

République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Université A. MIRA - BEJAIA



Faculté de Technologie
Département de Génie Electrique

Projet de Fin d'Etudes

Pour l'obtention du diplôme de Master en Automatique et informatique industrielle

Thème

Implémentation de réseaux de neurones profonds à l'aide d'un processeur ARM Cortex-M

Préparé par :
ALILOUCHE Mohamed
KASMI Salah
Loubar Salim

Dirigé par :
Mme MEZZAH Samia

Année Universitaire : 2019/2020

REMERCIEMENTS :

Nous rendons grâce à Dieu qui nous a donné l'aide, la patience et le courage pour accomplir ce travail.

On tient à adresser nos plus vifs remerciements à Mme MEZZAH Samia, pour nous avoir encadré et pour les recommandations qu'elle nous a prodiguées et qui nous ont été d'un grand apport.

Nous tenons à adresser nos plus vifs remerciements aux membres de jury, pour avoir accepté d'évaluer notre travail.

On tient à remercier nos familles respectives, surtout nos parents pour nous avoir donnés tous les moyens nécessaires afin d'accomplir nos études.

Sans oublier nos proches amis, et nos camarades qui nous aidés et encouragés, tout au long de ce cursus d'études.

Merci à tous ceux qui ont contribué à l'élaboration de ce travail de près ou de loin et qui méritent d'y trouver leurs noms.

Table des matières

REMERCIEMENTS :	1
Table des matières	i
Liste des Figures	iv
Liste des abréviations :	vi
INTRODUCTION GENERALE	1
Chapitre I : GENERALITES SUR LE DEEP LEARNING	2
I.1. Introduction	3
I.2. L'Intelligence Artificielle	3
I.3. Evolution du volume de données	3
I.4. Machine Learning	4
I.4.1. Définition	4
I.4.2. Applications	4
I.5. Deep Learning	4
I.6. Apprentissage supervisé et non-supervisé	5
I.6.1. Apprentissage supervisé	5
I.6.2. Apprentissage non supervisé	6
I.7. Réseaux de neurones	7
I.8. Apprentissage par descente de gradient	9
I.9. Apprentissage classique vs profond	11
I.10. Limites des réseaux de neurones :	11
I.10.1. Problématique du gradient évanescent liée aux réseaux profonds	11
I.10.2. Autres Problématiques liées aux réseaux profonds	12
I.11. Les principaux types de réseaux de neurones profonds	12
I.11.1. Les CNN	12
I.11.2. Les RNN	12
I.11.3. Les Auto-encodeurs	14
I.12. Exemples d'applications des réseaux de Deep learning	14
I.13. Conclusion	14
Chapitre II : CNN et Classification des Images	15
II.1. Introduction	16
II.2. Intérêt de la classification des images	16
II.3. Méthodes de Classifications	17

II.3.1. Méthode Supervisée	17
II.3.2. Méthode Non-Supervisée	18
II.4. Fonctionnement des CNN	18
II.5. Architecture globale d'un réseau convolutionnel	21
II.5.1. Les couches convolutionnelles	21
II.5.2. Couche de Pooling	22
II.5.3. Couche Batch Normalization	22
II.5.4. Couche Fully Connected	23
II.6. Conclusion.....	23
Chapitre III : La carte de développement STM32 Discovery	24
III.1. Intoduction	25
III.2. Les processeurs ARM	25
III.2.1. Les Fonctionnalités du Processeur ARM	25
III.2.2. Les différentes catégories de processeurs ARM.....	26
III.2.2.1. Profil A.....	27
III.2.2.2. Profil R.....	27
III.2.2.3. Profil M	27
III.3. La carte stm32F411.....	27
III.3.1. Performances	28
III.3.2. Efficacité énergétique	28
III.3.3. Intégration.....	29
III.4. Outil de développement	30
III.4.1. Bibliothèques CMSIS	30
III.4.2. La bibliothèque CMSIS-NN	31
III.4.3. Quantification en virgule fixe	31
III.4.4. Classification des fonctions	32
III.4.5. CIFAR 10	32
III.5. Conclusion	33
Chapitre IV : Implémentation d'un CNN sur la carte STM32.....	34
IV.1. Introduction	35
IV.2. L'architecture du CNN à implémenter	35
IV.2.1. Différentes fonctions du programme	35
IV.2.2. Implémentation des couches du CNN.....	36
IV.2.3. Description	38
IV.2.4. Technique de programmation.....	38
IV.2.5. Execution du programme.....	39
IV.3. Conclusion	40

CONCLUSION GENERALE41

Liste des Figures

Figure I-1: Evolution de la production de données [2]	3
Figure I-2: Deep learning une partie du machine learning et de l'IA	5
Figure I-3: Schéma de fonctionnement de l'apprentissage supervisé	5
Figure I-4: Extraction de caractéristiques dans l'apprentissage non supervisé	6
Figure I-5: Neurone biologique.....	7
Figure I-6: Neurone artificiel	7
Figure I-7: Réseau de neurones artificiels.....	8
Figure I-8: Réseau de neurones récurrent.....	8
Figure I-9: Graph montrant la meilleure valeur de l'erreur	9
Figure I-10: Itérations pour trouver le minimum global (poids idéal).....	10
Figure I-11: Méthode du gradient stochastique	10
Figure I-12: Apprentissage classique	11
Figure I-13: Apprentissage profond.....	11
Figure I-14: Fonction $\tanh(x)$	11
Figure I-15: Image de réseau de neurones naïf	12
Figure I-16: Fonctionnement d'un neurone récurrent	13
Figure I-17: Image montrant le fonctionnement d'un auto-encodeur.....	14
Figure II-1: Fonctionnement du filtre	18
Figure II-2: Filtre qui supprime les lignes verticales	19
Figure II-3: Filtre qui supprime les lignes horizontales	19
Figure II-4: Fonction Max pooling 2x2	20
Figure II-5: Résultat de l'application du Max pooling 2x2	20
Figure II-6: Architecture d'un CNN	21
Figure II-7: Fonction softplus	22
Figure II-8: Plusieurs filtres donnent plusieurs cartes en sortie.....	22
Figure III-1: Différents types de processeurs arm	27
Figure III-2: La carte stm32.....	28
Figure III-3: Disposition supérieure.....	29

Figure III-4: Eléments du CMSIS	31
Figure III-5: Diagramme fonctions CMSIS-NN	32
Figure III-6: Exemples d'images CIFAR-10 appartenant de chaque catégorie	33
Figure IV-1: l'architecture du réseau de neurones à implémenter	35
Figure IV-2: Image A.....	39
Figure IV-3: Image B.....	39
Figure IV-4: résultats du premier test	39
Figure IV-5: Résultats du deuxieme test.....	40

Liste des abréviations :

IDE : Integrated Development Environment

CNN : Convolutional Neural Network

RNA : Réseau de Neurones Artificiels

RNN : Recurrent Neural Network

LSTM : Long Short-Term Memory

IA : Intelligence Artificielle

MLP : Multi Layer Perceptron

PMC : Perceptron multicouche

RISC : Reduced instructions set computer

CISC : Complex instruction set computer

BAM : Batch Acquisition Mode

ART : Adaptive real time

SWD : serial wire debugging

CMSIS : Cortex Microcontroller Software Interface Standard

CIFAR : canadian institute for advanced research

SIMD : single instruction multiple data

DSP : digital signal processing

USART : Universal Synchronous/Asynchronous Receiver/Transmitter

SPI : Serial Peripheral Interface

I2S : Inter-IC Sound

I²C : Inter-Integrated Circuit

SDIO : Secure Digital Input Output

OTG : On-The-Go

INTRODUCTION GENERALE

Le phénomène de l'intelligence artificielle et l'intérêt porté aux robots et machines ne date pas d'hier. Imiter les aspects de l'intelligence humaine et les implémenter dans une machine était le but que les chercheurs avaient toujours ciblé afin de porter des solutions aux problèmes quotidiens. Les principaux problèmes rencontrés aujourd'hui découlent du flux important de données dans le monde. Ce flux augmente de manière exponentielle chaque année, ce qui rend leur traitement et leur exploitation très difficiles. Pour résoudre ce problème, les recherches scientifiques se sont concentrées sur de nouvelles solutions plus intelligentes en terme d'aptitude avancée d'apprentissage automatique, ce qui a conduit au développement de nombreuses structures de réseaux de neurones basées sur le concept du deep learning. Plus encore, les avancées technologiques dans le domaine des systèmes embarqués ont permis le déploiement des réseaux de deep learning dans plusieurs domaines, de la médecine, aux voitures, en passant par les appareils grand public tels que les smartphones, les tablettes et autres.

L'implémentation matérielle de réseaux de neurones profonds était réservé aux machines de calculs puissantes, mais grâce à l'avancée dans son développement, les réseaux de neurones, maintenant, peuvent tenir dans des circuits à faible puissance de calculs, on peut citer en exemple Advanced RISC Machines(ARM) qui produit des microprocesseurs qui permettent d'utiliser cette intelligence artificielle et la rendent plus accessible, ce qui donne aux développeurs et constructeurs, des outils puissants qui aident à mieux maîtriser cette technologie, et proposer des produits intelligents. Dans ce cadre, notre projet consiste à implémenter un réseau de deep learning dans une carte STM32F411 qui marche avec un microcontrôleur ARM, on prendra en exemple la reconnaissance d'images.

Ce mémoire est organisé en quatre chapitres. Nous présentons dans le chapitre I de ce mémoire, un tour d'horizon du domaine de l'intelligence artificielle et ses différentes branches, le chapitre II présentera principalement les réseaux de neurones convolutifs, leurs principes de fonctionnement en application au domaine de la reconnaissance et la classification d'images. Les chapitres III et IV sont consacrés à la description des outils et des étapes l'implémentation matérielle d'un réseau de neurones convolutif avec le microcontrôleur ARM choisi.

Chapitre I : GENERALITES SUR LE DEEP LEARNING

I.1. Introduction

L'intelligence artificielle est une technologie révolutionnaire, elle a donné naissance à plusieurs techniques qui sont utilisées globalement dans plusieurs domaines des sciences et technologies. Dans ce qui va suivre, nous présenterons les différentes techniques et technologies, quelques exemples d'utilisation, ainsi que le principe de leur fonctionnement.

I.2. L'Intelligence Artificielle

La définition à l'Intelligence Artificielle(IA) la plus populaire est celle donnée par l'un de ses créateurs Marvin Lee Minsky comme « la construction de programmes informatiques qui s'adonnent à des tâches qui sont, accomplies de façon plus satisfaisantes par des êtres humains car elles demandent des processus mentaux de haut niveau tels que : l'apprentissage perceptuel, l'organisation de la mémoire et le raisonnement critique » [1].

I.3. Evolution du volume de données

De nos jours, là où nous allons on trouve des données, que ça soit en envoyant ou recevant des messages, en utilisant les données cellulaires ou en activant le gps.

Le volume de ces données augmente d'une manière exponentielle, tel que l'homme a produit en 2005 environ 130 exabytes de données (1 exabytes = 1 000 000 000 000 000 bytes), en 2010 plus de 1200 exabytes, 7900 exabytes en 2015 et plus de 40900 exabytes en 2019 (donc plus de 40 petabytes) [2].

Cet énorme volume de données représente une réelle mine d'or, mais malheureusement les experts en mégadonnées ainsi que les machines ne peuvent pas tout traiter en utilisant les méthodes traditionnelles comme on peut le voir sur la figure I-1. C'est là qu'entre en jeu une technologie de pointe permettant la meilleure utilisation de ces données, qu'on appelle l'apprentissage automatique ou le machine learning.

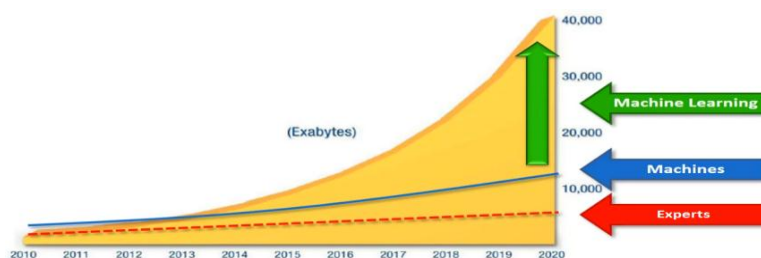


Figure I-1: Evolution de la production de données [2]

I.4. Machine Learning

I.4.1. Définition

Apprentissage automatique en français, est une branche importante de l'intelligence artificielle. Arthur Samuel, un pionnier dans ce domaine l'a défini comme « le domaine d'étude qui donne aux ordinateurs la capacité d'apprendre sans être explicitement programmé » [3].

L'apprentissage automatique se fait en deux étapes. La première est la phase d'apprentissage où y aura une extraction du maximum d'informations des données pour avoir un modèle. La deuxième est la phase de déploiement, une fois le modèle déterminé, de nouvelles données sont introduites afin de réaliser la tâche souhaitée.

I.4.2. Applications

De nos jours, cette technologie est présente dans divers domaines, notamment dans la reconnaissance faciale utilisée par facebook lorsque nous publions une photo nos amis sont tagués automatiquement, la saisie en utilisant la voix disponible sur tous les smartphones de nos jours, les robots marcheurs qui apprennent seuls, les sites internet qui collectent nos données afin d'affiner leur système de recommandation, comme le machine learning est utilisé dans le domaine médical afin d'assister au mieux les spécialistes, et tant d'autres utilisations.

I.5. Deep Learning

Apprentissage profond en français, est une branche importante du machine learning. Cette technologie repose sur les réseaux de neurones artificiels qui imitent le fonctionnement de notre cerveau. Elle est capable d'apprendre des données brutes qu'on lui donne sans intervention humaine. C'est grâce aux grandes quantités de données et la puissance de calcul que cette technologie a commencé à donner de bons résultats et a commencé à être utilisée.

Le deep learning fait partie d'une autre technologie plus vaste appelée machine learning qui fait partie à son tour du domaine de l'intelligence artificielle (IA) (figure I-2).

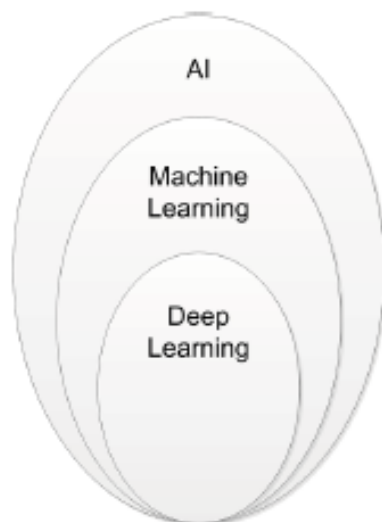


Figure I-2: Deep learning une partie du machine learning et de l'IA

I.6. Apprentissage supervisé et non-supervisé

I.6.1. Apprentissage supervisé

L'apprentissage supervisé permet à l'algorithme de s'entraîner grâce à des données d'apprentissage étiquetée appelées *Training set*, c'est-à-dire chaque donnée d'entrée possède son résultat de sortie, ceci l'aidera à établir des règles de comportements qu'il va prendre pour prédire automatiquement les résultats des entrées aléatoires qu'on utilisera comme test appelées *Test set*, et dans cette catégories on distingue deux cas, la régression quand on parle de données à valeurs continues (exemple :) et la classification quand on parle de données à valeur discrètes (exemple : Images) [4]. La figure I-3 , résume globalement comment marche l'approche supervisée.

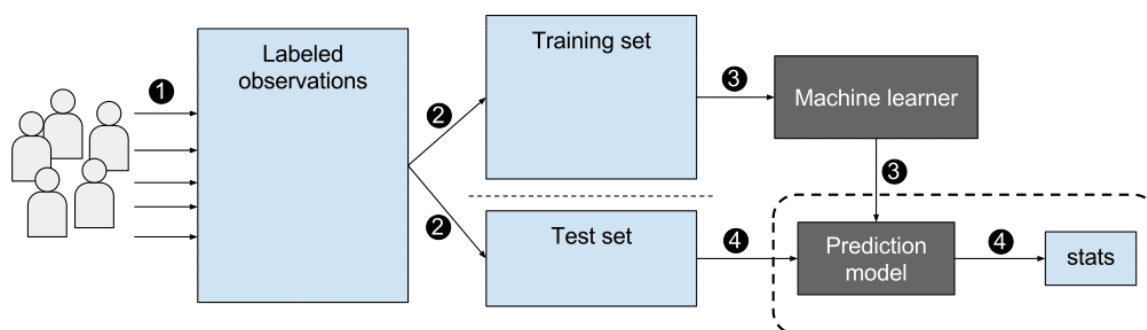


Figure I-3: Schéma de fonctionnement de l'apprentissage supervisé

I.6.2. Apprentissage non supervisé

L'apprentissage non supervisé, est une approche qui n'utilise pas de données étiquetées, l'algorithme va procéder par regroupement des observations en groupes homogènes qui possèdent les mêmes caractéristiques comme montré sur la figure I-5, cela lui servira donc comme entraînement pour ensuite arriver à faire de même pour les données utilisées comme test. [4]

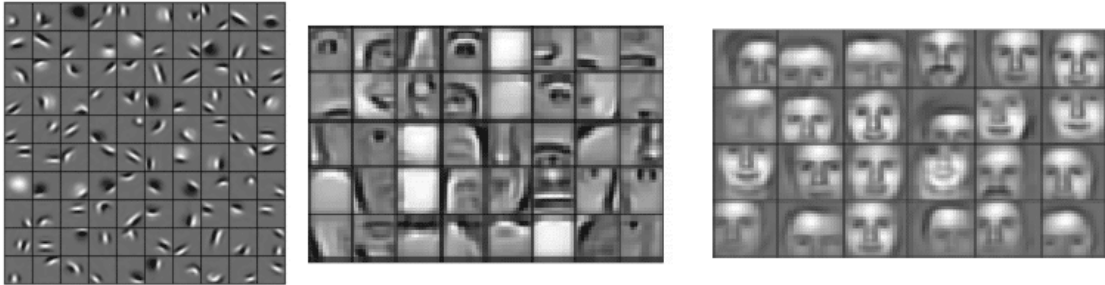


Figure I-4: Extraction de caractéristiques dans l'apprentissage non supervisé

Exemple :

Pour un réseau de neurones qui va différencier entre un chat et un chien en utilisant les deux manières précédentes, on aura deux issues complètement différentes :

- En utilisant l'apprentissage supervisé, il faut impérativement montrer des photos étiquetées où on précise chaque caractéristique propre au chat et au chien, par exemple, la forme d'oreille, des yeux, du nez et ainsi de suite, la distinction se fera grâce à ces étiquettes.
- Tandis qu'en utilisant l'apprentissage non-supervisé, on lui montre une multitude de photos de chats et de chiens, où l'intelligence artificielle apprend d'elle-même les caractéristiques de chaque animal, quand le réseau de neurone artificiel sera parfaitement entraîné, il arrivera à distinguer entre ces deux animaux, une fois terminé, il est nécessaire que des experts trouvent le nom des étiquettes, car si les groupes sont formés, ils ne sont pas nommés par l'algorithme.

Il existe d'autres méthodes d'apprentissage complètement différentes tel que l'apprentissage par renforcement qui a consisté à tester tous les cas possibles afin de trouver la solution optimale [5].

I.7. Réseaux de neurones

En s'inspirant du cerveau humain (figure I-5), on a créé des algorithmes qui reprennent son architecture neuronale où les synapses assurent les connexions avec les autres neurones, les dendrites sont les entrées, les axones sont les sorties, et enfin le noyau qui active les sorties selon les stimulations en entrées (figure I-6) [6].

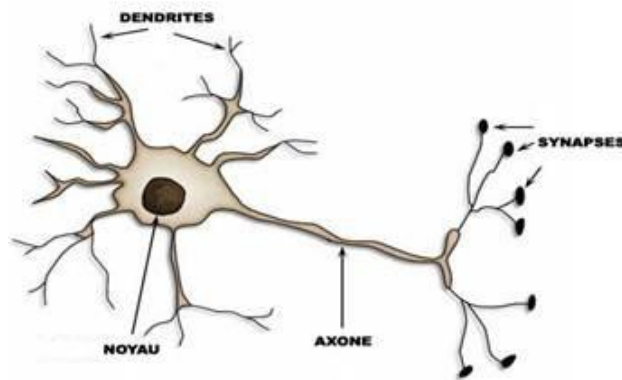


Figure I-5: Neurone biologique

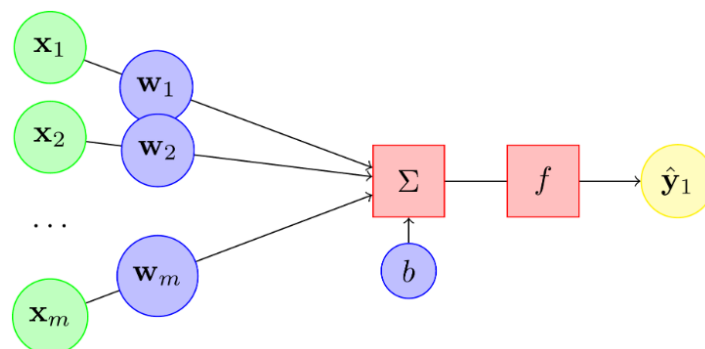


Figure I-6: Neurone artificiel

Où on a :

- Des **entrées**, notées x sous formes de vecteurs, représentant les dendrites ;
- Une **sortie**, notée y , représentant l'axone ;
- Des **paramètres**, notés w (poids) et b (biais), influençant le fonctionnement du neurone.

Équation d'un neurone formel :

$$\hat{y} = f(\langle w, x \rangle + b)$$

Chaque entrée est multipliée par un poids (un coefficient) w . Toutes les entrées sont additionnées à un biais b . Le résultat de la somme passe à travers une fonction de transfert f pour produire la sortie voulue.

Cependant, un seul neurone ne permet pas de répondre à des problèmes complexes [5]. Un réseau neuronal est l'association, en un graphe plus ou moins complexe, d'objets élémentaires, les neurones formels (figure I-7). Les principaux réseaux se distinguent par son organisation en couches, le nombre de neurones, récurrence, type d'apprentissage [6].

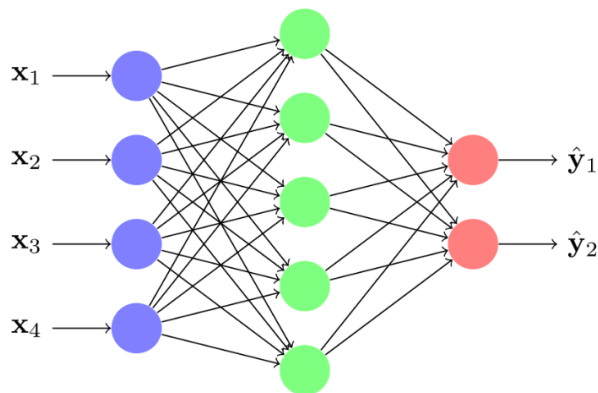


Figure I-7: Réseau de neurones artificiels

L'information va toujours des couches d'entrées aux couches de sorties. Ces réseaux peuvent être appris par descente de gradient. Ils sont adaptés aux données de tailles fixes, comme des images. Ils portent le nom de *perceptron multicouche (PMC)*, *Feed-Forward* ou *Multi Layer Perceptron (MLP)* en anglais.

Pour les données de tailles variables ou les séquences, on utilise les réseaux récurrents (figure I-8). Dans un réseau récurrent, il existe au moins un neurone qui reboucle vers sa propre couche ou une couche précédente [5].

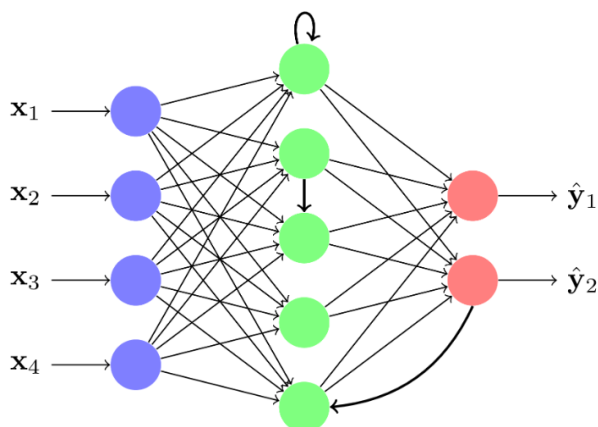


Figure I-8: Réseau de neurones récurrent

I.8. Apprentissage par descente de gradient

Les réseaux de neurones artificiels, apprennent par principe de propagation de données, les données sont propagées depuis l'entrée, puis activées grâce à une fonction d'activation, pour enfin avoir en sortie un résultat primaire.

Une fois le résultat obtenu, on le compare avec le résultat désiré, une erreur alors peut être calculée grâce à ce qu'on appelle la fonction objectif ou cost function en anglais qui est comme suite :

$$C = \sum \frac{(\hat{y} - y)^2}{2}$$

\hat{y} = La sortie du perceptron

y = La sortie ou le resultat désiré

Plusieurs autres fonctions existent, mais cette dernière demeure la plus commune et la plus utilisée.

Une fois l'erreur calculée, on la réinjecte dans le réseau, ce qui s'appelle la rétropropagation, cela nous permettra par la suite d'ajuster les poids de notre réseau de telle sorte à réduire la valeur C, plus cette valeur est petite plus notre réseau est précis et proche du comportement désiré.

L'ajustement du réseau se fait de plusieurs manières, la plus directe est de choisir arbitrairement plusieurs valeurs de poids, les utiliser dans le réseau, et calculer l'erreur à chaque reprise, le poids qui nous donne la valeur d'erreur la plus faible, est celui qui le plus proche du poids idéal.

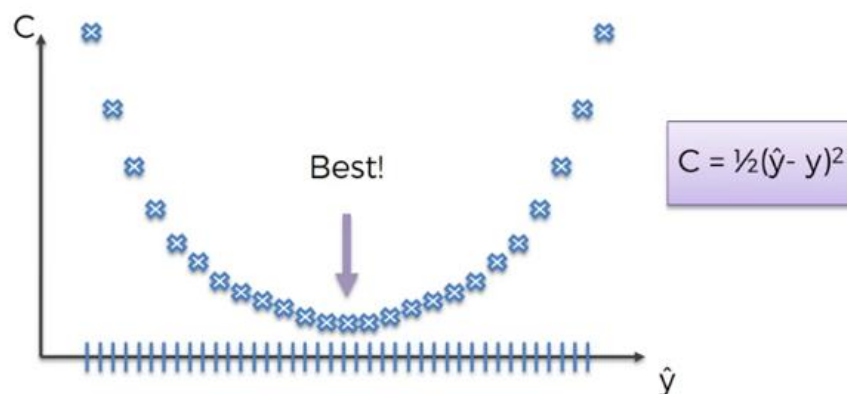


Figure I-9: Graph montrant la meilleure valeur de l'erreur

Cette solution a l'air simple, mais demande un temps énorme, et des ressources gigantesques lorsqu'on aura à faire un réseau avec une multitude de poids, ce qui nous fait entrer dans ce qu'on appelle le fléau de dimension, c'est-à-dire on sera engloutis dans des calculs interminables.

La méthode de « Décence de gradient » permet de simplifier les calculs et raccourcir considérablement le temps nécessaire à les trouver, son principe est de prendre de manière arbitraire un point du graph de la figure I-10, calculer sa pente afin déterminer la position du point suivant, et de trouver très rapidement le minimum global de la fonction objectif, ce minimum qui est la valeur idéale du poids.

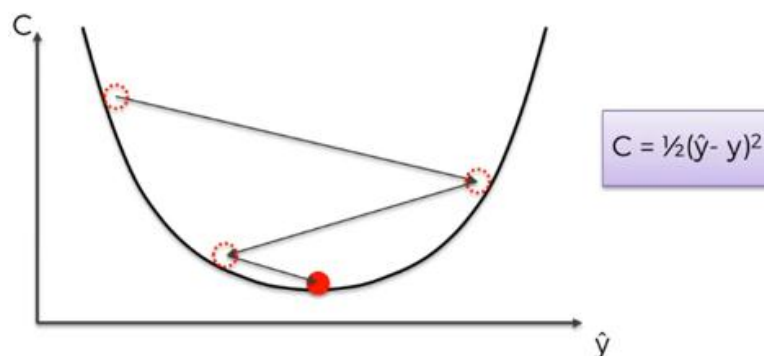


Figure I-10: Itérations pour trouver le minimum global (poids idéal)

Cette méthode en revanche se limite seulement à des fonctions dites convexes, ou à un seul minimum global, pour les fonctions non-convexes, cette méthode de descente de gradient n'est pas valable car cela pourrait nous induire à l'erreur et trouver un minimum local au lieu du global, dans ce cas on va alors s'armer d'un autre outil qui s'appelle "gradient stochastique" ou "Stochastic Gradient Descent" (figure I-11), cette dernière permet de résoudre le problème du minimum [7].

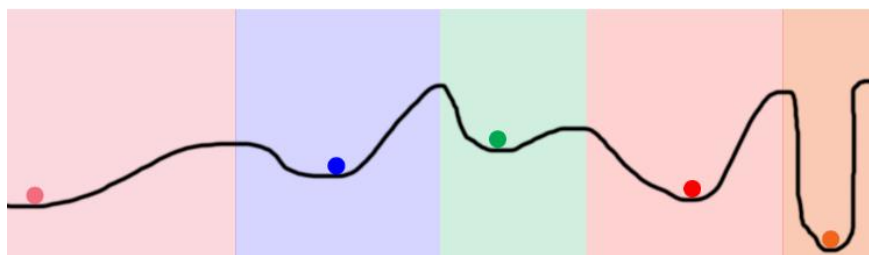


Figure I-11: Méthode du gradient stochastique

I.9. Apprentissage classique vs profond

Dans un apprentissage supervisé classique, on extrait d'abord des caractéristiques des données brutes, puis on les classe. Seule la dernière phase est apprise de façon supervisée, l'extraction de caractéristiques est statique.



Figure I-12: Apprentissage classique

Dans un apprentissage profond, on va directement travailler sur les données brutes. La phase d'extraction des caractéristiques est elle aussi apprise, et n'est plus statique.

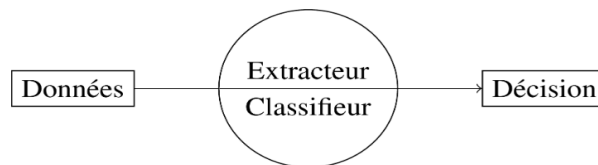


Figure I-13: Apprentissage profond

On peut multiplier les couches pré-apprises et les couches du PMC pour des problèmes de grandes dimensions. On appelle alors le réseau de neurones un réseau de neurones profond ou Deep Neural Network (DNN).

I.10. Limites des réseaux de neurones :

I.10.1. Problématique du gradient évanescant liée aux réseaux profonds

On remarque très vite que plus nous avons un grand nombre de couche, plus l'apprentissage par rétropropagation du gradient est difficile. On appelle ce phénomène Gradient évanescant. En effet, lorsque les unités des couches supérieures sont saturées, peu de gradient est transmis aux couches précédentes. Par exemple sur la figure I-14, pour la fonction \tanh en dessous de ($x = -4$) et au-dessus de ($x = +4$), le gradient est très faible [5].

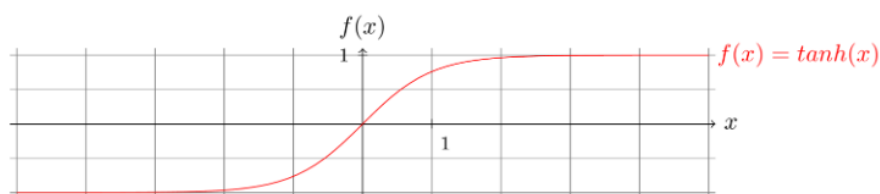


Figure I-14: Fonction $\tanh(x)$

I.10.2. Autres Problématiques liées aux réseaux profonds

Un problème très commun avec les réseaux profonds est une implémentation naïve de ce dernier pour une image, où chaque pixel est relié à tous les neurones de la première couche. Il en résulte un grand nombre de paramètres, qui sont difficilement ajustable par descente de gradient [5].

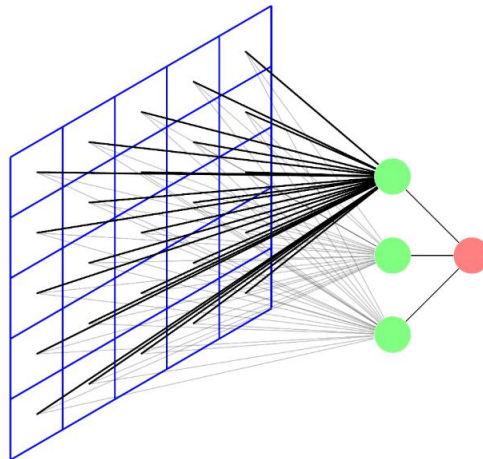


Figure I-15: Image de réseau de neurones naïf

Ainsi que pour que cette méthode donne de bons résultats, le sujet doit être le seul élément sur l'image et au centre de celle-ci. C'est là qu'entre en jeu les réseaux neuronaux convolutionnels.

I.11. Les principaux types de réseaux de neurones profonds

I.11.1. Les réseaux de neurones convolutifs

L'idée derrière un réseau de neurone convolutifs est de filtrer les images avant d'entraîner le réseau, après avoir filtré les images, les caractéristiques de celles-ci peuvent être plus prééminentes, et on peut ensuite les repérer pour identifier quelque chose [8] (nous allons plus détailler dans le chapitre II). Cette technologie est très efficace dans le traitement de données de tailles fixes tel que les photos.

I.11.2. Les réseaux de neurones récurrents

Pour résoudre les problèmes des CNN, qui était l'oubli des données précédentes, une nouvelle forme de réseaux de neurones a été mise en œuvre, et qui s'appelle les réseaux de neurones récurrents.

Les réseaux de neurones récurrents ou RNN sont dotés de connexions récurrentes entre ses couches, l'information qui y circule est non seulement propagée en sortie, mais aussi réinjectée en entrée, ce qui lui donne la faculté de se *souvenir* d'un élément à un instant $t-1$ en plus de *reconnaitre* les éléments à un instant t , ceci procure aux RNN une mémoire à court terme [9].

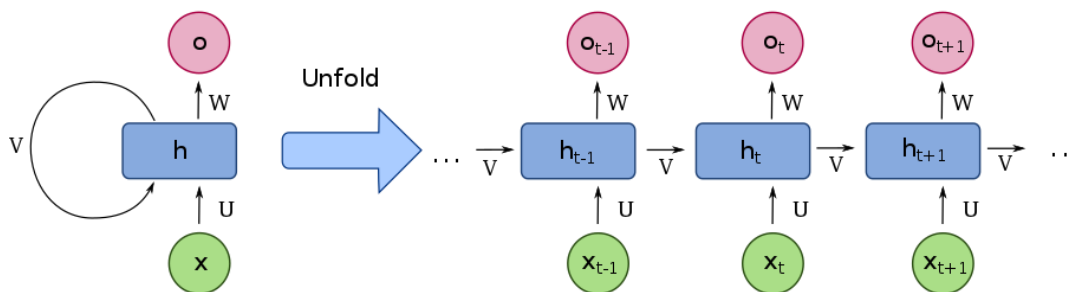


Figure I-16: Fonctionnement d'un neurone récurrent

Les RNN sont utilisés dans plusieurs domaines, particulièrement là où les données sont à taille variable, on peut citer comme exemples, la reconnaissance de paroles, traductions automatiques et autres [10].

L'utilisation des méthodes de rétropropagation du gradient peut s'avérer efficace mais peut engendrer la *disparition ou explosion du gradient*, ces derniers font *oublier* les données au fil du temps et d'amoindrir l'erreur calculée ce qui va fausser l'apprentissage du réseau, ce qui a conduit à la création d'une forme plus évoluée des RNN à savoir les LSTM (Long short term memory ou mémoire à long et court terme), ces derniers sont dotés d'une *mémoire* qui permet de trier les données, et en utiliser que celle qui sont nécessaires, ces tris sont fait grâce à trois portes principales, à savoir, la porte d'oubli, la porte d'entrée et la porte de sortie. [11]

La porte d'entrée comme son nom l'indique permet de recevoir de nouvelles données et décider desquelles il faut ajouter ou modifier, le tri de ces données se fait par la porte d'oubli, si celles-ci ne sont pas pertinentes elles sont donc supprimées, une fois que le tri et l'ajout de données sont terminés, elle sont donc données en résultat par le biais de la porte de sortie, par ce processus donc les LSTM ont la faculté donc de se *souvenir* ou *oublier* des données selon les besoins de la situation, et selon l'utilisation on en distingue aussi trois types de réseaux ; *One to many*, *Many to one* et *Many to Many* .

I.11.3. Les Auto-encodeurs

Introduits en 1980, les auto-encodeurs sont des réseaux de neurones non supervisés qui sont fait pour apprendre comment compresser et coder des données de manière efficace, et ceci est fait par la réduction des dimensions de données achevée en enlevant le maximum possible de bruit, ils sont composés de deux parties : l'encodeur et le décodeur.

L'encodeur est constitué d'un ensemble de couches de neurones qui sont responsables du traitement de données et de leurs transformations en nouvelles représentations, qui sont à leur tour traitées par les décodeurs, ces derniers font une reconstitution des représentations reçues afin d'essayer d'avoir les données de départ.

Une erreur est alors générée une fois que l'entrée et la sortie soient comparées, et cette dernière sert alors de donnée d'entraînement pour l'auto-encodeur, cette erreur alors permettra alors d'ajuster ses paramètres afin de la réduire. [12]

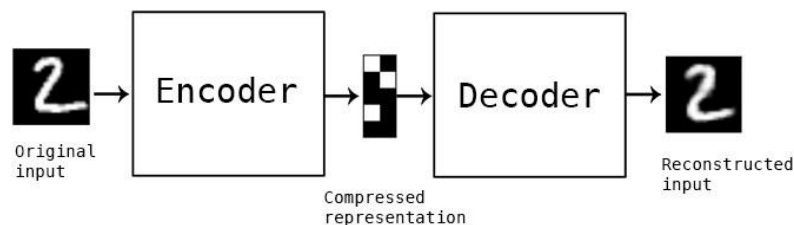


Figure I-17: Image montrant le fonctionnement d'un auto-encodeur.

I.12. Exemples d'applications des réseaux de Deep learning

A l'heure actuelle, la technologie de l'apprentissage profond est présente dans tous les domaines, et parmi les applications qu'on peut citer :

Les voitures autonomes, détection de fraude, traitement du langage et reconnaissance vocale, assistants virtuels, reconnaissance visuelle, coloration des images en noir et blanc, traduction automatique, description du contenu des photos et tant d'autres.

I.13. Conclusion

Dans ce chapitre, nous avons mis en avant les généralités du Deep learning, de son fonctionnement, des principes de bases des réseaux de neurones, des différentes couches qui les composent.

Chapitre II : CNN et Classification des Images

II.1. Introduction

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

Dans ce qui va suivre on explique d'avantage la classification d'image, et comment utiliser le réseaux de neurones convolutif dans ce domaine.

II.2. Intérêt de la classification des images

Les photos sont constamment prises, partagées et cherchées par le monde entier, grâce au développement technologique actuel (Appareils photos numériques, caméscopes, téléphones mobiles...etc.) on a des images a grande définition et à la portée de tout le monde. Ces dernières sont généralement accompagnées par des signatures numériques comme par exemple des descriptions, titres, tags et autres qui permettent à ces photos d'être trouvées, toutes ces données sont alors utilisées dans des procédés de détection et de reconnaissance d'images.

Cette tâche de classier les images est relativement simple et aisée pour l'être humain, en se basant sur ses connaissances et son sens de la vision, il peut reconnaître n'importe quelle forme ou scène, mais là où ça devient compliqué et difficile pour lui, est l'important nombre de données existantes de nos jours, et classier cela, peut s'avérer quasiment impossible et peut demander un temps colossal pour y arriver et ainsi que la répétition exhaustive de la tâche va le drainer rapidement de toute énergie et motivation, et c'est là que la technique de reconnaissance d'image ou classification d'image a été mise au point, elle a pour but de classier ces images dans des catégories selon des caractéristiques données, par exemple si l'image contient une forme ou un élément précis, un animal ou un objet donné et ainsi de suite, et ce grâce à l'évolution de l'intelligence artificielle et au développement du machine Learning.

On retrouve dans ce domaine deux catégories importantes, la classification de l'image, et la recherche d'image (ou récupération d'images traduit de l'anglais Image retrieval) [13].

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

La recherche ou récupération d'images est un procédé qui nous permet de trouver des images similaires à ce qu'on cherche et existe en deux catégories, selon le texte cherché qui consiste à trouver des images qui comporte les mêmes mots clés employés à la recherche, ou selon un exemple d'image, la recherche donc va nous apporter des résultats similaires au contenu de l'image employé comme exemple de recherche.

Grâce à l'avancée importante faite dans ce domaine de reconnaissance, plusieurs domaines professionnels font appel à cette technologie, on peut citer par exemple :

1. Le domaine médical, les CNN sont utilisés pour détecter des cellules cancéreuses.
2. Reconnaissance de scènes dans les appareils photos et téléphones mobiles.
3. Dans les voitures intelligentes qui peuvent détecter les piétons et l'environnement.
4. Détection d'iris et des visages, à des fins de sécurité.

Des réseaux de neurones convolutifs sont alors utilisés pour cette reconnaissance, et pour qu'ils soient plus précis et performants ils doivent être entraîné avec un grand nombre de données d'images, qui comporteront une description du contenu de celle-ci, par exemple, des mots clés pour décrire son contenu, un titre pour la résumer, et grâce à ça le CNN s'entraînera à détecter et reconnaître des images similaires [14].

II.3. Méthodes de Classifications

Dans ce qui suit on va donc s'intéresser à deux méthodes de classification qui sont les plus principalement utilisées, c'est-à-dire, l'approche supervisée et non-supervisée.

II.3.1. Méthode Supervisée

Cette méthode consiste à classer des données selon des classes et catégories déjà prédéfinies.

Avec l'utilisation de données de référence comme données d'apprentissage (par exemple le cifar10 ou cifar100) [23], qui sont déjà classées en catégories, par exemple : voitures, chat, chien, camion...etc. on pourra créer un modèle qui, quand on l'exposera a une nouvelle

image ou donnée, va prédire à quelle classe ils appartiennent selon les similarités trouvées avec les données utilisées comme référence (apprentissage) on peut citer comme exemple, la reconnaissance d'un chat, avec la forme de ses yeux, ses moustaches ...etc.

II.3.2. Méthode Non-Supervisée

Cette méthode en revanche, ne requière pas un apprentissage au préalable, dans ce cas, c'est au programme d'extraire des caractéristiques spécifiques dans des images et les regrouper selon la similarité, on parle alors de Clustering, qui est défini comme étant de regrouper des données similaires sans l'aide d'étiquetage.

II.4. Fonctionnement des CNN

Un filtre est simplement un ensemble de multiplicateur. Et l'utilisation de ces filtres pour détecter les caractéristiques est appelée étape de convolution [15].

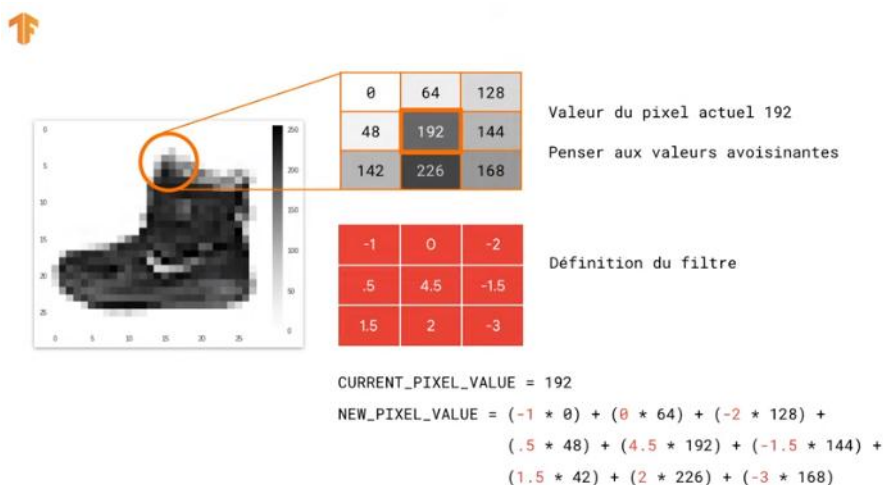


Figure II-1: Fonctionnement du filtre

Dans ce cas par exemple, ce pixel spécifique a la valeur 192, en rouge nous avons le filtre, on multiplie 192 par 4.5 et chaque un de ses voisins par les valeurs respectives du filtre, la nouvelle valeur de ce pixel est la somme de tous ces résultats.

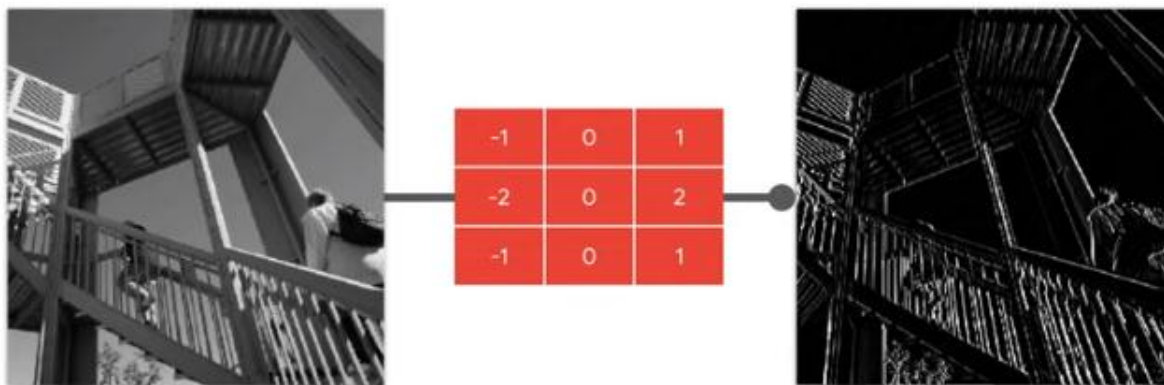


Figure II-2: Filtre qui supprime les lignes verticales

Avec ce filtre par exemple, ne garde que les lignes verticales.

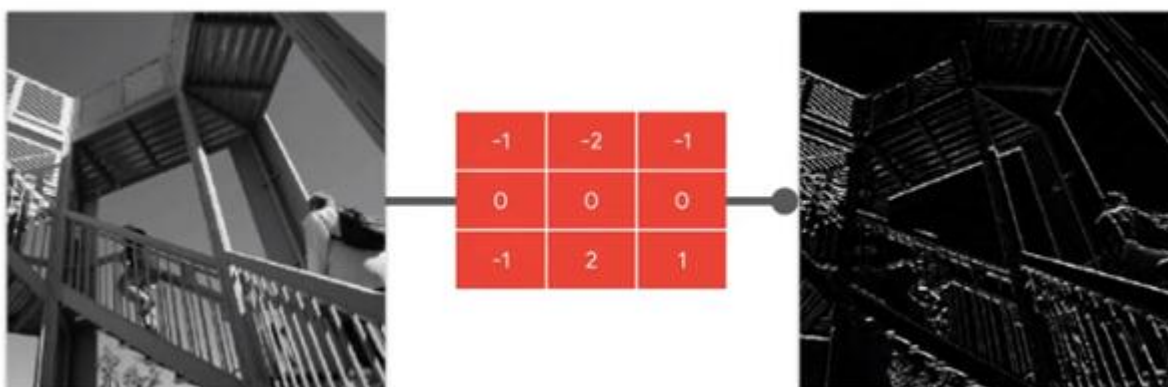


Figure II-3: Filtre qui supprime les lignes horizontales

Avec ce filtre par exemple, supprime presque tous sauf les lignes horizontales. Cette façon de faire peut-être combinée avec ce qu'on appelle la **couche de pooling** ou mise en commun, une forme de sous-échantillonnage de l'image qui permet le regroupement des pixels des filtres en un sous-ensemble. Ainsi par exemple la fonction de pooling maximum 2*2 regroupe des pixels en ensembles de 2*2 et comme résultat, elle sélectionne la valeur d'origine maximale. L'image sera réduite au quart de sa taille d'origine mais ses caractéristiques seront conservées.

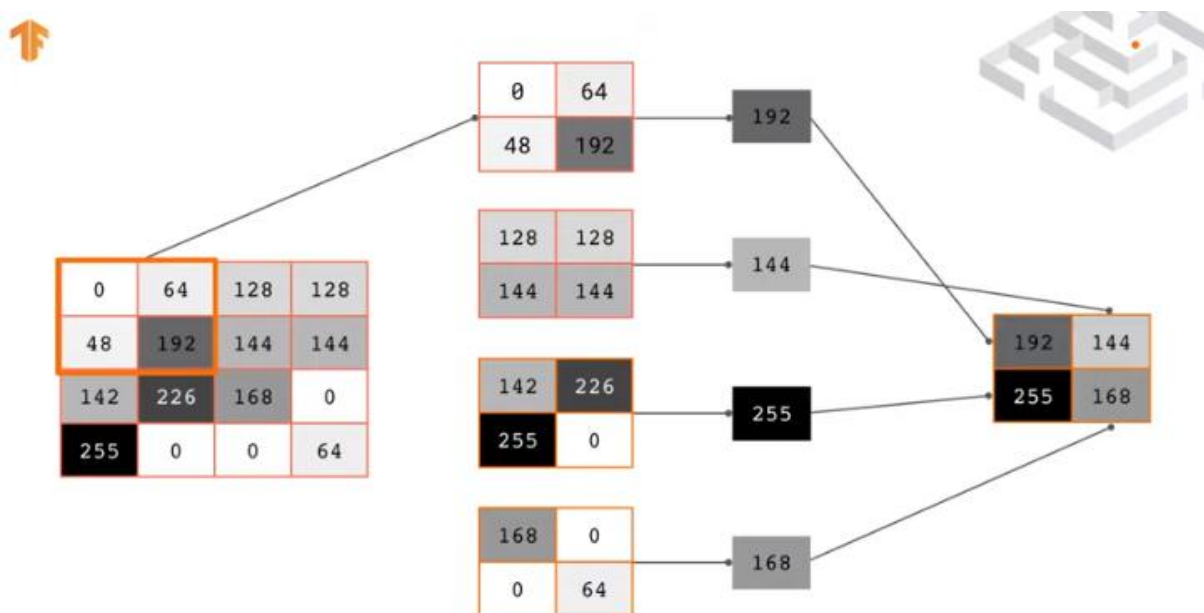


Figure II-4: Fonction Max pooling 2x2



Figure II-5: Résultat de l'application du Max pooling 2x2

L'image à droite est un quart de la taille de l'image de gauche, mais les caractéristiques des lignes verticales ont été conservées et améliorées.

Enfin nous avons les couches entièrement connectées ou Fully connected, qui est entièrement connectée à la sortie de la couche précédente [15].

Une question très importante nous traverse l'esprit, d'où viennent ces filtres ?

Les filtres sont appris par le réseaux de neurone, ce sont des paramètres communs des neurones, ainsi quand notre image est introduite dans la couche de convolution, un certain nombre de filtres initialisés de manière aléatoire sont appliqués à l'image, les résultats de ces derniers sont introduits dans la couche suivante et les correspondances sont effectuées par le réseau neuronal et c'est de fur et à mesure que les filtres qui produisent les images

avec les meilleures correspondances sont appris et le processus s'appelle extraction de caractéristiques [8].

II.5. Architecture globale d'un réseau convolutionnel

Les couches convolutionnelles peuvent être empilées pour former un réseau ; elles sont accompagnées de couches dites de *pooling* qui permettent de réduire la taille des cartes de caractéristiques, et de couches de *batch normalization* qui effectuent un recentrage et une normalisation des données. Dans le cadre d'une tâche de classification, le DNN se termine par une ou plusieurs couches complètement connectées.

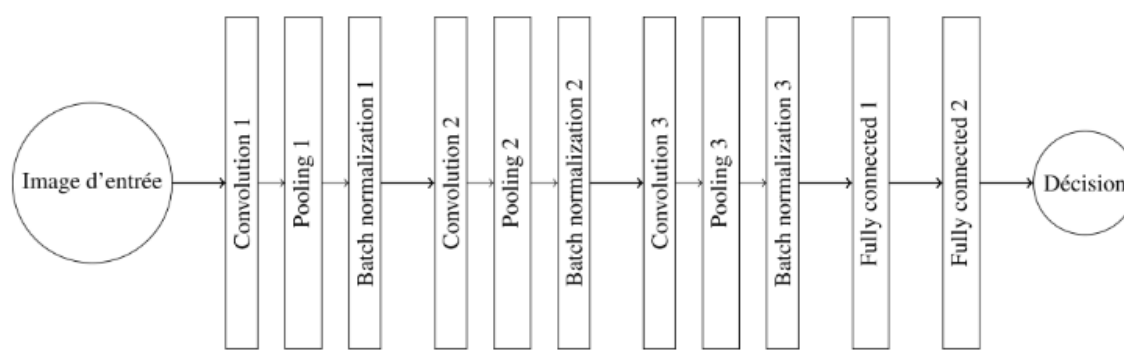


Figure II-6: Architecture d'un CNN

Ce réseau contient 3 séries de couches convolutionnelles, pooling et batch normalization, puis de deux couches fully connected. Nous avons déjà vu les couches fully connected dans les chapitres précédents. Nous allons donner quelques détails supplémentaires à la couche convolutionnelle introduite dans la section précédente, et décrire *pooling* et *batch normalization*.

II.5.1. Les couches convolutionnelles

Afin, d'une part, d'éviter la saturation des unités et d'autre part, de diminuer le phénomène d'évanescence du gradient, on peut utiliser comme fonction d'activation, une fonction qui ne présente pas de palier. Typiquement, c'est la fonction *relu* (.....) ou une approche dérivable comme la *softplus* qui seront utilisées pour les couches convolutionnelles.

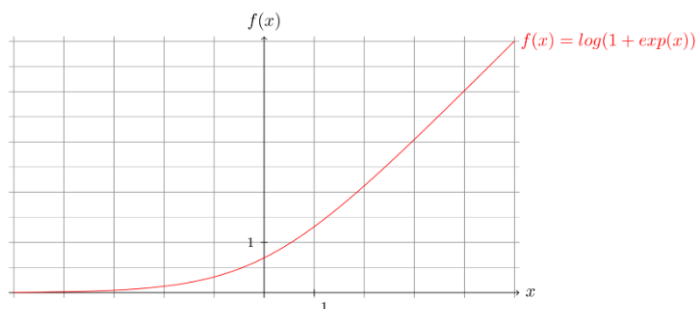


Figure II-7: Fonction softplus

Lorsque n filtres sont utilisés dans une même couche, on génère n cartes de sortie en parallèle

:

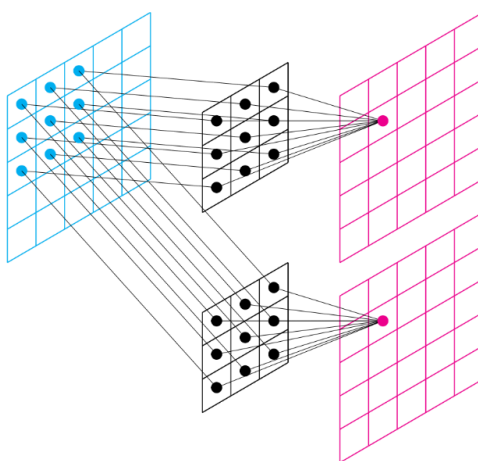


Figure II-8: Plusieurs filtres donnent plusieurs cartes en sortie

II.5.2. Couche de Pooling

La couche de pooling consiste à un rééchantillonnage des données. Celui-ci peut se faire suivant divers opérateurs, comme la moyenne et le max. Dans une tâche de classification, la localisation des caractéristiques étant peu importante, un pooling avec un opérateur max sera préféré.

II.5.3. Couche Batch Normalization

La couche de batch normalization est une autre technique de réduction du phénomène d'évanescence du gradient, elle sert à améliorer la vitesse, les performances et la stabilité des réseaux de neurones artificiels. Il est utilisé pour normaliser la couche d'entrée par recentrage et redimensionnement [16].

II.5.4. Couche Fully Connected

La couche fully-connected constitue toujours la dernière couche d'un réseau de neurones, convolutif ou non – elle n'est donc pas caractéristique d'un CNN.

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

La dernière couche fully-connected permet de classifier l'image en entrée du réseau : elle renvoie un vecteur de taille N , où N est le nombre de classes dans notre problème de classification d'images. Chaque élément du vecteur indique la probabilité pour l'image en entrée d'appartenir à une classe [17].

II.6. Conclusion

Dans ce chapitre nous avons mis en avant la classification d'images ainsi que l'utilisation de l'apprentissage automatique, plus précisément les réseaux convolutionnels de deep learning afin d'automatiser et d'optimiser ce processus.

Chapitre III : La carte de développement STM32 Discovery

III.1. Introduction

Ce chapitre présente la carte de développement utilisé dans notre projet. Cette carte comporte un microcontrôleur 32 bit à cœur ARM Cortex-M caractérisé par une grande vitesse d'exécution et une consommation optimisée. L'implémentation du réseau de neurones profond choisi sera effectuée en utilisant l'environnement de développement Keil MDK avec la bibliothèque CMSIS.

III.2. Les processeurs ARM

Un processeur ARM fait partie d'une famille de CPU basée sur l'architecture RISC développée par Advanced RISC Machines (ARM).

ARM fabrique des processeurs multicœurs RISC 32 bits et 64 bits. Les processeurs RISC sont conçus pour exécuter un plus petit nombre de types d'instructions informatiques afin de pouvoir fonctionner à une vitesse plus élevée, exécutant ainsi plus de millions d'instructions par seconde (MIPS). En supprimant les instructions inutiles et en optimisant les voies, les processeurs RISC fournissent d'excellentes performances à une fraction de la demande de puissance des appareils CISC.

Les processeurs ARM sont largement utilisés dans les appareils électroniques grand public tels que les smartphones, tablettes, lecteurs multimédias et autres appareils mobiles, tels que les appareils portables. En raison de leur jeu d'instructions réduit, ils nécessitent moins de transistors, ce qui permet une taille de puce plus petite pour les circuits intégrés. La taille plus petite du processeur ARM, sa complexité réduite et sa faible consommation d'énergie les rendent adaptés à des appareils de plus en plus miniaturisés [18].

Les processeurs ARM M4 et M7 supporte des instructions spéciales SIMD ce qui accélère les multiplications matricielles et avec le microcontrôleur ARM M4 intégrant des instructions DSP lui permettant de fonctionner plus vites avec ces instructions spéciales.

III.2.1. Les Fonctionnalités du Processeur ARM

➤ Architecture de chargement / stockage

C'est une architecture dans laquelle seules les instructions de chargement et de stockage accèdent à la mémoire, toutes les autres instructions utilisent des registres comme opérandes.

➤ **Un jeu d'instructions orthogonales**

Un jeu d'instructions orthogonales est une architecture de jeu d'instructions où tous les types d'instructions peuvent utiliser tous les modes d'adressage.

➤ **Exécution principalement en un seul cycle**

Instructions simples mais puissantes qui s'exécutent en un seul cycle à une vitesse d'horloge élevée.

➤ **Conception à économie d'énergie améliorée**

Les processeurs ARM dispose d'une meilleure efficacité énergétique comparée aux autres processeurs concurrents.

➤ **États d'exécution 64 et 32 bits pour des performances évolutives élevées**

Certain processeurs ARM offre la possibilité de changer l'état d'exécution de 64 bits à 32 bits.

➤ **Prise en charge de la virtualisation matérielle**

Un composant informatique créé dans le cadre de la virtualisation est appelé composant virtuel ou logique et peut être utilisé de la même manière que son équivalent physique.

La conception simplifiée des processeurs ARM permet un traitement multicœur plus efficace et un codage plus facile pour les développeurs. Bien qu'ils n'aient pas le même débit de calcul brut que les produits d'Intel, leader du marché x86, les processeurs ARM dépassent parfois les performances des processeurs Intel pour les applications qui existent sur les deux architectures [19].

III.2.2. Les différentes catégories de processeurs ARM

L'architecture d'ARM assure une meilleure sécurité, une large compatibilité, des performances élevées et une efficacité énergétique. Elle se décline en trois variétés optimisées pour différents cas d'utilisation: profil A pour des applications riches et diversifiées, profil R pour le temps réel et profil M pour les microcontrôleurs.

Toutes les conceptions de CPU basées sur Arm sont construites sur la même architecture, garantissant la compatibilité logicielle [19].

Architecture ARM: pour divers besoins de traitement embarqué

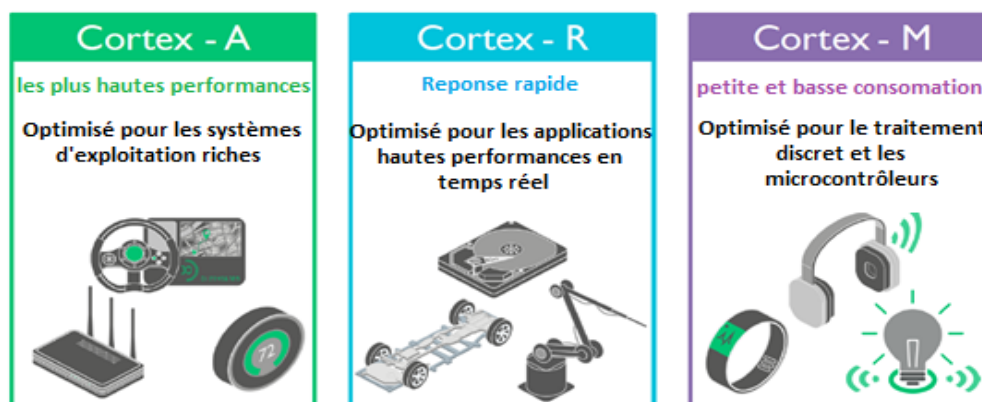


Figure III-1: Différents types de processeurs arm

III.2.2.1. Profil A

Optimisé pour les applications riches et les systèmes d'exploitation hautes performances, les processeurs de la famille A sont utilisés dans des applications de calcul complexes, notamment des serveurs, des téléphones mobiles, des unités centrales automobiles et des passerelles IoT.

III.2.2.2. Profil R

Le Profile R est utilisé lorsqu'une réponse en temps réel ou déterministe est requise, comme les applications critiques pour la sécurité. Les cas d'utilisation incluent l'équipement médical et le transport, en particulier dans les fonctions de direction, de freinage et de signalisation des véhicules.

III.2.2.3. Profil M

Le profil M est utilisé lorsque l'efficacité énergétique, la conservation de l'énergie et la taille sont essentielles. Il est particulièrement adapté aux puces invisibles et profondément intégrées, tels que les produits intelligents, ils sont utilisés dans les petits capteurs, les modules de communication et les produits pour la maison intelligente.

III.3. La carte stm32F411

Les cartes à microcontrôleurs STM32F411 font partie des gammes STM32 Dynamic Efficiency™. Ces appareils sont le niveau d'entrée de la série F4 hautes performances et offrent le meilleur équilibre entre la consommation d'énergie dynamique (en mode exécution) et les performances de traitement.

Les microcontrôleurs STM32F411 offrent les performances du noyau Cortex®-M4 avec unité à virgule flottante, fonctionnant à 100 MHz, tout en atteignant des valeurs de consommation d'énergie faibles en modes marche et arrêt.



Figure III-2: La carte stm32

Avec un nouveau mode d'acquisition par lots BAM, optimisant la consommation d'énergie pour le traitement par lots de données, la carte STM32F411 élève l'efficacité dynamique. Ce BAM permet d'échanger des lots de données via des périphériques de communication avec le reste de l'appareil (y compris le CPU) étant en mode d'économie d'énergie [20].

III.3.1. Performances

Avec une vitesse d'exécution à 100 MHz, le STM32F411 offre 125 performances DhrystoneMIPS / 339 CoreMark exécutées à partir de la mémoire Flash, avec des états d'attente nulle à l'aide de l'accélérateur ART de ST. Les instructions de traitement de signaux digitaux DSP et l'unité à virgule flottante élargissent la gamme d'applications adressables.

III.3.2. Efficacité énergétique

Le processeur à 90 nm de ST, l'accélérateur ART et la mise à l'échelle dynamique de la puissance permettent à la consommation de courant lors de l'exécution à partir de la mémoire Flash d'être aussi faible que 100 $\mu\text{A} / \text{MHz}$. En mode d'arrêt, la consommation d'énergie peut être aussi faible que 10 μA .

III.3.3. Intégration

- Microcontrôleur STM32F411VET6 avec 512 Ko de mémoire Flash, 128 Ko de RAM dans un boîtier LQFP100
- ST-LINK / V2 intégré avec commutateur de mode de sélection pour utiliser le kit en tant que ST-LINK / V2 autonome (avec connecteur SWD pour la programmation et le débogage)
- Alimentation de la carte: via bus USB ou à partir d'une tension d'alimentation externe de 5 V
- Alimentation externe de l'application: 3 V et 5 V
- L3GD20: Gyroscope de sortie numérique à 3 axes avec capteur de mouvement ST MEMS
- LSM303DLHC: système intégré ST MEMS comprenant un capteur d'accélération linéaire numérique 3D et un capteur magnétique numérique 3D
- MP45DT02: Capteur audio ST MEMS, microphone numérique omnidirectionnel

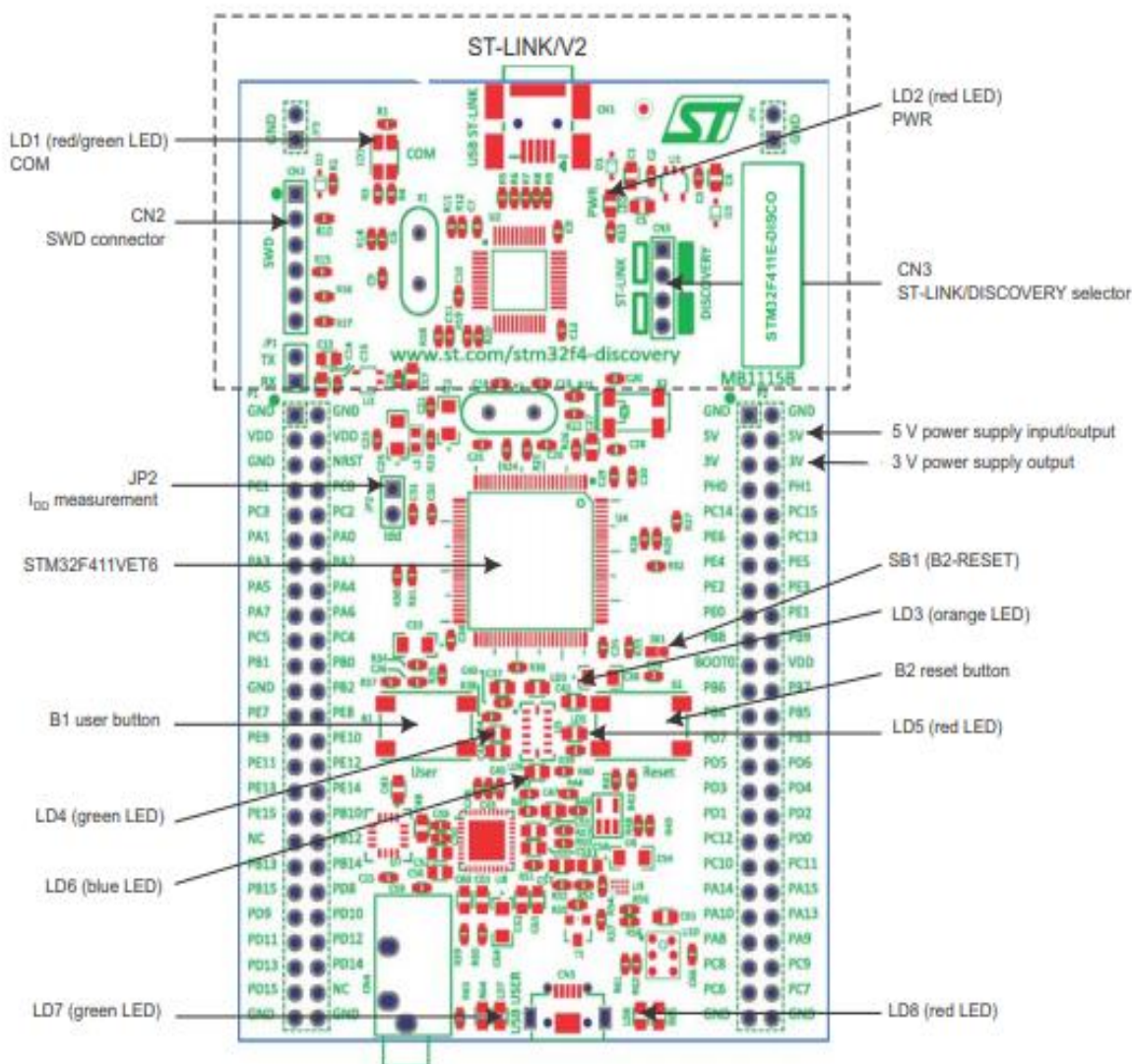


Figure III-3: Disposition supérieure

- Les packages disponibles vont de 49 à 100 broches.
- 3x USARTs fonctionnant jusqu'à 12,5 Mbit / s,
- 5x SPI (multiplexé avec I2S) fonctionnant jusqu'à 50 Mbit / s,
- 3x I²C jusqu'à 1Mbps
- 1x SDIO fonctionnant jusqu'à 48 MHz et disponible sur tous les packages,
- 1x USB 2.0 OTG pleine vitesse,
- 2x I²S full duplex jusqu'à 32 bits / 192KHz,
- 3x simplex I²S jusqu'à 32 bits / 192 kHz,
- ADC 12 bits atteignant 2,4 MSPS,
- 11 timers, 16 et 32 bits, fonctionnant jusqu'à 100 MHz

III.4. Outil de développement

Les STM32 supportent plusieurs environnements de développement dont les plus répandus sont Keil MDK-ARM, et les IDEs basés sur Eclipse et GCC . Keil MDK inclut l'IDE et le débogueur μ Vision, le compilateur Arm C / C ++ et les composants middleware essentiels. Le développement en utilisant ces outils est renforcé par un ensemble de bibliothèques spécialisées, tel que CMSIS, qui permettent une interface simple et cohérente avec n'importe quel processeur Cortex-M et ses périphériques.

III.4.1. Bibliothèques CMSIS

Le CMSIS est une couche d'abstraction matérielle indépendante du fournisseur pour les microcontrôleurs basés sur des processeurs Arm Cortex. Le CMSIS définit des interfaces d'outils génériques et permet une prise en charge cohérente des périphériques. Il fournit des interfaces logicielles simples au processeur et aux périphériques, simplifiant la réutilisation des logiciels, réduisant la courbe d'apprentissage pour les développeurs de microcontrôleurs et réduisant le temps de mise sur le marché des nouveaux appareils.

Le CMSIS est défini en étroite collaboration avec divers fournisseurs de logiciels et de hardware et fournit une approche commune d'interface avec les périphériques, les systèmes d'exploitation en temps réel et les composants middleware. Il est destiné à permettre la combinaison de composants logiciels de plusieurs fournisseurs de middleware [21].

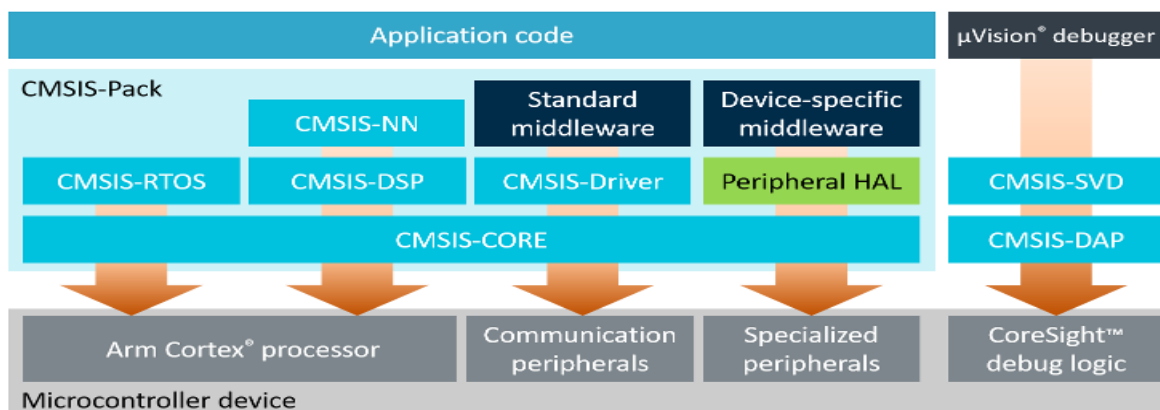


Figure III-4: Eléments du CMSIS

III.4.2. La bibliothèque CMSIS-NN

La bibliothèque CMSIS NN est une collection de noyaux de réseaux de neurones efficaces développés pour maximiser les performances et minimiser l'empreinte mémoire des réseaux de neurones sur les cœurs de processeur Cortex-M [22].

La bibliothèque est divisée en un certain nombre de fonctions couvrant chacune une catégorie spécifique :

- Fonctions de convolution
- Fonctions d'activation
- Fonctions de couche entièrement connectées
- Fonctions de mise en commun
- Fonctions Softmax
- Fonctions mathématiques de base

La bibliothèque possède des fonctions distinctes pour fonctionner sur différents types de données de poids et d'activation, y compris des entiers 8 bits (q7_t) et des entiers 16 bits (q15_t). La description des noyaux est incluse dans la description de la fonction.

III.4.3. Quantification en virgule fixe

La quantification en virgule fixe permet d'éviter le calcul coûteux en virgule flottante et réduit l'empreinte mémoire pour stocker à la fois les poids et les activations, ce qui est essentiel pour les plates-formes à ressources limitées.

- 1- La quantification des pondérations en virgule flottante 32 bits en pondérations en virgule fixe 8 bits pour le déploiement réduit la taille du modèle de 4X.
- 2- Les opérations sur les nombres entiers à virgule fixe sont beaucoup plus rapides que les opérations en virgule flottante dans les microcontrôleurs classiques.

III.4.4. Classification des fonctions

Les fonctions peuvent être classées en deux segments :

- Les NNFunctions incluent les fonctions qui implémentent les types de couches de réseaux neuronaux, tels que la convolution, couche entièrement connecté (c'est-à-dire le produit interne), la mise en commun et l'activation. Ces fonctions sont utilisées par le code d'application pour implémenter les applications d'inférence de réseau neuronal.
- NNSupportFunctions comprend différentes fonctions utilitaires, telles que la conversion de données et les tables de fonctions d'activation, qui sont utilisées dans NNFunctions [22].

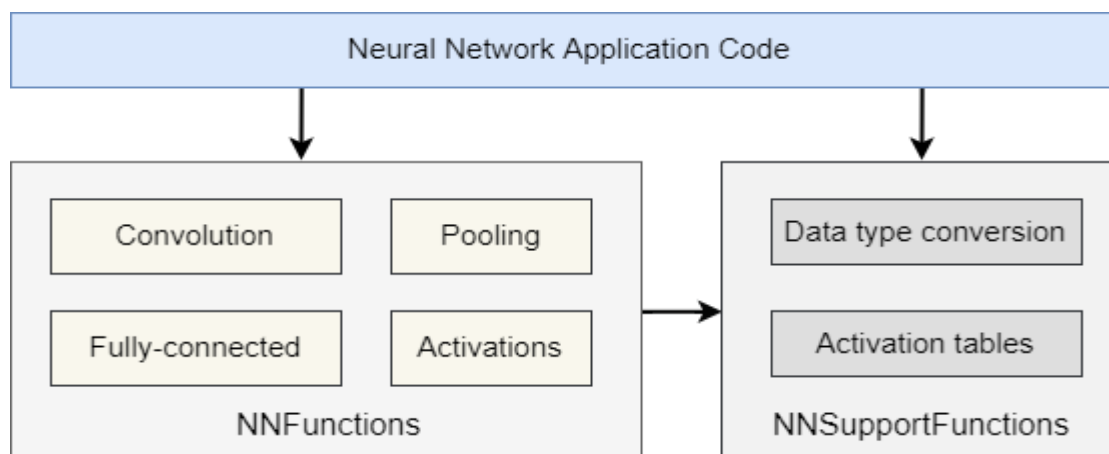


Figure III-5: Diagramme fonctions CMSIS-NN

III.4.5. CIFAR 10

L'ensemble de données CIFAR-10 se compose de 60000 images couleur 32x32 en 10 classes, avec 6000 images par classe. Il existe 50000 images d'entraînement et 10000 images de test.

L'ensemble de données est divisé en cinq lots de formation et un lot de test, chacun contenant 10000 images. Le lot de test contient exactement 1000 images sélectionnées au hasard dans chaque classe. Les lots de formation contiennent les images restantes dans un ordre aléatoire, mais certains lots de formation peuvent contenir plus d'images d'une classe

que d'une autre. Entre eux, les lots de formation contiennent exactement 5000 images de chaque classe [23]. Les étiquettes de classe et leurs valeurs entières standard associées sont données par la figure III-6 avec quelques exemples.

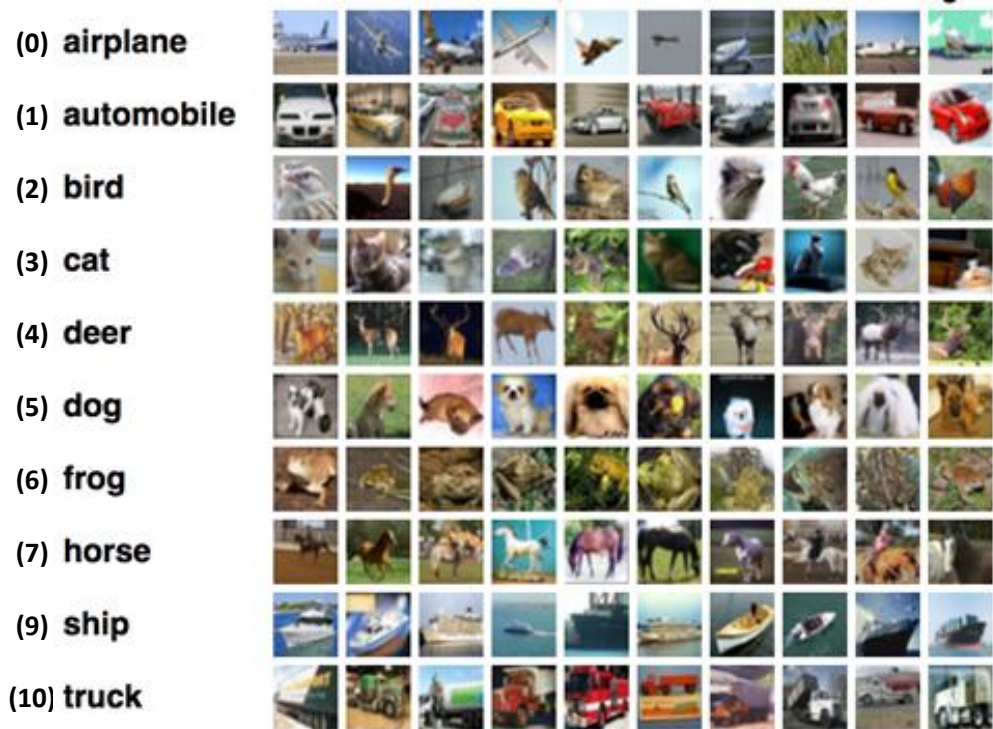


Figure III-6: Exemples d'images CIFAR-10 appartenant de chaque catégorie

III.5. Conclusion

Dans ce chapitre on a présenter les processeurs ARM, la carte stm32f411 qui inclut un processeur ARM cortex M4, ces performances, efficacité énergétique et les différents composants intégrés dans la carte.

On a aussi présenté Arm CMSIS-NN qui permet de maximiser les performances et minimiser l'empreinte mémoire des réseaux neuronaux sur les cœurs de processeur Arm® Cortex®-M.

Chapitre IV : Implémentation d'un CNN sur la carte STM32

IV.1. Introduction

Dans ce chapitre nous nous consacrons à l'implémentation du CNN dans notre microcontrôleur, dans un premier temps nous allons présenter le programme et ses différents composants, étapes et son fonctionnement, puis nous finirons avec le résultat du traitement de certaines images et les éléments clés qui permettent l'augmentation de la vitesse d'exécution de 4 fois.

IV.2. L'architecture du CNN à implémenter

Le CNN à implémenter est constitué de 7 couches dont 3 convolutives, 3 couches de mise en commun (pooling), une couche entièrement connectée et une couche de classification softmax, comme le montre le schéma illustré ci-dessous.

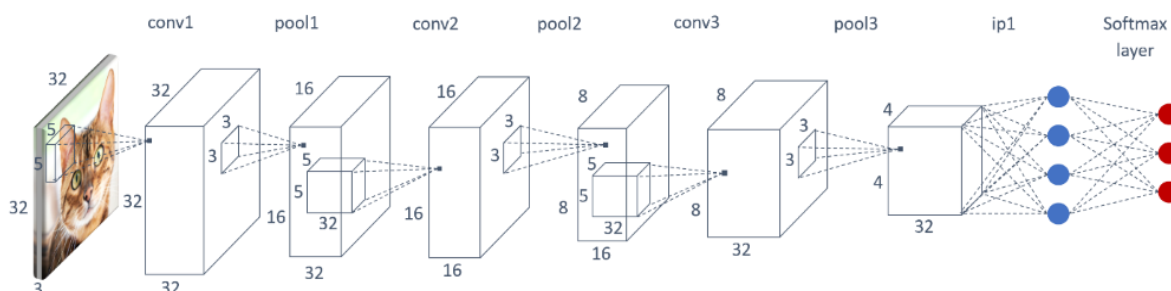


Figure IV-1: l'architecture du réseau de neurones à implémenter

Le programme reçoit une image et applique une couche convolutive suivie d'une couche de mise en commun pour extraire ses éléments clés et réduire sa taille avant d'être traitée par la couche entièrement connectée qui renvoie la décision de classification finale.

Notre réseau de neurones est entraîné avec le lot de données cifar 10, ce qui lui permet de classer des images appartenant à une des 10 classes contenues dans le lot.

IV.2.1. Différentes fonctions du programme

Le programme inclut plusieurs fonctions convolutives

1- `arm_convolve_HWC_q7_basic`:

La fonction `arm_convolve_HWC_q7_basic` est la version la plus basique conçue pour fonctionner pour n'importe quel tenseur d'entrée et dimension de poids de forme carrée.

2- `arm_convolve_HWC_q7_fast`:

La fonction `arm_convolve_HWC_q7_fast`, comme son nom l'indique, fonctionne plus vite que la précédente mais exige que les canaux de tenseur d'entrée soient multiples de 4 et que les canaux de tenseur de sortie (nombre de filtres) soient multiples de 2.

3- `arm_convolve_HWC_q7_RGB`:

`arm_convolve_HWC_q7_RGB` est construit exclusivement pour la convolution avec des canaux de tenseur d'entrée égal à 3 avec quatre groupes de paramètres qui sont: entrée, noyau de filtre, biais et sortie, cette fonction est généralement appliquée à la toute première couche convolutive prenant des données d'image RGB en entrée.

4- `arm_convolve_HWC_q7_fast_nonsquare`:

`arm_convolve_HWC_q7_fast_nonsquare` similaire à `arm_convolve_HWC_q7_fast`, mais peut prendre un tenseur d'entrée de forme non carrée.

En ce qui concerne les couches entièrement connectées, les deux options les plus distinctes sont:

```
arm_fully_connected_q7
arm_fully_connected_q7_opt
```

Le premier fonctionne avec une matrice de poids régulière et l'autre avec le suffixe "_opt" est optimisé pour la vitesse mais la matrice de poids de la couche doit être ordonnée au préalable de manière entrelacée. La réorganisation peut être réalisée de manière transparente à l'aide d'un script générateur de code.

IV.2.2. Implémentation des couches du CNN

Le CNN est constitué de 7 couches :

1. La première couche inclut `arm_convolve_HWC_q7_RGB`

```
// conv1 img_buffer2 -> img_buffer1
arm_convolve_HWC_q7_RGB(img_buffer2, CONV1_IM_DIM, CONV1_IM_CH, conv1_wt, CONV1_OUT_CH, CONV1_KERNEL_DIM, CONV1_PADDING,
                        CONV1_STRIDE, conv1_bias, CONV1_BIAS_LSHIFT, CONV1_OUT_RSHIFT, img_buffer1, CONV1_OUT_DIM,
                        (q15_t *) col_buffer, NULL);
```

Comme nous l'avons expliqué cette fonction est appliquée à la première couche convolutive dans le traitement des images.

2. La deuxième couche inclut deux API (Application Programming Interface), `arm_maxpool_q7_HWC` () et `arm_relu_q7` () .

```
// pool1 img_buffer1 -> img_buffer2
arm_maxpool_q7_HWC(img_buffer1, CONV1_OUT_DIM, CONV1_OUT_CH, POOL1_KERNEL_DIM,
                  POOL1_PADDING, POOL1_STRIDE, POOL1_OUT_DIM, NULL, img_buffer2);
```

Le résultat est découpé en une série de rectangles de n pixels de côté ne se chevauchant pas. Et prend la plus grande valeur en sortie.

```
arm_relu_q7(img_buffer1, CONV2_OUT_DIM * CONV2_OUT_DIM * CONV2_OUT_CH);
```

La fonction relu prend la valeur de chaque pixel si il est supérieur à 0 il renvoie la même valeur sinon il renvoie 0.

3. La troisième couche inclut `arm_convolve_HWC_q7_fast ()`

```
// conv2 img_buffer2 -> img_buffer1
arm_convolve_HWC_q7_fast(img_buffer2, CONV2_IM_DIM, CONV2_IM_CH, conv2_wt, CONV2_OUT_CH, CONV2_KERNEL_DIM,
                        CONV2_PADDING, CONV2_STRIDE, conv2_bias, CONV2_BIAS_LSHIFT, CONV2_OUT_RSHIFT, img_buffer1,
                        CONV2_OUT_DIM, (q15_t *) col_buffer, NULL);
```

4. La quatrième couche inclut `arm_maxpool_q7_HWC ()` pour une taille de bufferA non nulle.

5. La cinquième couche inclut à nouveau l'API convolve `arm_convolve_HWC_q7_fast()`.

```
// conv3 img_buffer2 -> img_buffer1
arm_convolve_HWC_q7_fast(img_buffer2, CONV3_IM_DIM, CONV3_IM_CH, conv3_wt, CONV3_OUT_CH, CONV3_KERNEL_DIM,
                        CONV3_PADDING, CONV3_STRIDE, conv3_bias, CONV3_BIAS_LSHIFT, CONV3_OUT_RSHIFT, img_buffer1,
                        CONV3_OUT_DIM, (q15_t *) col_buffer, NULL);
```

6. La sixième inclut à nouveau `arm_maxpool_q7_HWC()`.

7. La septième inclut l'API `arm_fully_connected_q7_opt ()`, cette fonction optimisée est conçue pour fonctionner avec une matrice de poids entrelacée.

```
arm_fully_connected_q7_opt(img_buffer2, ip1_wt, IP1_DIM, IP1_OUT, IP1_BIAS_LSHIFT, IP1_OUT_RSHIFT, ip1_bias,
                          output_data, (q15_t *) img_buffer1);
```

Le résultat de cette dernière alimentera la couche softmax qui produira une probabilité pour classer la sortie dans l'une des dix classes.

```
arm_softmax_q7(output_data, 10, output_data);
```

IV.2.3. Description

Dans le fichier source `arm_nnexamples_cifar10.cpp`, pour insérer les données dans la RAM de 128 Ko pendant le calcul, deux variables de tampon sont créées et réutilisées entre les couches.

```
q7_t      col_buffer[2 * 5 * 5 * 32 * 2];
```

```
q7_t      scratch_buffer[32 * 32 * 10 * 4];
```

`col_buffer` stocke la sortie `im2col` (image vers colonne) pour les couches convolutives,

`scratch_buffer` stocke les données d'activation (sorties de couche intermédiaire)

```
q7_t      *img_buffer1 = scratch_buffer;
q7_t      *img_buffer2 = img_buffer1 + 32 * 32 * 32;
```

La fonction `arm_convolve_HWC_q7_fast` crée une couche convolutive qui prend le contenu de `img_buffer2` comme données d'entrée, et les sorties vers `img_buffer1`.

```
// conv2 img_buffer2 -> img_buffer1
arm_convolve_HWC_q7_fast(img_buffer2, CONV2_IM_DIM, CONV2_IM_CH, conv2_wt, CONV2_OUT_CH, CONV2_KERNEL_DIM,
    CONV2_PADDING, CONV2_STRIDE, conv2_bias, CONV2_BIAS_LSHIFT, CONV2_OUT_RSHIFT, img_buffer1,
    CONV2_OUT_DIM, (q15_t *) col_buffer, NULL);
```

Elle utilise également `col_buffer` comme mémoire interne pour exécuter l'algorithme d'image convolutionnelle en colonne.

La couche d'activation ReLu suivante agit sur le `img_buffer1` lui-même, puis le même buffer sera l'entrée pour la couche de mise en commun (maxpool) suivante où cette couche sort vers `img_buffer2` et ainsi de suite.

```
arm_relu_q7(img_buffer1, CONV1_OUT_DIM * CONV1_OUT_DIM * CONV1_OUT_CH);

// pool1 img_buffer1 -> img_buffer2
arm_maxpool_q7_HWC(img_buffer1, CONV1_OUT_DIM, CONV1_OUT_CH, POOL1_KERNEL_DIM,
    POOL1_PADDING, POOL1_STRIDE, POOL1_OUT_DIM, NULL, img_buffer2);
```

Depuis l'espace RAM contraint, nous ne pouvons pas attribuer généreusement une grande partie de la mémoire à ces deux tampons.

Au lieu de cela, nous allouons juste assez d'espace mémoire pour le modèle à utiliser.

IV.2.4. Technique de programmation

La fonction `im2col` convertit la convolution en multiplication matricielle on arrangeant les données de manière à ce que la sortie de convolution puisse être obtenue par

multiplication matricielle. Elle améliore le parallélisme dans la convolution en utilisant les fonctionnalités SIMD du processeur mais introduit une surcharge mémoire.

Nous utilisons la fonction `im2col` qui transformera l'image d'entrée ou le lot en une matrice, puis nous multiplions cette matrice par une version remodelée de notre noyau. Ensuite, à la fin, nous remodelons cette matrice multipliée en une image avec l'opération `col2im`.

IV.2.5. Execution du programme

Les étapes d'initialisation de l'IDE pour fonctionner avec la carte `stm32f411` est détailler en annexe. Après la compilation et l'entrée en mode debugger nous exécutons le programme et on obtient le résultat suivant :

Les dix classes suivant le meme ordre du cifar 10 avec chacune la probabilité d'appartenance de l'image à cette dernière.

Nous allons procédé au traitement de deux images et voir le resultat obtenu.



Figure IV-2: Image A



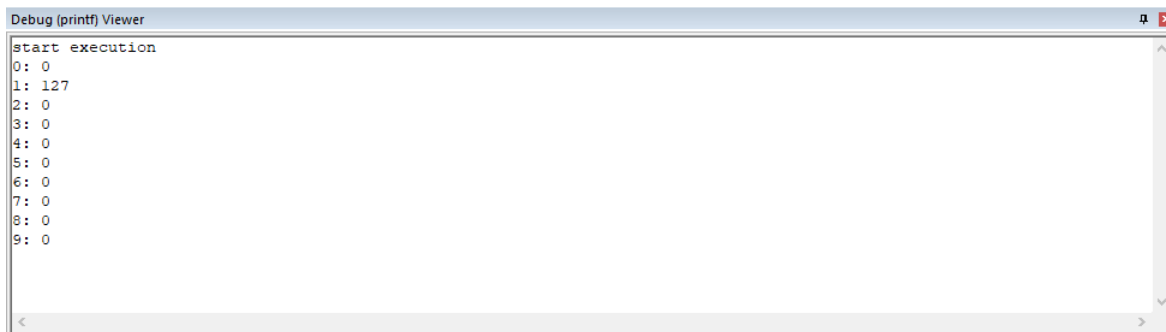
Figure IV-3: Image B

```

Debug (printf) Viewer
start execution
0: 0
1: 0
2: 0
3: 102
4: 0
5: 0
6: 25
7: 0
8: 0
9: 0
    
```

Figure IV-4: résultats du premier test

Dans cet exemple ci-dessus le programme a procédé au traitement de l'image A et on a obtenu un resultat correct qui est la classe 3 (Chat).



```
Debug (printf) Viewer
start execution
0: 0
1: 127
2: 0
3: 0
4: 0
5: 0
6: 0
7: 0
8: 0
9: 0
```

Figure IV-5: Résultats du deuxième test

Dans cet exemple ci-dessus le programme a procédé au traitement de l'image B et on a obtenu un resultat correct qui est la classe 1 (Voiture).

IV.3. Conclusion

Dans ce chapitre nous avons fait la présentation du programme ces différents composants et étapes, et apres execution on a obtenu le resultat via le debug viewer de l'IDE keil mdk qui affiche le resultat de la classification de l'image.

CONCLUSION GENERALE

L'Intelligence Artificielle est un domaine qui est très prisé de nos jours. Du machine learning au deep learning, toutes ces technologies se sont révélées très utiles dans plusieurs domaines. De la médecine à l'informatique en passant par l'ingénierie, tous bénéficient de cette avancée technologique.

L'utilisation de microcontrôleurs 32 bit à cœur ARM Cortex-M ne cesse de croître dans la réalisation d'applications électroniques embarquées. Lorsque des critères tels que la vitesse d'exécution, le besoin de connectivité et d'IHM, ou encore une consommation optimisée sont déterminants, l'utilisation de composants comme le STM32 semble particulièrement adaptée.

L'existence de ce type de microcontrôleur a rendu possible l'intégration de réseaux de neurones profonds à des systèmes à base de microcontrôleurs, ce qui aide à faire une implémentation matérielle de ces derniers. De ce fait notre projet s'est porté sur cela, on a pu réaliser un programme de reconnaissance d'images grâce à la carte STM, on l'a programmée en utilisant le langage C, et on a abouti à un résultat convainquant. Grâce à la bibliothèque CIFAR 10 on a pu entraîner notre modèle à détecter certains objets et animaux, et on a testé grâce à des images arbitrairement choisies, et le résultat était correct à chaque essai.

Ce projet peut être porté vers à peu près n'importe quel domaine, c'est-à-dire grâce à la flexibilité du microcontrôleur, on peut inclure de nombreux capteurs, par exemple on peut placer une caméra, pour ainsi avoir une détection en temps réel des objets, et peut servir dans par exemple le domaine de l'automobile autonome.

Les références bibliographiques :

- [1] La rédaction de Futura, Qui sont les pionniers de l'intelligence artificielle ? <https://www.futura-sciences.com/tech/questions-reponses/intelligence-artificielle-sont-pionniers-intelligence-artificielle-4907/> (25/07/2018)
- [2] Kirill Eremenko, Hadelin de Ponteves, Deep learning A-Z : Hands on artificial neuronal network, <https://www.udemy.com/course/deeplearning/>, (05/2020)
- [3] Andrew Ng, What is machine learning, <https://www.coursera.org/lecture/machine-learning/what-is-machine-learning-Ujm7v> (05/2020)
- [4] Thèse de Doctorat par Thomas Laloë. Sur quelques problèmes d'apprentissage supervisé et non supervisé. Mathématiques Université Montpellier II - Sciences et Techniques du Languedoc, 2009.
- [5] Romain Herault, Clement Chatelain, Initiez vous au deep learning, <https://openclassrooms.com/fr/courses/5801891-initiez-vous-au-deep-learning> (05/2020)
- [6] Wikistat, Réseaux de neurones, Université de Toulouse <https://www.math.univ-toulouse.fr/~besse/Wikistat/pdf/st-m-app-rn.pdf>
- [7] Andrew Trask, A Neural Network in 13 lines of Python (Part 2 - Gradient Descent) <https://iamtrask.github.io/2015/07/27/python-network-part2/>
- [8] Laurence Monorey, introducing convolutional neural networks <https://codelabs.developers.google.com/codelabs/tensorflow-lab3-convolutions/#0>
- [9] Article scientifique de Razvan Pascanu, Tomas Mikolov et Yoshua Bengio, "On the difficulty of training recurrent neural networks" V2 2013
- [10] : Article Scientifique de Sepp Hochrieter, Jurgen Schmidhube, Long short term memory publié en 1997
- [11] : Article scientifique de Alex Sherstinsky Fundamentals of Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM) Network 2018
- [12] : Article scientifique de Pierre Baldi, Autoencoders, Unsupervised Learning, and Deep Architectures 2012
- [13] Thomas Mensink. Learning Image Classification and Retrieval Models. Information Retrieval Université de Grenoble, 2012. English
- [14] Matthieu Guillaumin, Jakob Verbeek and Cordelia Schmid, 'Multimodal semi-supervised learning for image classification' LEAR, INRIA Grenoble, Laboratoire Jean Kuntzmann Jun 2010, San Francisco, United States.
- [15] Renders, Jean-Michel, « Algorithmes génétiques et réseaux de neurone », Hermes Paris 1994.

[16] Firdaouss Doukkali , "Batch normalization in Neural Networks".
towardsdatascience.com. 20/10/2017

[17] Pascal Monasse, Kimia Nedjahi, « Classez et segmentez des données visuelles », <https://openclassrooms.com/fr/courses/4470531-classez-et-segmentez-des-donnees-visuelles> (2020)

[18] Margaret Rouse, "ARM processor" <https://whatis.techtarget.com/definition/ARM-processor#:~:text=An%20ARM%20processor%20is%20one,bit%20RISC%20multi%2Dcore%20processors> Janvier 2015

[19] ARM support, why arm cpu architecture, <https://www.arm.com/why-arm/architecture/cpu>, 2020

[20]STMicroelectronics support,
https://www.st.com/content/st_com/en/products/microcontrollers-microprocessors/stm32-32-bit-arm-cortex-mcus/stm32-high-performance-mcus/stm32f4-series/stm32f411.html#overview

[21] ARM Ltd, <http://www.keil.com/pack/doc/CMSIS/General/html/index.html> 9 Avril 2020

[22] ARM Ltd, <https://www.keil.com/pack/doc/CMSIS/NN/html/index.html> 9 Avril 2020

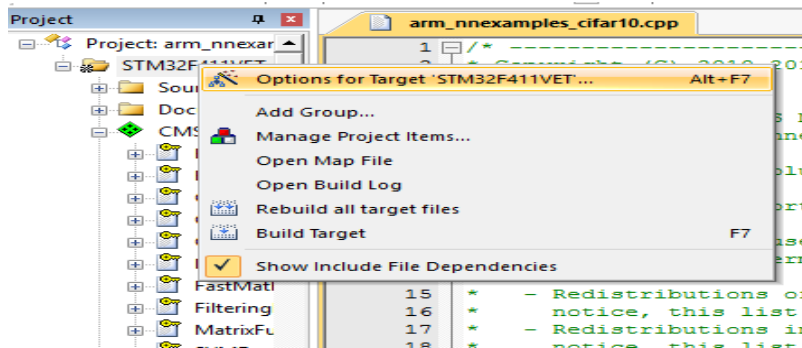
[23]Jason Brownlee, “Deep Learning for Computer Vision”
<https://machinelearningmastery.com/how-to-develop-a-cnn-from-scratch-for-cifar-10-photo-classification/> 28 aout 2020

Annexe :

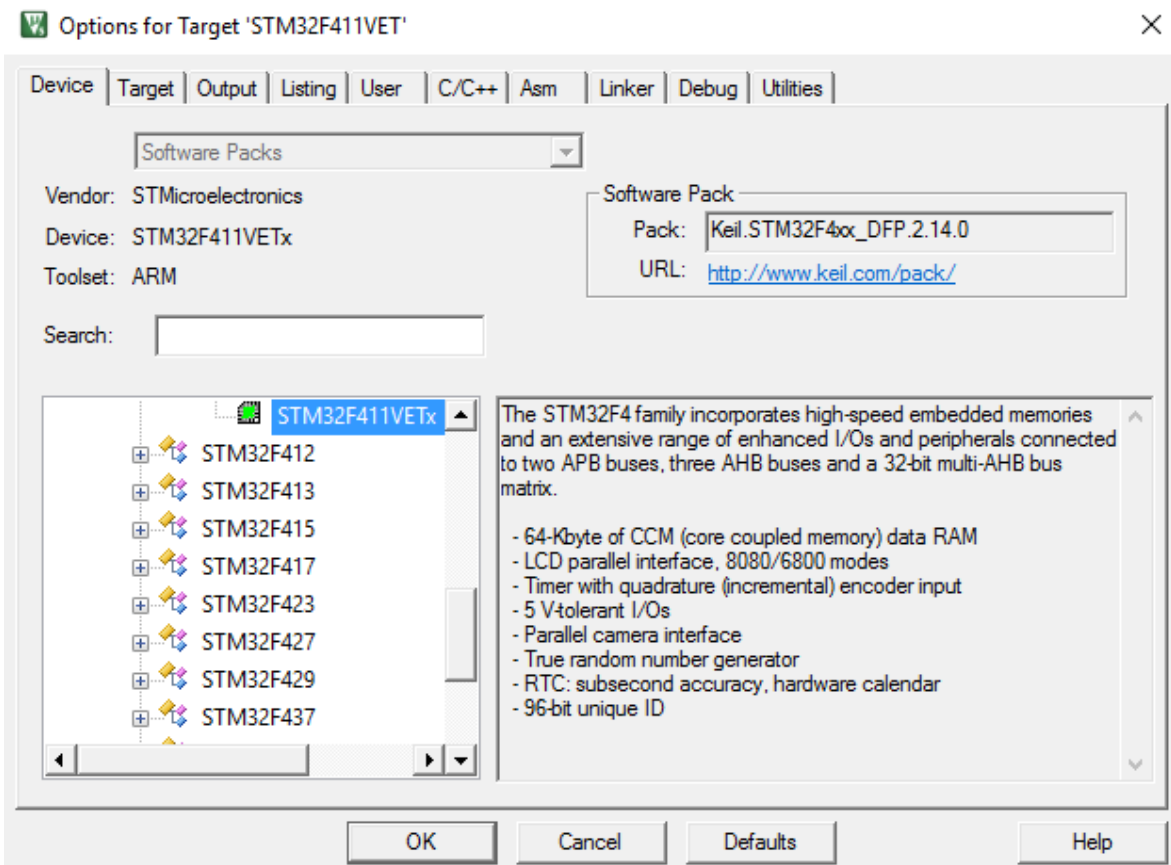
Étapes d'initialisation du programme :

A. Choisir le type de carte

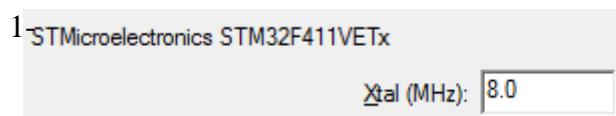
1-



2-



B. Reglages de l'horloge et memoires



2- réglage mémoire

Read/Only Memory Areas					Read/Write Memory Areas				
default	off-chip	Start	Size	Startup	default	off-chip	Start	Size	NoInit
<input type="checkbox"/>	ROM1:			<input type="radio"/>	<input type="checkbox"/>	RAM1:			<input type="checkbox"/>
<input type="checkbox"/>	ROM2:			<input type="radio"/>	<input type="checkbox"/>	RAM2:			<input type="checkbox"/>
<input type="checkbox"/>	ROM3:			<input type="radio"/>	<input type="checkbox"/>	RAM3:			<input type="checkbox"/>
on-chip					on-chip				
<input checked="" type="checkbox"/>	IROM1:	0x8000000	0x100000	<input checked="" type="radio"/>	<input checked="" type="checkbox"/>	IRAM1:	0x20000000	0x20000	<input type="checkbox"/>
<input type="checkbox"/>	IROM2:			<input type="radio"/>	<input type="checkbox"/>	IRAM2:	0x10000000	0x10000	<input checked="" type="checkbox"/>

C. Choix de connexion avec la carte

Linker | Debug | Utilities

Use: ST-Link Debugger Settings

D. Dans les paramètres :

1- Réglage de la fréquence du moniteur de série

Cortex-M Target Driver Setup

Debug | Trace | Flash Download

Core Clock: MHz Trace Enable

2- Choisir le type de mémoire flash

Cortex-M Target Driver Setup

Debug | Trace | Flash Download

Download Function

Erase Full Chip Program

Erase Sectors Verify

Do not Erase Reset and Run

RAM for Algorithm

Start: Size:

Programming Algorithm

Description	Device Size	Device Type	Address Range
STM32F4xx 512kB Flash	512k	On-chip Flash	08000000H - 0807FFFFH

Start: Size:

Add Remove

Résumé

L'intelligence artificielle nous permet de créer des machines ressemblant de plus en plus à l'humain dans le fait de pouvoir traiter des tâches qui étaient auparavant réservées uniquement à lui, grâce au développement des matériaux informatiques et les techniques de programmation qui allègent les ressources demandées, les perspectives offertes par cette technologie dans les différents domaines sont très prometteuses.

Dans l'optique d'intégrer cette technologie dans de petites puces électroniques pour gadgets intelligents, nous avons présenté dans notre projet l'implémentation d'un réseau de neurones qui était réservé aux machines puissantes dans un microcontrôleur de faible puissance fonctionnant avec des mémoires allant dans l'ordre de kilo octets, et nous avons choisi un microcontrôleur stm32f411 équipé d'un processeur ARM cortex M4 pour ces nombreuses technologies intégrées comme le SIMD, la FPU et DSP. Quant à l'outil de développement nous avons utilisé l'IDE keil mdk basé sur l'IDE éclipse, le débogueur μ Vision, le compilateur Arm C / C ++ et les composants middlewares essentiels.

Abstract

The artificial intelligence allows us to create machines that look like more and further more like humans in the fact of of the capability of treating tasks that was reserved only for humans, thanks to the development of the computers hardware and programming techniques to lighten the required ressources. the prospects offered by this technology in the various fields are very promising.

In order to integrate this technology in small electronic chips for smart gadgets, we have presented in our project how to implement a neuronal network that was reserved for powerfull machines in a low power microcontroller functioning with memories ranging in the order of kilobytes, and we have chosen an stm32f411 microcontroller equiped with an ARM M4 microprocessor for it integrated technologies like the SIMD, FPU and DSP. As for the development tool we have used the IDE keil mdk based on the IDE eclipse, the μ Vision debugger, the compiler ARM C \ C++ and the essentials middlewares components.