classe de la base de données utilisées pour la classification de cinq types d'armes.*

Classe	Nombre d'images
Assault_Rifle (fusil d'assaut)	927
Grenade	940
Hunting_Rifle (fusil de chasse)	849
Knife (couteau	1076
Pistol (pistolet)	1170

La figure III .1 présente des exemples aléatoires d'échantillons de chaque classe de la base de données.



Figure III***.1 :*** *Exemples d'images de chaque classe de la base de données*

Nous souhaitons mentionner les étapes suivantes que nous avons effectuées avant d'utiliser les images de la base de données :

III.2.2 Redimensionnement :

Comme l'indique la figure III.1, les images de la base de données n'ont pas les mêmes dimensions et puisque le réseau CNN nécessite des images d'entrée de même dimension il est donc nécessaire d'unifier les dimensions de toutes les images.

Au début, nous avons essayé d'utiliser directement la fonction *imresize* incluse dans le toolbox *image processing* de *Matlab,* mais nous avons constaté que cette fonction introduit des déformations importantes sur les images ce qui influe sur la classification. La figure III.2.b montre l'effet de déformation de la fonction *imresize* appliquée sur un échantillon de la base de données (figure 3.2.a). On remarque clairement que l'image qui appartient à la classe *Hunting_Rifle* peut être facilement confondu avec la classe Pistol après l'application d'un simple redimensionnement.

Pour résoudre le problème de déformation introduit par la fonction *imresize*, nous avons appliqué un opérateur de padding pour rendre l'image carrée, ensuite nous avons appliqué la fonction *imresize* sur l'image carrée. L'opération de padding consiste à ajouter des pixels blancs (255) sur la plus petite dimension de l'image pour obtenir une matrice carrée. En appliquant cette technique de redimensionnement les images résultantes ne contiennent aucune déformation comme le montre l'exemple de la figure III.2.c

Pour étudier l'effet de la taille de l'image sur la classification, nous avons choisi trois tailles différentes : 100×100, 150×150 et 200×200.



a b *c*

Figure III.2 *Illustration de la déformation introduite par le redimensionnent (a) image originale, (b) image redimensionnée par l'application directe de imresize, (c) image redimensionnée par l'application de padding suivie de imresize.*

III.2.3 Conversion des images en niveaux de gris et en couleurs RVB :

Après inspection de la totalité des images de la base de donnée nous avons trouvé que la majorité des images sont en couleurs RVB c'est-à-dire de dimension n×m×3, mais un nombre limité d'images sont monochromes de dimension n×m×1. A partir de ce mélange nous avons

formé deux bases de données, une base en niveau de gris obtenue par la transformation des images RVB en image niveau de gris et une autre base RVB obtenue par la transformation des images monochrome en images RVB.

Pour réaliser cette tache nous avons développé un programme MATLAB qui nous permet de changer toutes les images dans la base de données en niveau de gris tout en préservant les images qui étaient à l'origine en niveau de gris



Figure* III.*3 *: Exemples d'images de la base de données avant et après leur c*onversion en niveaux de gris

Les images en niveau de gris présentent des avantages en termes de réduction de la taille des données, de simplification du traitement et de conservation des caractéristiques structurelles importantes. Ce qui facilite les tâches de classification, d'analyse et de reconnaissance des motifs.

III. 3 Conception et évaluation du réseau CNN :

Cette étape critique de notre travail consiste à choisir la structure du réseau CNN qui va être ensuite implémenté sur le circuit FPGA et à tester ses performances de classification. Nous avons utilisé la version 2018 du logiciel *Matlab* qui contient un toolbox très riche en fonctions spécialement conçu pour le développement des réseaux CNN.

III.3.1 Architecture du réseau CNN proposé :

On se qui concerne l'architecture du réseau proposée, nous avons opté pour un bon compromis entre la complexité du réseau et la justesse de sa classification. Nous avons constaté que l'utilisation d'un réseau de taille moyenne ayant quatre couches de convolution permet d'obtenir de bons résultats de classification. L'architecture du réseau choisi est expliquée dans la figure III.4.

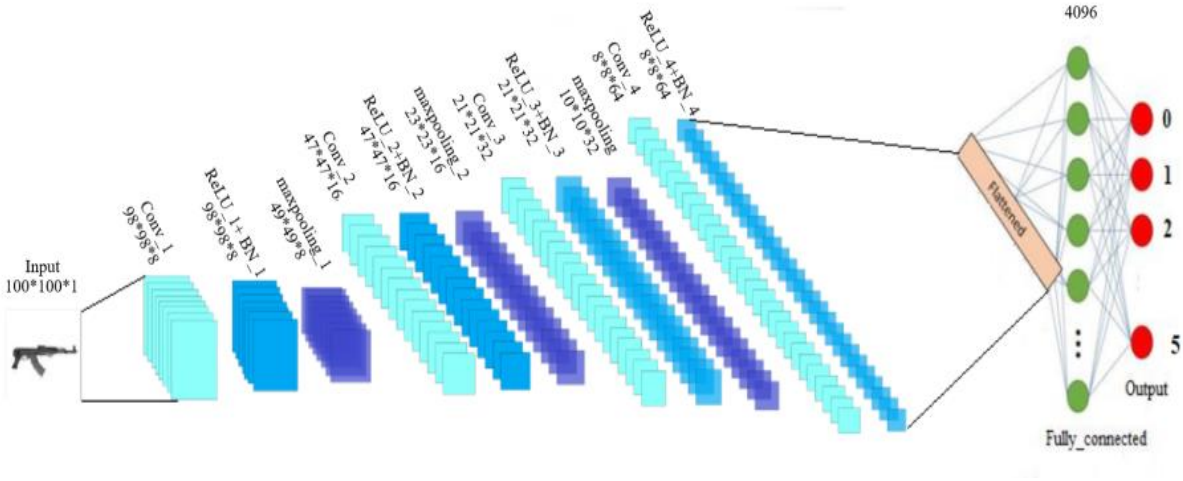


Figure III.4 : *l'architecture du réseau CNN proposé*

Le réseau est composé de 19 couches dont la première est la couche d'entrée et la dernière est la couche de classification. Entre la couche d'entrée et celle de sortie on trouve : quatre couches de convolution (conv_1 à conv_4), quatre couches de normalisation et d'activation (ReLu1+BN_1 à ReLu4+BN_4), trois couches de sous échantillonnage (maxpooling_1 à maxpooling_3) et une couche entièrement connectée (Fully_connected). On remarque que le réseau possède une structure régulière puisque chaque couche de convolution est suivie d'une couche de normalisation et d'activation. En plus, chaque couche d'activation est suivie d'une couche de sous échantillonnage à l'exception de la dernière couche.

Le tableau III.2 donne le nombre d'entrées (input feature maps) et le nombre de sortie (output feature maps) poue chacune des quatre couches de convolution.

Tableau III.2 nombres d'entrées et nombre de sorties pour les couches de convolution du réseau proposé.

Couche	**Nombre d'entrées**	**Nombre de sortie**
Conv_1	1 pour gray et 3 pour RVB	8
Conv_2	8	16
Conv_3	16	32
Conv_4	16	64

III.3.2 Conception et évaluation du réseau :

Le programme *Matlab* que nous avons développé pour concevoir le réseau CNN et pour évaluer ses performances de classification peut être divisé en quatre partis qui sont :

- Préparation du data store pour l'apprentissage et le test.
- Détermination de la structure du réseau.
- Apprentissage du réseau.
- Evaluation du réseau.

III.3.2.1 Préparation du data store pour l'apprentissage et le test :

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 i*mageDatastore* 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*.

III.3.2.2 Détermination de la structure 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 « [ ] ». Chaque couche du réseau est définie en utilisant une fonction *Matlab* spécifié comme le montre le tableau III.3.

Tableau III.3 *Les fonctions Matlab utilisées pour la détermination des différentes couches du réseau CNN.*

Couche	**Fonction Matlab**	**Couche**	**Fonction Matlab**
Entrée	*imageInputLayer*	activation	*reluLayer*
Convolution	*convolution2dLayer*	Sous échantillonnage	*maxPooling2dLaye*
Normalisation	*batchNormalizationLayer*	Entièrement connectée	*fullyConnectedLayer*

III.3.2.3 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*. Les options de base qu'on peut choisir pour l'apprentissage sont :

- *solverName* : elle spécifie l'algorithme d'optimisation utilisée pour ajuster les valeurs des filtres et des biais lors de la phase d'apprentissage. Pour les réseaux CNN, l'algorithme le plus utilisée est le *« Gradient Descent with momentum : sgdm»*. C'est un algorithme d'optimisation itératif utilisé en machine learning pour trouver les valeurs des filtres et des biais qui assure le minimum de la fonction cout.
- *Plots* : cette option spécifie les graphes à afficher durant la phase d'apprentissage.
- *Verbose :* Indicateur pour afficher les informations sur la progression de l'entraînement.
- *MaxEpoch :* c'est le nombre maximal d'époch à utiliser pour l'entraînement. Un époch correspond à un seul passage en avant et en arrière de la totalité des données d'apprentissage à travers le réseau. Pour les bases de données de grande taille, on ne pout pas transmettre l'intégralité de la base de données au réseau neuronal en une seule fois. Ainsi, on divise les données en nombre de lots ou parties appelé *batch*.

- *Shuffle* : Option pour le mélange des données
- *ValidationData* : les données à utiliser pour la validation durant le processus d'apprentissage.
- *ValidationFrequency* : c'est le nombre d'itération séparant de validations consécutives durant l'apprentissage. Une itération correspond au passage d'un seul batch. Donc, le nombre total d'itérations est égale au nombre de batch par epoch multiplier par le nombre total d'epochs.

Après le choix des options d'apprentissage, on lance l'opération d'apprentissage comme suivant :

net=trainNetwork(TrainImages,layers, options);

On donne trois paramètres à la fonction *trainNetwork* qui est *:*

- *TrainImages* : c'est le nom de la partie d'apprentissage du Data store.
- *layers* : c'est la structure du réseau.
- *options* : c'est les options d'apprentissage définies dans le paragraphe précédent.
  La figure III.5 montre l'évolution de l'étape d'apprentissage pour un réseau qui ne contient pas de couche de normalisation. Cette figure montre bien l'intérêt de la couche de normalisation pour notre réseau puisque sans normalisation la justesse de classification est très faible (23%).

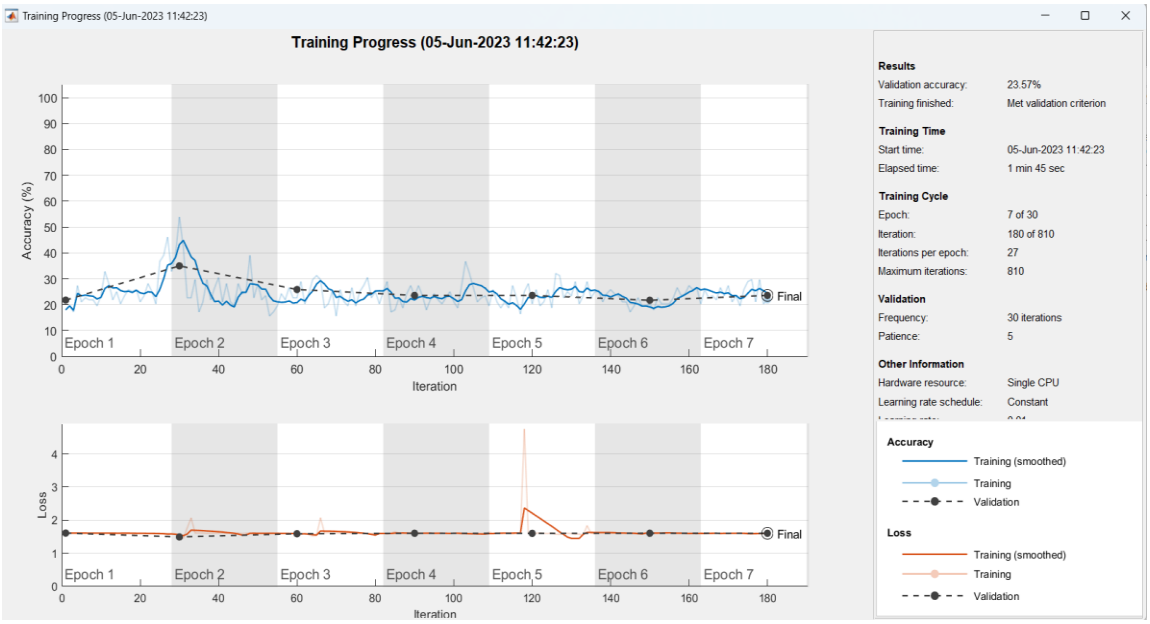


Figure III.5 : *Justesse et erreur de classification pour un réseau CNN sans couche de normalisation.*

La figure III.6 représente l'évolution de la justesse de classification et de la fonction de perte pour un même réseau CNN, mais avec l'utilisation de la normalisation de batch. Cette figure montre clairement l'effet positif de la normalisation de batch sur l'augmentation significative de la justesse et de la vitesse d'apprentissage du réseau. En

effet, on constate une amélioration considérable de la justesse qui passe de 23,53% à 97,11%.

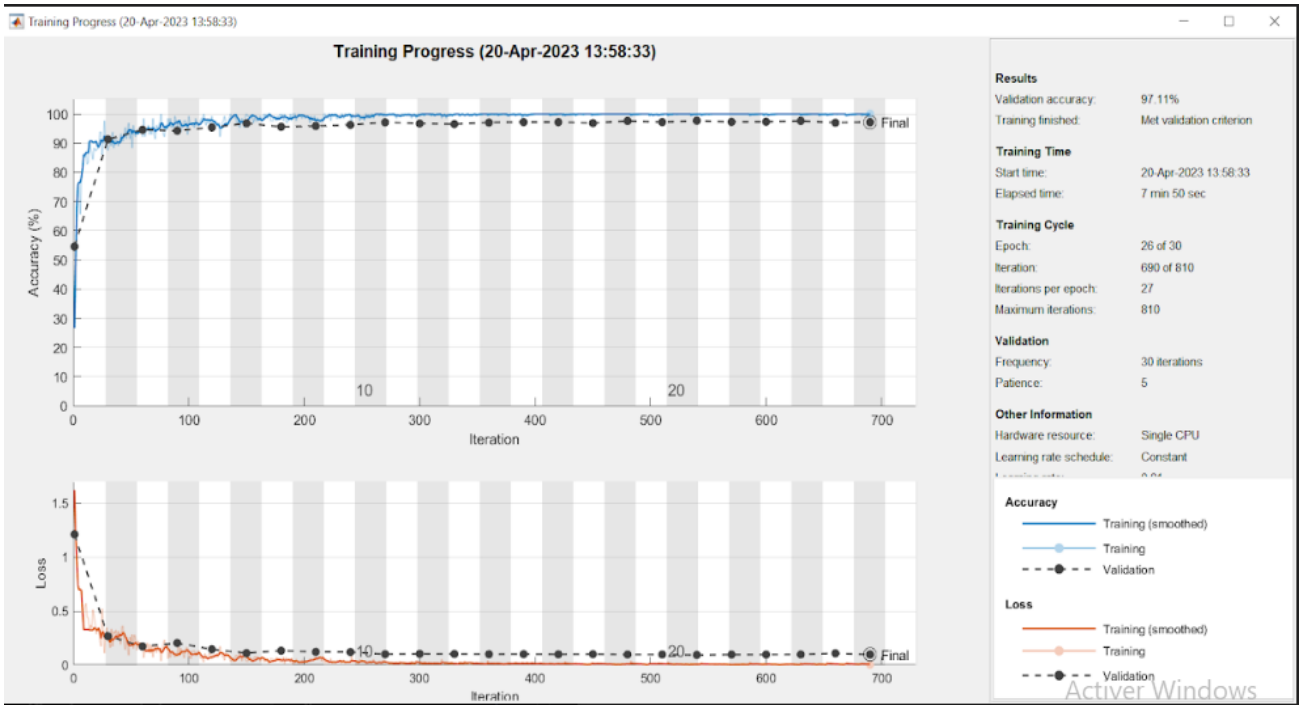


Figure III. 6 : *justesse et erreur de classification pour un réseau CNN avec couche de normalisation.*

III.3.2.4 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.

Dans notre travail nous avons visé l'évaluation de l'influence des paramètres suivants sur les performances de classification :

- Dimension de l'image d'entrée : nous avons testé trois taille différentes 200×200, 150×150 et 100 ×100.
- Nombre de canaux de l'image d'entrée : nous avons utilisé des images en niveaux de gris (un seul canal) est des images en couleur RVB (3 canaux)
- Taille des filtres des différentes couches de classification : nous avons essayé plusieurs
- combinaisons de filtres de taille 11×11, 7×7, 5×5 et 3×3.

Le tableau III.4 récapitule les résultats de classification en termes de justesse (accuracy) obtenus pour les différentes valeurs des paramètres d'évaluation.

Tableau III .4 : Résumé des résultats de classification obtenu pour les différentes combinaisons de paramètres d'évaluation.

Architecture	Avec batch normalisation						Sans batch normalisation
	Images en niveaux de gris (gray)			Images couleurs (RVB)			Images {gray ,RVB}
	100 ×100	150 ×150	200 ×200	100 ×100	150 ×150	200 ×200	100,150,200
3_3_3_3	97.11%	94.96%	93.15%	96.84%	96.67%	93.75%	23.59%
5_3_3_3	97.88%	96.77%	93.75%	97.58%	95.56%	91.3%3	23.59%
5_5_3_3	96.57%	94.56%	94.35%	96.09%	94.76%	90.86%	23.59%
7_5_3_3	95.20%	92.23%	90.23%	95.78%	94.34%	91.75%	23.59%
11_5_3_3	92.71%	90.98%	92.80%	94.43%	93.95%	88.96%	23.59%
11_7_5_3	92.23%	93.86%	91.65%	94.15%	94.53%	89.25%	23.59%

La première colonne du tableau donne la combinaison de filtres utilisés pour les différentes couches de convolution. Par exemple, la combinaison 7_5_3_3 signifie que le réseau utilise des filtres de tailles : 7×7 pour la première couche de convolution, 5×5 pour la deuxième couche de convolution et 3×3 pour la troisième et la quatrième couche de convolution.

A partir du tableau on peut tirer les remarques suivantes :

- Le meilleur résultat est obtenu pour la combinaison *5_3_3_3* avec un taux de justesse de ***97.88%.*** Ce taux est largement acceptable pour la majorité des systèmes de classification.
- La taille d'images d'entrée qui donne le meilleur résultat pour la plupart des cas est 100×100.
- Les taux de justesse supérieurs à 96% sont tous obtenus pour des images en niveaux de gris.

Les résultats du tableau nous ont permis de choisir la combinaison de paramètres qui va être implémenté sur le circuit FPGA. Pour les filtres, nous avons choisi la combinaison *3_3_3_3* puisqu'elle utilise des filtres de même dimension pour toutes les couches de convolution et elle assure un taux de justesse de 97.11% qui est très proche du taux maximal de 97.88%. Nous avons opté pour cette solution car la structure régulière des filtres facilite la tâche d'implémentation sur circuit FPGA. Nous avons choisi la taille 100×100 car elle assure le meilleur taux de classification en comparaison avec les tailles 150×150 et 200×200.

Après le choix du réseau à implémenter, nous allons décrire dans le paragraphe suivant les différents circuits de calcul pour les différentes couches du réseau.

III.4 Conception des circuits de calcul pour implémentation sur FPGA :

Notre travail se concentre principalement sur une étape cruciale : la conception des circuits qui exécutent les différentes opérations du réseau CNN. Nous avons pris soin de concevoir ces circuits en prenant en compte la relation entre le temps, les ressources et la complexité de la conception.

Une fois conçus, ces circuits sont convertis en RTL (registre transfert Level) et implémentés sur le circuit FPGA cible. Pour réaliser cette conception, nous avons utilisé le logiciel System Générateur, qui nous a permis de concevoir les circuits de convolution, et de BN (batch Normalusation), de ReLU et de Max pooling. De plus, pour la mise en forme des données après chaque étape de calcul, nous avons utilisé le logiciel *Matlab*.

Nous avons commencé par la conception de circuits de calcul complètement parallèle. La conception parallèle permet d'obtenir une architecture d'implémentation très rapide mais elle consomme un nombre énorme de ressources FPGA. Dans la plupart des cas un circuit FPGA complet ne suffit pas pour implémenter une architecture complètement parallèle d'un réseau CNN, même s'il est simple.

Pour résoudre ce problème nous avons proposé des circuits de calcul série basés sur la réutilisation des ressources. Ce qui nous a permis d'implémenter efficacement notre réseau.

III.4.1 Circuits de calcul parallèle :

III.4.1.1 Circuit de calcul parallèle de la convolution :

Le schéma de la figure III.7 explique le circuit de calcul parallèle correspondant à une couche de convolution. Le circuit reçoit n_i entrées parallèles notées *entrée_1* à *entrée_*n_i et permet de calculer n_o sorties en parallèle notées *conv_out_1* à *conv_out_*n_o. Le circuit de calcul est composé de n_o groupes de calcul de convolution dont chaque groupe s'occupe du calcul d'une seule sortie d'une façon parallèle et indépendante des autres groupes. A son tour, un groupe de calcul n'est composé de ni unités de convolution élémentaires notées *conv_i_j* et d'un sommateur à ni entrées, chaque unité calcul la convolution entre une entrée et un seul filtre.

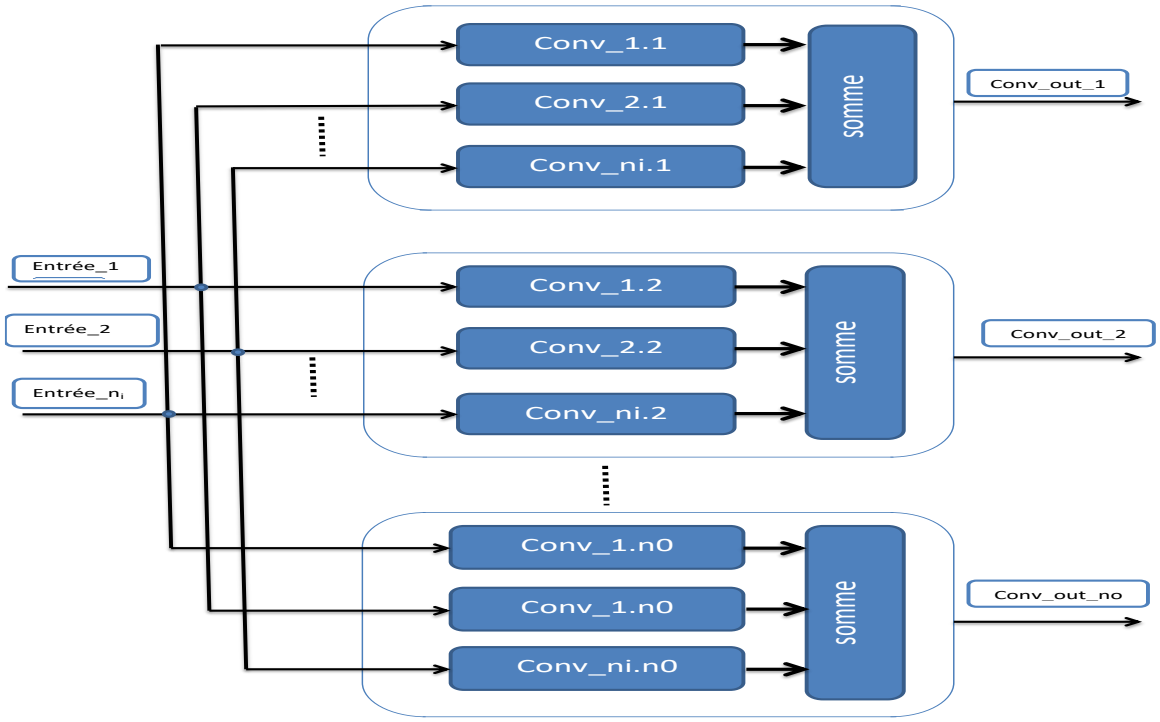


Figure III.7 : *Schéma fonctionnel du circuit de calcul parallèle correspondant à une couche de convolution.*

Il convient de souligner que chaque unité de convolution élémentaire *conv_i_j* comprend 9 multiplieurs, 9 additionneurs pour les nombres réels, 9 éléments de mémorisation pour le stockage des valeurs de filtre de convolution en plus des éléments de retard, comme l'indique le schéma de la figure III.8.

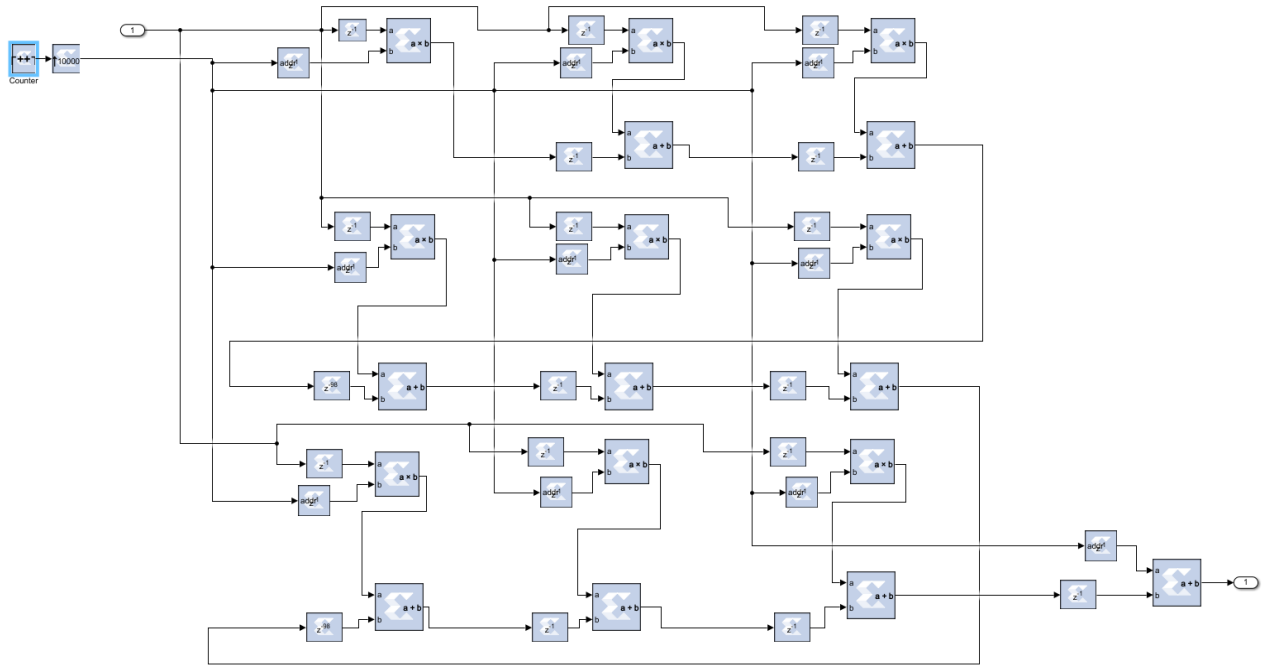


Figure III.8 : *Schéma détaillé d'une unité de convolution élémentaire conv_i_j.*

Le tableau III.5 donne les caractéristiques de chaque circuit de calcul de convolution correspondant aux quatre couches de convolution de notre réseau.

Tableau III.5 Caractéristiques des circuits de calcul parallèle des différentes couches de convolution

Couche	Nombre d'entrées (ni)	Nombre de sorties (no)	Dimension des entrées	Dimension des sorties	Retard de début calcul (en cycle)	Nombre de multiplieurs	Nombre d'additionneurs	Eléments de mémoire
1	1	8	100×100	98×98	204	72	72	80
2	8	16	49×49	47×47	102	1152	1160	1280
3	16	32	23×23	21×21	50	4608	4624	5120
4	32	64	10×10	8×8	24	18432	18464	20480

Il est important de noter que les circuits de calcul introduisent un retard au début du calcul de chaque matrice, en plus d'un retard de deux cycles horloge au début de chaque ligne de la matrice de sortie. De plus, chaque branche génère le résultat de la convolution sous forme d'un vecteur unique. Nous avons utilisé un programme *Matlab* pour éliminer les retards et pour convertir les vecteurs en matrice afin de comparer avec les résultats de calcul de *Matlab*.

La figure III.9 montre le schéma de calcul parallèle correspondant à la deuxième couche de convolution conçu en utilisant *System Generator*.

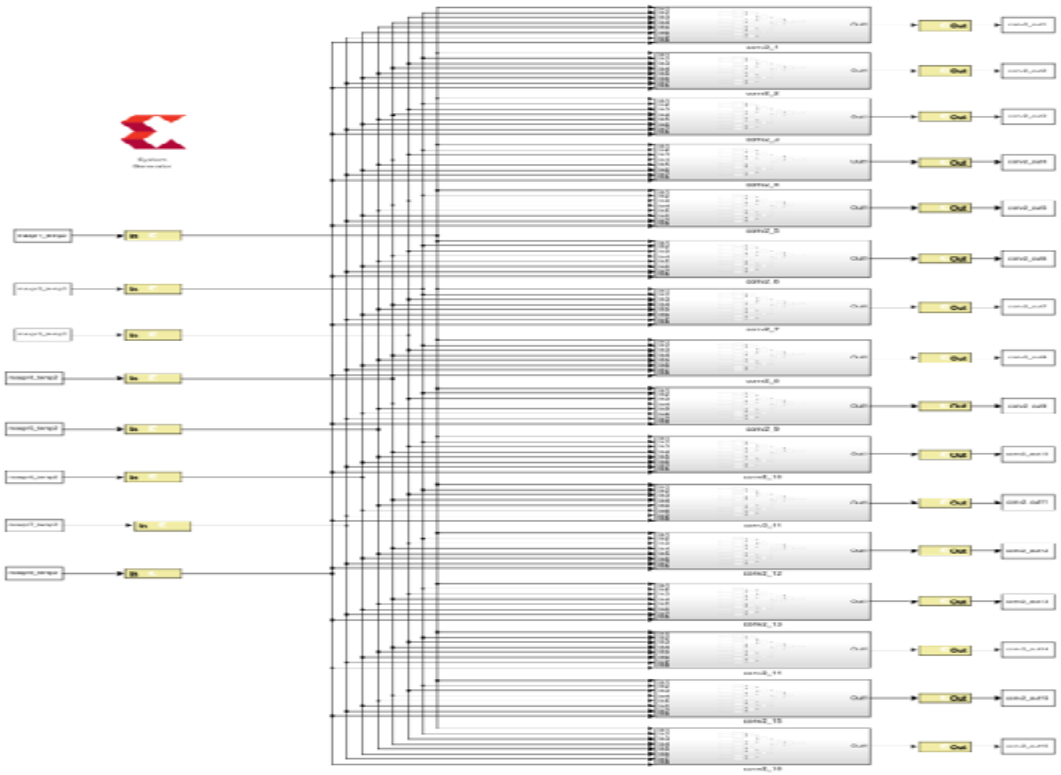


Figure III.9 *: schéma global du circuit de calcul parallèle de la deuxième couche de convolution.*

III.4.1.2 Circuit de calcul parallèle de la normalisation (BN) et d'activation et ReLU 1 et Max pooling 1 :

Le circuit de calcul de BN (batch normalisation) et d'activation (ReLu) est simple, il est composé de n_o branches identiques dont chaque branche permet de faire la normalisation et l'activation ReLu d'une sortis de la couche de convolution qui le précède comme montré dans la figure III.10.

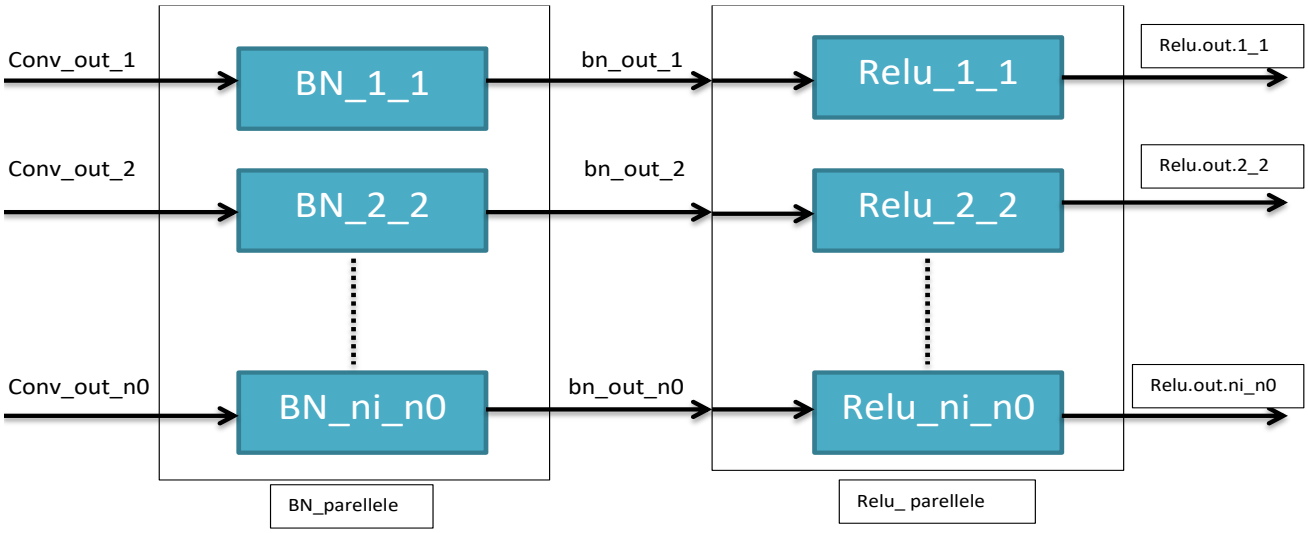


Figure III.10 *Schéma fonctionnel du circuit de calcul parallèle de normalisation BN et d'activation ReLu.*

Chaque branche de normalisation BN contient un soustracteur, deux multiplieurs et un additionneur et des éléments de mémorisation et chaque branche d'activation contient un multiplexeur, un extracteur du bit de signe et un élément de mémorisation.

Le tableau III.6 donne les caractéristiques de chaque circuit de calcul de normalisation BN correspondant aux quatre couches BN de notre réseau.

Tableau III.6 Caractéristiques des circuits de calcul parallèle des différentes couches de normalisation BN.

Couche	Nombre de branches (no)	Dimension des entrées/ sortie	Nombre de multiplieurs	Nombre d'additionneurs/ soustracteurs	Eléments de mémoire
BN_1	8	98×98	16	16	32
BN_2	16	47×47	32	32	64
BN_3	32	21×21	64	64	128
BN_4	64	8×8	128	128	256

Le tableau III.7 donne les caractéristiques de chaque circuit de calcul d'activation ReLu correspondant aux quatre couches ReLu de notre réseau.

Tableau III.7 Caractéristiques des circuits de calcul parallèle des différentes couches de ReLu.

Couche	Nombre de branches (no)	Dimension des entrées/ sortie	Nombre de Multiplexeurs	Eléments de mémoire
ReLu_1	8	98×98	8	8
ReLu_2	16	47×47	16	16
ReLu_3	32	21×21	32	32
ReLu_4	64	8×8	64	64

III.4.1.c Le circuit de calcul parallèle de MaxPooling :

La structure du circuit de calcul parallèle correspondant à la couche de Maxpooling est similaire à celle de normalisation et d'activation du point de vue de nombre de branches. La figure III.11 montre que ce circuit est composé de n_o branches dont chaque branche réalise l'opération de MaxPooling d'une entrée indépendamment des autres branches. Chaque branche est composée d'un élément de retard, d'un élément de sous-échantillonnage de fréquence et de deux éléments de calcul du maximum.

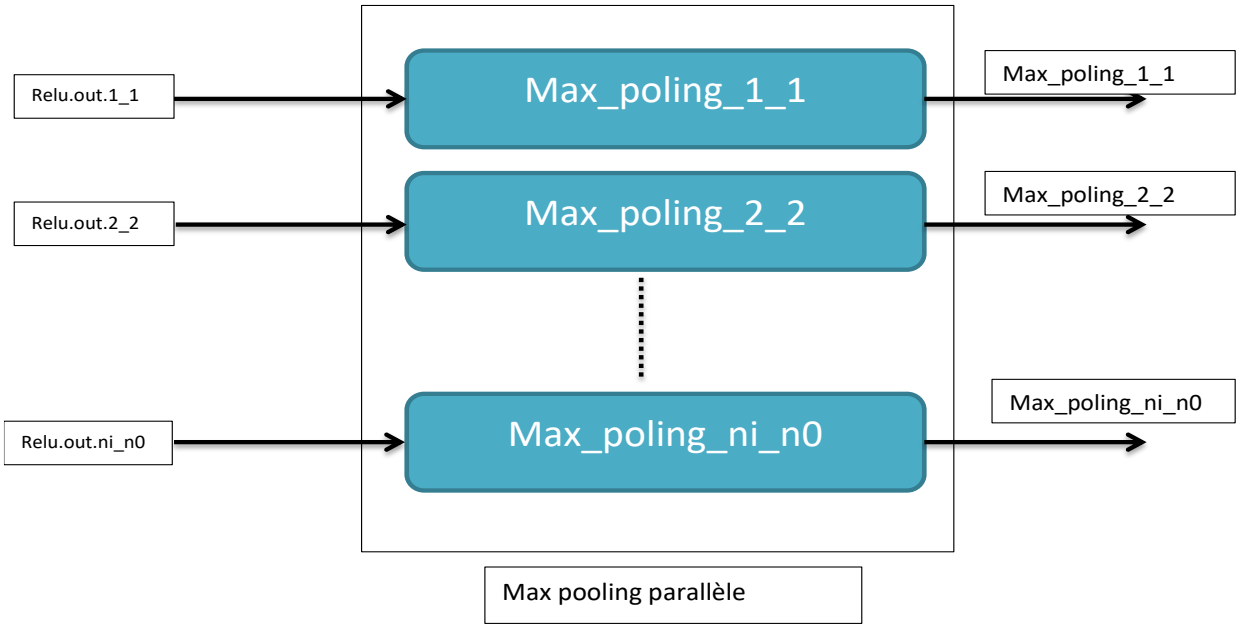


Figure III.11 *: Schéma fonctionnel du circuit de calcul parallèle de MaxPooling.*

Le tableau III.8 donne les caractéristiques de chaque circuit de calcul de MaxPooling correspondant à notre réseau.

Tableau III.8 Caractéristiques des circuits de calcul parallèle des différentes couches de MaxPooling

couche	Nombre de branches (no)	Dimension des entrées	Dimension des sorties	Retard de début calcul (en cycle)	Nombre de Multiplexeurs	Nombre de comparateurs	Nombre de sous- échant de fréquence
MaxPooling_1	8	98×98	49×49	50	16	16	8
MaxPooling_2	16	46×46	23×23	24	32	32	16
MaxPooling_3	32	20×20	23×23	11	64	64	32

III.4.1.4 Le circuit de calcul parallèle de la couche complètement connectées et de classification :

Le circuit de la couche complètement connectée se divise en deux parties distinctes, comme illustré dans la Figure III.12. La première partie assure un calcul parallèle des scores de classification pour les six classes d'armes. La deuxième partie calcule le score maximum ainsi que le numéro de la classe correspondante.

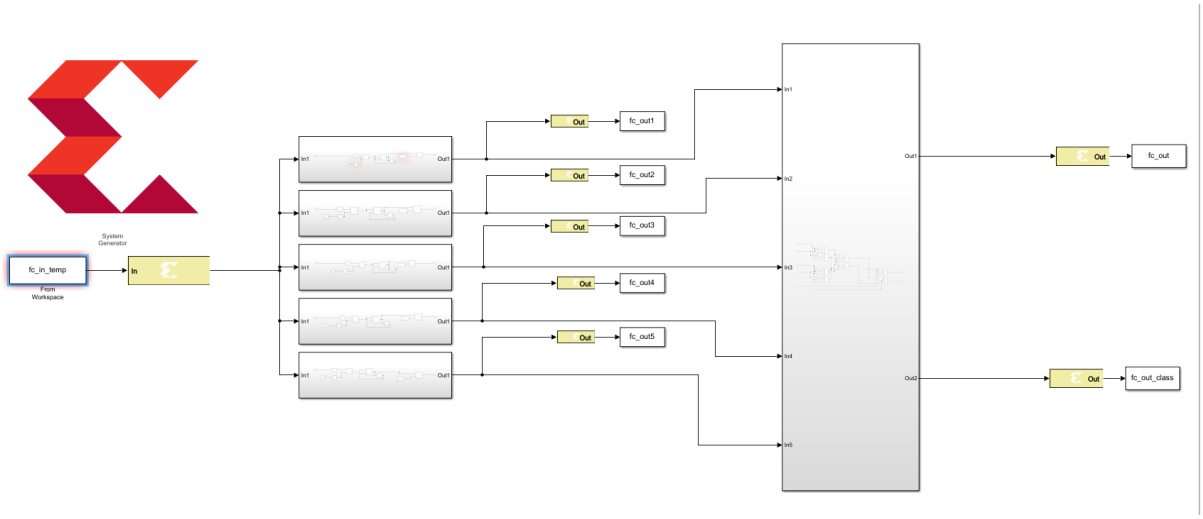


Figure III.12 : *La structure du circuit de calcul de la couche complètement connectées*

Le circuit de calcul de la couche entièrement connectée et de classification contient : 5 multiplieurs, 10 additionneurs, 5×4096 éléments de mémoire, 8 comparateurs et 8 multiplexeurs.

III.4.2 Circuits de calcul série :

Les tableaux III.3 à III.6 montrent que les circuits de calcul parallèle nécessitent un nombre important d'éléments de calcul et de mémorisation ce qui rend leur implémentation sur circuit FPGA très difficile si ce n'est pas impossible.

Il est nécessaire de faire appel à des circuits basés sur le calcul série au lieu du calcul parallèle. Par exemple, au lieu d'utiliser un circuit qui contient huit branches parallèles pour calculer huit sorties en parallèle, on utilise une seule branche pour calculer les huit sorties l'une après l'autre.

Cette méthode est plus efficace du point de vue consommation de ressources mais elle nécessite plus de temps pour effectuer les calculs.

III.4.2.1 Circuit de calcul série de la convolution :

Le schéma de la figure III.13 explique le circuit de calcul série correspondant à une couche de convolution. Au lieu d'utiliser n_o groupes parallèles pour calculer les n_o maps de caractéristiques en parallèle, un seul groupe est utilisé pour calculer les n_o maps en série.

Étant donné que le même circuit est utilisé pour calculer les différentes maps de caractéristiques de sortie, et sachant que chaque sortie utilise son propre filtre, il est important de mettre en place une procédure permettant une utilisation correcte des filtres. Pour résoudre cette problématique, on a utilisé un circuit qui repose sur l'utilisation de mémoires et d'un compteur. Les mémoires sont utilisées pour stocker les valeurs des filtres, tandis que le compteur est utilisé pour générer les adresses d'accès aux mémoires.

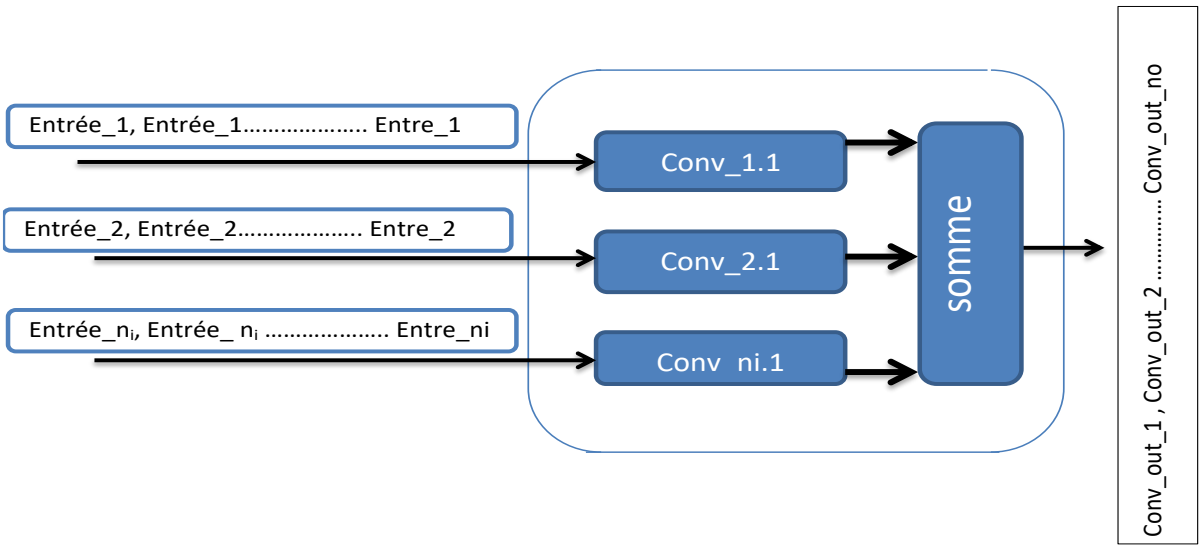


Figure III.13 : *Schéma fonctionnel du circuit de calcul série correspondant à une couche de convolution.*

Le tableau III.9 donne les caractéristiques de chaque circuit de calcul de convolution correspondant aux quatre couches de convolution de notre réseau.

Tableau III.9 : Caractéristiques des circuits de calcul série des différentes couches de convolution

couche	Nombre de multiplieurs	Nombre d'additionneurs	Eléments de mémoire
1	9	9	10
2	72	72	73
3	144	144	145
4	288	288	289

III.4.2.2 - Circuit de calcul série de la normalisation (BN) et d'activation et ReLU 1 et Max pooling 1 :

Comme le montre le schéma fonctionnel de la figure III.14, le circuit de calcul série de la normalisation et d'activation contient une seule branche permettant d'obtenir les n_os sortis en série. Le circuit BN nécessite l'utilisation de mémoires et d'un compteur pour gérer les paramètres de normalisation utilisés durant chaque phase de calcul.

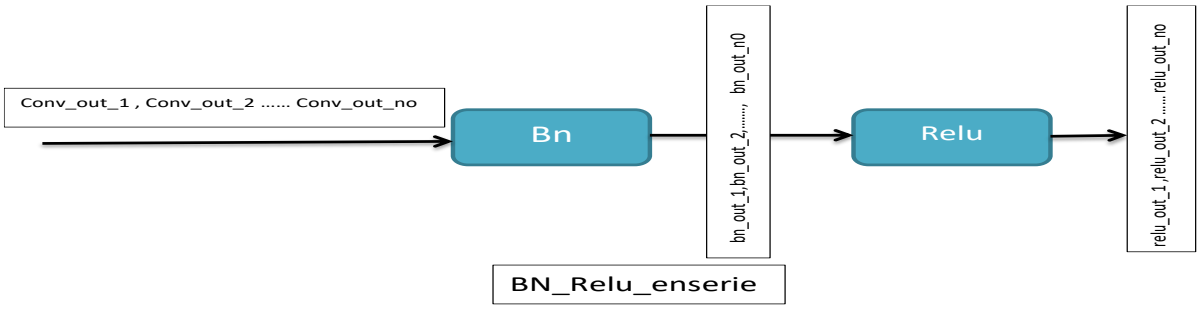


Figure III.14 *: Schéma fonctionnel du circuit de calcul série de normalisation BN et d'activation ReLu.*

Le tableau III.10 donne les caractéristiques de chaque circuit de calcul de normalisation BN et d'activation ReLu correspondants de notre réseau.

Tableau III.10 : Caractéristiques des circuits de calcul série des différentes couches de normalisation BN et d'activation ReLu.

	Normalisation (BN)			Activation ReLu	
Couche	Nombre de multiplieurs	Nombre d'additionneurs/ soustracteurs	Eléments de mémoire	Nombre de Multiplexeurs	Eléments de mémoire
1	2	2	4	1	1
2	2	2	4	1	1
3	2	2	4	1	1
4	2	2	4	1	1

III.4.2.3 Le circuit de calcul série de MaxPooling :

Comme pour le cas du circuit de calcul de normalisation BN et d'activation Relu, le circuit de calcul série du Maxpoling aussi contient une seule branche de calcul comme l'illustre le schéma fonctionnel de la figure III.15.

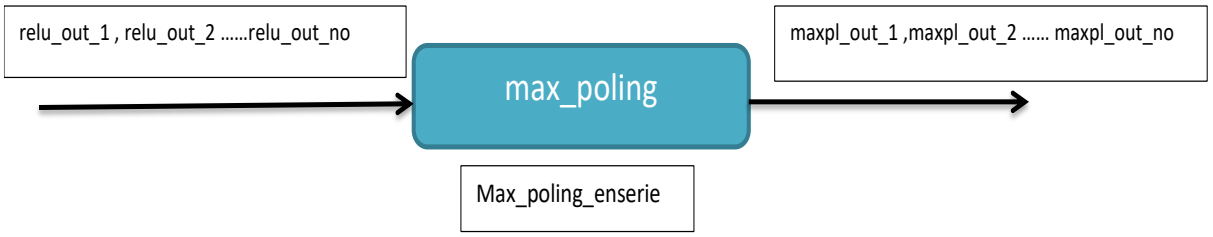


Figure III.15 : *Schéma fonctionnel du circuit de calcul série de MaxPooling.*

Le tableau III.11 donne les caractéristiques de chaque circuit de calcul de MaxPooling correspondant à notre réseau.

Tableau III.11 Caractéristiques des circuits de calcul série des différentes couches de MaxPooling

couche	Nombre de Multiplexeurs	Nombre de comparateurs	Nombre de sous- échant de fréquence
MaxPooling_1	2	2	1
MaxPooling_2	2	2	1
MaxPooling_3	2	2	1

III.5 Résultats d'implémentation des circuits sur FPGA :

Après conception des différents circuits en utilisant le logiciel *System Generator*, et après vérification du bon fonctionnement de chaque circuit par comparaison de leurs sorties avec les résultats de calcul de *Matlab* nous avons généré le modèle *RTL* (Register Transfert Level) de chaque circuit. Le modèle RTL est lancé sous l'environnement *Vivado* pour faire l'implémentation sur FPGA.

III.5.1 Implémentation des circuits de calcul parallèle.

Les conceptions des circuits de calcul du modèle CNN proposées dans l'étape précédente offrent de meilleures performances en termes de temps d'exécution grâce à leur approche parallèle. Cependant, ces conceptions nécessitent une consommation importante de ressources et leur implémentation sur une seule carte FPGA peut s'avérer difficile, voire impossible, surtout lorsqu'on utilise la représentation en virgule flottante des données.

Le tableau III.12 présente les ressources du circuit FPGA Virtex 7 XC7VX485T-2FFG1761, consommées par les différents circuits de calcul ainsi que la puissance total et la puissance dynamique consommée par les différentes couches du réseau.

Tableau III.12 : *ressources du circuit Virtex 7 XC7VX485T-2FFG1761 et puissance consommées par les différentes couches du circuit parallèle.*

	Slice		LUT		FF		DSP		BRAM		POWER(W)
	Utilisation	Pourcentage	Utilisation	Pourcentage	Utilisation	Pourcentage	Utilisation	Pourcentage	Utilisation	Pourcentage	Total Utilisation
Conv1	**38844**	**4.26%**	**33365**	**10.99%**	**3584**	**0.59%**	**216**	**7.71%**	**0**	**0%**	**1.49**
BN1 ReLU1 Maxp1	**1286**	**0.01%**	**8189**	**2.7%**	**1254**	**0.21%**	**48**	**1.71%**	**0**	**0%**	**0.587**
Conv2	**492032**	**545.28%**	**533840**	**175.84%**	**57344**	**9.44%**	**3456**	**123.36%**	**0**	**0%**	**2.98**
BN2 ReLU2 Maxp2	**2572**	**0.02%**	**16378**	**5.4%**	**2508**	**0.42%**	**96**	**3.42%**	**0**	**0%**	**1.174**
Conv3	**19888128**	**2726.4%**	**100095**	**174.4%**	**10752**	**9.44%**	**13824**	**493.44%**	**0**	**0%**	**4.47**
BN3 ReLU3 Maxp3	**5144**	**0.4%**	**32756**	**10.8%**	**5016**	**0.84%**	**192**	**6.84%**	**0**	**0%**	**2.348**
Conv4	**7955251 2**	**8722.44%**	**133460**	**348.8 %**	**14336**	**18.88 %**	**55296**	**1973.76%**	**0**	**0%**	**5.96**
BN4 ReLU4	**10288**	**0.08%**	**65512**	**21.6%**	**10032**	**1.68%**	**384**	**13.68%**	**0**	**0%**	**3.94**
FC	**4704**	**0.51%**	**4185**	**1.38%**	**380**	**0.06%**	**15**	**0.54%**	**20**	**1.94%**	**0.603**

III.5.2 Implémentation des circuits de calcul série avec représentation des données en virgule flottante :

Le tableau III.13 montre les ressources matérielles utilisées du circuit FPGA Virtex 7XC7VX485T-2FFG1761 et la puissance consommée par les différents circuits de calcul série. Les données dans les différentes étapes de calcul sont représentées en virgule flottante simple précision.

Tableau III.13 : ressources *du circuit Virtex 7 XC7VX485T-2FFG1761 consommées par différant couches de calcule en série (virgule* flottante*).*

	Slice		**LUT**		**FF**		**DSP**		**BRAM**		**POWER(W)**		
											Total	Dynamic	
	Utilisation	Pourcentage	Utilisation	Pourcentage	Utilisation	Pourcentage	Utilisation	Pourcentage	Utilisation	Pourcentage	Utilisation	Utilisation	Pourcentage
Conv1	**1162**	**1.53%**	**4088**	**1.35%**	**732**	**0.12%**	**27**	**0.96%**	**5**	**0.49%**	**1.105**	**0.856**	**77%**
BN1	**259**	**0.34%**	**863**	**0.28%**	**56**	**0.01%**	**6**	**0.21%**	**2**	**0.19%**	**0.251**	**0.009**	**4%**
ReLU1 Maxp1	**102**	**0.13%**	**260**	**0.09%**	**193**	**0.03%**	**0**	**0%**	**0**	**0%**	**0.270**	**0.028**	**10%**
Conv2	**9645**	**12.71%**	**33621**	**11.07%**	**4951**	**0.82%**	**216**	**7.71%**	**36.5**	**3.54%**	**0.711**	**0.462**	**65%**
BN2	**267**	**0.35%**	**867**	**0.29%**	**55**	**0.01%**	**6**	**0.21%**	**2**	**0.1%**	**0.251**	**0.009**	**3%**
ReLU2 Maxp2	**61**	**0.08%**	**214**	**0.07%**	**131**	**0.02%**	**0**	**0%**	**0**	**0%**	**0.255**	**0.013**	**5%**
Conv3	**17789**	**23.44%**	**62869**	**20.71%**	**8838**	**1.46%**	**432**	**15.43%**	**72.5**	**16.04%**	**1.138**	**0.884**	**78%**
BN3	**264**	**0.35%**	**868**	**0.29%**	**53**	**0.01%**	**6**	**0.21%**	**2**	**0.19%**	**0.251**	**0.009**	**4%**
ReLU3 Maxp3	**305**	**0.09%**	**198**	**0.07%**	**100**	**0.02%**	**0**	**0%**	**0**	**0%**	**0.254**	**0.012**	**5%**
Conv4	**34724**	**47.75%**	**124136**	**40.89%**	**17652**	**2.91%**	**854**	**30.86%**	**144.5**	**14.03%**	**2.01**	**1.742**	**87%**

BN4	**272**	**0.35%**	**866**	**0.29%**	**50**	**0.01%**	**6**	**0.19%**	**2**	**0.21%**	**0.287**	**0.045**	**16%**
ReLU4	**4**	**<0.01%**	**16**	**0.01%**	**0**	**0%**	**0**	**0%**	**0**	**0%**	**7.874**	**7.566**	**95%**
FC	**1257**	**1.65%**	**4185**	**1.38%**	**380**	**0.06%**	**15**	**0.54**	**20**	**1.94%**	**0.603**	**0.357**	**59%**

Les résultats figurant dans le tableau III.13 montrent que l'utilisation d'une méthode de réutilisation des ressources (calcul en série) permet une significative en ressource et ne consommation d'énergie qui facilite l'implémentation e la totalité du réseau sur un seul circuit FPGA.

III.5.3 Implémentation des circuits de calcul série avec représentation des données en virgule fixe :

Les formats à virgule flottante et à virgule fixe sont des méthodes fréquemment employées pour la représentation des nombres réels dans les systèmes informatiques.

Pour une longueur de 32 bits, le format à virgule flottante suit généralement la norme IEEE 754. Il est constitué de trois composantes principales : le bit de signe (1 bit), l'exposant (8 bits) et la mantisse (23 bits). Ces composantes sont utilisées pour représenter de manière dynamique les nombres réels avec une précision variable et une gamme étendue La figure III.14 représente une donnée en virgule flottante.



Figure III.16 : *Représentation des données en virgule flottante.*

En revanche, pour le format à virgule fixe de n bits, la répartition des bits peut varier en fonction des besoins spécifiques de l'application. Cependant, une configuration courante consiste à allouer m bits pour la partie entière y compris un bit de signe et n-m autres bits pour la partie fractionnaire. La figure III.17 représente une donnée en virgule fixe avec n=32 et m=16. Cette représentation offre une précision constante pour tous les nombres représentés, mais la gamme des valeurs pouvant être représentées est limitée en fonction du nombre de bits alloués.



Figure III.17 : *Représentation des données en virgule fixe*

Le Tableau III.14 montre les ressources FPGA utilisées par les circuits de calcul série et la puissance consommée par les circuits en cas de la représentation des données en virgule fixe du réseau CNN proposés, qui ont été convertis en RTL (niveau de transfert de registre) pour examiner l'impact de la représentation des données en virgule fixe sur les ressources consommées lors de la mise à jour. Fonctionne sur un circuit FPGA Virtex 7XC7VX485T-2FFG1761.

Tableau III.14 : *Ressources du circuit Virtex 7 XC7VX485T-2FFG1761 et puissance consommée par les différentes couches de calcul en série avec les données en virgule fixe.*

	Slice		**LUT**		**FF**		**DSP**		**BRAM**		**POWER(W)**		
											Total	Dynamic	
	Utilisation	Pourcentage	Utilisation	Pourcentage	Utilisation	Pourcentage	Utilisation	Pourcentage	Utilisation	Pourcentage	Utilisation	Utilisation	Pourcentage
Conv1	**124**	**0.16%**	**432**	**0.14%**	**457**	**0.08%**	**9**	**0.32%**	**5**	**0.49%**	**0.594**	**0.349**	**59%**
BN1	**31**	**0.04%**	**60**	**0.02%**	**50**	**0.01%**	**4**	**0.14%**	**2**	**0.19%**	**0.272**	**0.020**	**11%**
ReLU1 Maxp1	**53**	**0.07%**	**135**	**0.04%**	**121**	**0.02%**	**0**	**0%**	**0**	**0%**	**0.260**	**0.018**	**7%**
Conv2	**768**	**0.51%**	**1966**	**0.65%**	**3107**	**0.51%**	**72**	**2.57%**	**36.5**	**3.54%**	**0.276**	**0.030**	**11%**
BN2	**24**	**0.03%**	**52**	**0.02%**	**43**	**0.01%**	**3**	**0.11%**	**0**	**0%**	**0.245**	**0.003**	**1%**
ReLU2 Maxp2	**32**	**0.04%**	**112**	**0.04%**	**86**	**0.01%**	**0**	**0%**	**0**	**0%**	**0.254**	**0.008**	**3%**

Conv3	**1350**	**1.78%**	**3483**	**1.5%**	**5466**	**0.9%**	**144**	**5.14%**	**72.5**	**7.4%**	**0.294**	**0.045**	**15%**
BN3	**21**	**0.03%**	**53**	**0.02%**	**40**	**0.01%**	**2**	**0.07**	**2**	**0.19%**	**0.245**	**0.003**	**1%**
ReLU3 Maxp3	**30**	**0.04%**	**81**	**0.03%**	**67**	**0.01%**	**0**	**0%**	**0**	**0%**	**0.250**	**0.008**	**3%**
Conv4	**2328**	**3.7%**	**4960**	**1.63%**	**8326**	**1.37%**	**288**	**10.29 %**	**144.5**	**14.03 %**	**0.332**	**0.077**	**23%**
BN4	**16**	**0.02%**	**46**	**0.02%**	**36**	**0.01%**	**3**	**0.11%**	**2**	**0.19%**	**0.268**	**0.026**	**10%**
ReLU4	**3**	**0.01%**	**10**	**0.01%**	**0**	**0%**	**0**	**0%**	**0**	**0%**	**5.091**	**4.810**	**94%**
FC	**130**	**0.17%**	**301**	**0.1%**	**265**	**0.04%**	**10**	**0.36%**	**12.5**	**1.21%**	**0.416**	**0.172**	**41%**

En se basant sur les résultats d'implémentation des tableaux III.10 à III.12 on peut faire les remarques suivantes :

- Les circuits de calcul parallèle possèdent l'avantage de rapidité de calcul, mais ils nécessitent un nombre énorme de ressources FPGA et consomment une puissance élevée ce qui rend leur implémentation sur FPGA impossible.
- Parmi toutes les couches du réseau CNN, la couche de convolution est la plus exigeante en ressources et en puissance.
- L'utilisation de la technique de réutilisation de ressources et de la représentation des données en virgule fixe permettent de réduire les ressources FPGA et la consommation de puissance d'une façon très significative. Et cela permet d'implémenter efficacement le réseau CNN proposé sur un seul circuit FPGA.

Conclusion

Ce chapitre représente les différentes étapes que nous avons suivies pour arriver finalement à l'implémentation efficace d'un circuit de classification de cinq classes d'armes sur un circuit FPGA de type Virtex 7 XC7VX485T-2FFG1761. Le premier défi que nous avons rencontré est le manque de base de données. Pour cela, nous avons commencé par collecter des images pour compléter la base de données téléchargée sur internet en suite nous avons fait un prétraitement des images pour unifier leurs dimensions. Après, nous avons développé une architecture qui présente un bon compromis entre justesse de classification et simplicité d'implémentation. Ensuite, nous avons fait la conception des circuits de calcul des différentes couches du réseau CNN en utilisant le logiciel *system generator*. Nous avons fait une conception de calcul parallèle et autre pour le calcul en série réduire l'utilisation des ressources le plus possible dans la carte FPGA.

Après l'étape de conception, nous avons passé à l'étape d'implémentations des circuits de calcul en utilisant le logiciel Vivado 2018. Les résultats d'implémentation montrent un gain très important en ressources et en puissance consommée lorsqu'en utilise le calcul série au lieu du calcul parallèle et la représentation en virgule fixe à la place de la représentation en virgule flottante. À la fin, nous avons fait une vérification sur matériel en utilisant la technique de Co-simulation sur une carte FPGA VC 707 qui contient un circuit FPGA de type Virtex 7 XC7VX485T-2FFG1761.

Conclusion générale

Dans ce projet de fin d'études nous avons effectué une implémentation efficace d'un système de classification automatique d'armes à base de réseaux de neurones convolutifs CNN sur un circuit FPGA.

Nous avons commencé notre travail par une étude théorique approfondie sur les concepts fondamentaux de l'intelligence artificielle IA, de l'apprentissage automatique ML, du Deep Learning et nous nous avons concentré plus précisément sur les réseaux de neurones convolutifs (CNN). Les réseaux CNN forment une classe très importante du Deep Learning où ils ont prouvé leur efficacité dans le domaine de la classification des images. Ils sont constitués essentiellement de couches de convolution, de normalisation BN, d'activation ReLu, de sous échantillonnage MaxPooling et en fin de couches entièrement connectées FC. Nous avons vu que les couches les plus importantes parmi ces couches sont les couches de convolution qui permettent d'extraire les caractéristiques pertinentes qui servent ensuite à faire la classification.

Dans la deuxième partie nous avons réalisé une étude sur les circuits logiques reprogrammables de type FPGA. Cette étude nous a permis de comprendre l'architecture interne de ce type de circuit ainsi que les composants de base de cette architecture. Nous avons aussi compris le principe de fonctionnement, de programmation ainsi que les logiciels relatifs au développement d'une application sur circuit FPGA.

L'étude théorique nous a permis d'acquérir des connaissances précieuses et très utiles pour notre formation sur un domaine technologique très émergeant qui est le domaine du Deep Learning et implémentation sur FPGA.

Dans la partie pratique de notre travail, nous avons collecté et fait le prétraitement des données nécessaires à la conception et à l'évaluation des réseaux CNN pour la classification des armes. Nous avons conçu plusieurs modèles de réseau CNN pour la classification, et finalement, nous avons retenu un modèle simple composé de quatre couches de convolution. Ce modèle a produit des résultats de classification très satisfaisants, avec une précision élevée de 97,11%.

Ensuite, nous avons fait la conception des différents circuits de calcul correspondants aux différentes couches du réseau CNN. Nous avons commencé par la conception de circuits de calcul parallèle mais nous avons constaté que ce type consomme trop de ressources FPGA surtout pour les circuits de convolution. Pour faire face à ce problème nous avons fait appel à des circuits de calcul série qui sont basés sur le principe de réutilisation de ressources. Pour réduire d'avantage les ressource consommées nous avons représenté les données en format virgule fixe au lieu de la virgule flottante.

Enfin, les résultats obtenus montrent que nous avons réussi à limiter l'utilisation des ressources où les résultats d'implémentation démontrent que le réseau ne consomme que 6.6% de slice, 4.22% de LUT, 2.98% de flip-flop, 19.11% de DSP, et 27.24% de BRAM du circuit FPGA Virtex 7 XC7VX485T-2FFG1761. De plus, les circuits proposés permettent également de diminuer la consommation de puissance dynamique du circuit.

Comme perspectives de notre travail on peut citer :

1. Améliorer la justesse de la classification par l'élaboration d'une base de données qui contient des milliers ou des dizaines de milliers d'images par classe.
2. Implémenter l'architecture du réseau sur une plateforme numérique comme FPGA ou ZYNQ.
3. Proposer une architecture pour faire la détection des armes par l'utilisation d'autres techniques basées sur les réseaux CNN (YOLO par exemple).

Bibliographie

[1]…Abd El-Maksoud, A.J., et al. *FPGA Design of High-Speed Convolutional Neural Network Hardware Accelerator*, in 2021 3rd Novel Intelligent and Leading Emerging Sciences Conference (NILES), 2021.

[2]...Huang, S.-C. and T.-H. Le, *Principles and labs for deep learning*: Academic Press, 2021

[3]…M. Benbrahim. *Introduction à l'Intelligence Artificielle*, Université Mostefa Ben Boulaide Batna, 2020

[4]…MAHESH, Batta. *Machine learning algorithms-a review*, International Journal of Science and Research (IJSR).[Internet], 2020, vol. 9, p. 381-386.

[5]...Mohammed, M., M.B. Khan, and E.B.M. Bashier, *Machine learning: algorithms and* applications*,* Crc Press, 2016.

[6]…AMANULLAH, Mohamed Ahzam, HABEEB, Riyaz Ahamed Ariyaluran, NASARUDDIN, Fariza Hanum, *et al. Deep learning and big data technologies for IoT security*, Computer Communications, 2020, vol. 151, p. 495-517.

[7]…Géron, A., *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow: Concepts, tools, and techniques to build intelligent systems* ", O'Reilly Media, Inc, 2019.

[8]...JONES, M. Tim. *Deep learning architectures and The rise of artificial intelligence.* IBM DeveloperWorks, Armonk, 2017.

[9]… H. Sak, A. Senior, F. Beaufays, *Long short-term memory recurrent neural network architectures for large scale acoustic modeling*, in: Fifteenth Annual Conference of the International Speech Communication Association, 2014.

[10]…Bisson, Valentin, *Algorithmes d'apprentissage pour la recommandation*, thèse de doctorat, Université de Montréal, 2012.

[11]…BOUBAYA, Semail, BERBIT, Djamel, *et al. Deeplearning pour la segmentation d'images*, Thèse de doctorat. university of M'sila, 2021.

[12]…ELMOHRI Mohamed Abdessalem, TOBBAL Abdelhafid, *Implémentation d'une méthode d'extraction des caractéristiques des signaux vibratoires sur une plateforme FPGA*, 2021.

[13]…https://coursee.org/blog/artificial-intelligence/neural-network-implementation-in-python/.

[14]. LI, Fei-Fei, KARPATHY, Andrej, et JOHNSON, Justin, *Convolutional neural networks for visual recognition*, 2015.

[15]…YAHIAOUI, ABDENNOUR. *Système de Discrimination Visages/Faux Visages par Réseaux de Neurones Convolutifs (CNN).* 2021.

[16]...IOFFE, Sergey et SZEGEDY, Christian. Batch normalization: *Accelerating deep network training by reducing internal covariate shift*. , International conference on machine learning. Pmlr, pp. 448-456, 2015.

[17]…BEN LAZREG, Mohsen. *Recherche de l'information dans les réseaux de neurones convolutifs pré-entraînés*, Thèse de doctorat. École de technologie supérieure, 2020.

[18]…EL HABIB MOUSSA et CHAHBOUB Mohammed Ridha, *Implémentation efficace d'un réseau de neurones convolutif (CNN) sur circuit FPGA*. Mémoire de master, université de Médéa, 2022.

[19]…BERKANI Mohamed Rafik Aymene, ACHIR Zahra, *Etude et implémentation d'un système connecté intelligent pour la surveillance de l'état de santé des patients souffrant d'arythmie cardiaque*, Mémoire de master, université de Médéa, 2022.

[20]…Mr Mokri Mohammed Zakaria, *Classification des images avec les réseaux de neurones convolutionnels*. Mémoire de master, Université Abou Bakr Belkaid Tlemcen, 2017

*[21]…*Yamashita, R., et al., *Convolutional neural networks: an overview and application in radiology.* Insights into imaging, 9(4): p. 611-629, 2018.

*[22]…*Gheysen, G., J.-D. Legat, and C. De Vleeschouwer, *Accelerating Convolutional Neural Networks for FPGA using depthwise separable convolution and pruning*, mémoire de Master, Ecole polytechnique de Louvain, 2020.

[23]…Redouane ZAHIRA,Aboubakre BELARBI, *implémentation des réseau de neurones de type FFANN avec fonction seuil sur circuit FPGA*, mémoire de PFE d'ingénieur ,septembre 2007.

[24]…MAUNDER, Anurag, RAYES, Ammar, et MIN, Paul S. *Analysis of routing policies in broad-band networks*, Canadian Journal of Electrical and Computer Engineering, vol. 20, no 3, p. 127-137, 1995.

[25]…BAILEY, Donald G. *Design for embedded image processing on FPGAs*. John Wiley & Sons, 2011.

[26] …Liu, S., et al., *Robotic computing on FPGAs*, Synthesis Lectures on Computer Architecture, 16(1): p. 1-218, 2021

[27] …HAJJI, Bekkay, MELLIT, Adel, et BOUSELHAM, Loubna. *A Practical Guide for Simulation and FPGA Implementation of Digital Design*. Springer, 2022.

[28]…CHEKIRED, Fathya. *Etude et implémentation d'une commande MPPT neuro-floue sur FPGA*. 2008. Thèse de doctorat. Ecole nationale polytechnique.

[29] Xilinx, *7 Series DSP48E1 Slice User Guide UG479 (v1.10)* , 2018.

[30]…SAPTONO, Debyo. *Conception d'un outil de prototypage rapide sur le FPGA pour des applications de traitement d'images,* Thèse de doctorat, Dijon, 2011.

[31] …NAZIM, Houadj Mohamed. *Synthèse d'un module de cryptage RSA sur un circuit de type FPGA*, Thèse de doctorat, Université Mouloud Mammeri, 2014

[32] …FAROOQ, Umer, MARRAKCHI, Zied, MEHREZ, Habib, *et al. FPGA architectures: An overview. Tree-Based Heterogeneous FPGA Architectures: Application Specific Exploration and Optimization*, p. 7-48, 2012.

[33]… PRZYBUS, Brent, *Xilinx redefines power, performance, and design productivity with three new 28 nm fpga families: Virtex-7, kintex-7, and artix-7 devices*, Xilinx White Paper, 2010.

[34] ... Xilinx, *Virtex-7 FPGA Overview*, 2013.

[35] … Xilinx, *Zynq-7000 All Programmable SoC Overview*, 2012.

[36]… https://docs.xilinx.com/v/u/en-US/ug570-ultrascale-configuration

[37]… HASSEN, M. Néjib. *Conception d'un système de reconnaissance des images et application des FPGA à l'accélération de l'algorithme d'apprentissage profond*, Thèse de doctorat, Université de Sousse, 2021.

[38]…Xilinx, *Vivado Design Suite User Guide Getting Started UG910 (v2018.2)*, 2018

[39] ….EZZEDDINE, Mazen. *Modélisation et réalisation de la couche physique du système de communication numérique sans fil, WiMax, sur du matériel reconfigurable*, mémoire de Master, université de Montreal, 2009.

[40] … https://github.com/vkaya24100/Different-Weapon-Types-datasets consulté le 04/05/2023.