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 Génie Électrique

Mémoire de fin d'études de Master

Filière : Électronique
Spécialité : Électronique des systèmes embarqués

Implémentation d'une méthode d'extraction des caractéristiques des signaux vibratoires sur une plateforme FPGA

Présenté par :

Mr. ELMOHRI Mohamed Abdessalem

Mr. TOBBAL Abdelhafid

Proposé et Dirigé par :

Mr. CHADLI Samir

Dr. BENGHERBIA Billel

Promotion 2021

Remerciement

Avant tout, louange et gratitude à Dieu le tout puissant pour la volonté, la santé, la force et la patience qu'il nous a donné dans l'accomplissement de ce travail et l'atteinte de cet objectif.

Nous remercions du fond du cœur nos chers parents pour leurs soutient et leur aide.

Nous adressons nos sincères remerciements à nos promoteurs *Mr. CHADLI Samir* et *Mr. BENGHERBIA Billel* pour la qualité de leur encadrement, leur patience et leurs compétences.

Nos vifs remerciements aux membres du jury *Mr. MAAZOUZ Mohamed* et *Mr. REBIAI Mohamed* pour l'honneur qu'ils nous font en acceptant d'évaluer ce travail.

ملخص :

نقدم في هذا العمل طريقة لاستخراج خصائص إشارات الاهتزاز الناتجة عن الآلات الدوارة، وبشكل أكثر تحديدًا تلك الصادرة عن محامل الكريات. تتضمن الطريقة المقترحة المعالجة MFCC ومن ثم التنبؤ بموقع الخلل على المحمل باستخدام شبكة عصبية تلافيفية. تم توزيع المعالجة على مجموعة من IP-core باستخدام برنامج Vivado HLS، تـم تنفيذ المعالجة المقترحة على منصةFPGA Pynq-Z2 . و تم تنفيذ شبكة CNN على مفتاح تسريع مادي NCS2 من .Intel لقد أثبت النظام المقترح فعاليته وقدم نتائج تصنيف مرضية للغاية.

الكلمات المفتاحية : إشارات إهتزازية، الصيانة الوقائية، محامل الكريات ، MFCC، CNN، HLS، FPGA، PYNQ، Intel NCS2.

Résumé :

Dans ce travail nous exposons une méthode d'extraction des caractéristiques des signaux vibratoires issues des machines tournantes pour les défauts de roulements à billes. La méthode proposée comporte un prétraitement MFCC des signaux puis une prédiction de l'emplacement du défaut sur le roulement à l'aide d'un réseau de neurones convolutifs (CNN) Le prétraitement a été réparti sur un ensemble d'IP-core personnalisés réalisés grâce au logiciel Vivado HLS. Le système de traitement proposé a été implémenté sur la plateforme FPGA Pynq-Z2 de la firme TUL. Le réseau CNN quant à lui a été implémenté sur la clé d'accélération matérielle NCS2 d'Intel. Le système proposé a prouvé son efficacité et a donné des résultats de classification très satisfaisants.

Mots clé : Signaux vibratoires, Maintenance préventive, Roulements à billes, MFCC, CNN, HLS, FPGA, PYNQ, Intel NCS2.

Abstract :

In this work we expose a method of classifying vibratory signals from rotating machines, and more specifically those from ball bearings. The proposed method comprises an MFCC preprocessing of the signals then a prediction of the location of the defect on the bearing using a convolutional neural network. The preprocessing was distributed over a set of IP-core produced using Vivado HLS software, then implemented on the Pynq-Z2 FPGA platform. The CNN network was implemented on an Intel NCS2 hardware accelerator. The proposed system has proven its effectiveness and has given excellent classification results.

Key words : Vibration signals, Preventive maintenance, Ball bearings, MFCC, CNN, HLS, FPGA, PYNQ, Intel NCS2.

Table des matières

Liste des figures I

Liste des tableaux IV

Introduction générale 1

Chapitre I: Diagnostic des machines tournantes par analyse vibratoire 2

I.1 Introduction 3

I.2 Stratégie de maintenance 3

I.2.1 Définition de la maintenance 3

I.2.2 Objectif de la maintenance 3

I.2.3 Rôle de la maintenance dans une entreprise 4

I.2.4 Les types de maintenance 4

I.3 Implication de l'IA et l'IOT dans la maintenance 6

I.4 Analyse vibratoire des machines tournantes 7

I.4.1 Définition d'une vibration 8

I.4.2 Caractéristique d'une vibration 9

I.4.3 Grandeurs associées 9

I.4.4 Analyse temporelle et analyse fréquentielle 10

I.4.4.1 Analyse temporelle 10

I.5 Les défauts des roulements 12

I.5.1 Structure mécanique d'un roulement 13

I.5.2 Causes des défauts des roulements à billes 14

I.5.2.1 L'usure 14

I.5.3 Les fréquences des défauts 17

I.5.4 Les types de défauts 18

I.6 Conclusion 19

Chapitre II: Classification des signaux vibratoires 20

II.1 Introduction 21

II.2 Deep learning 21
II.3 Détection des anomalies 22
II.4 Type d'apprentissage 22
II.4.1 Apprentissage supervisé 22
II.4.2 Apprentissage non-supervisé 23
II.5 Les réseaux de neurones 23
II.5.1 Neurone biologique 23
II.5.2 Architecture des réseaux de neurones artificiels 24
II.5.3 Le perceptron 24
II.5.4 Fonction d'activation 25
II.5.5 Types de réseaux de neurones 27
II.6 Extraction des caractéristiques des signaux vibratoires 32
II.6.1 Kurtogramme 32
II.6.2 Spectrogramme 33
II.6.3 MFCC 33
II.7 Application de la méthode MFCC sur les signaux vibratoires 38
II.7.1 Base de données choisie 38
II.7.2 Traitement MFCC sur les signaux vibratoires 40
II.7.3 Création d'un dataset MFCC 41
II.7.4 Le modèle CNN 42
II.7.5 Résultats de classification 43
II.8 Conclusion 45
Chapitre III: Implémentation, tests et résultats 46
III.1 Introduction 47
III.2 Système implémenté 47
III.2.1 Schéma synoptique 47
III.2.2 Description du système 48
III.2.3 Le SoC Zynq-7000 48

III.2.4 Interfaces AXI 49
III.3 La plateforme PYNQ 50
III.3.2 La carte Pynq-Z2 52
III.4 Implémentation du classificateur CNN 53
III.4.1 Modèle IA sur plateformes embarquées 53
III.4.2 Présentation de la clé NCS2 53
III.4.3 Le cadre de travail OpenVino 54
III.4.4 Implémentation du réseau de neurones CNN 54
III.5 Implémentation du prétraitement sur plateforme FPGA 55
III.5.1 La synthèse haut niveau HLS 55
III.5.2 Vivado HLS 56
III.5.3 Procédure de création d'un IP-core sous Vivado HLS 56
III.5.4 Création des différents IP-cores 60
III.5.5 Banc de filtres 65
III.5.6 Transformé en cosinus discrète 65
III.5.7 Organigramme 66
III.5.8 Résultats de l'implémentation 67
III.6 Résultats du classificateur CNN 69
III.7 Conclusion 70
Conclusion générale 72
Références 73
Annexes 77
Annexe A 78
Annexe B 79

Liste des figures

Figure I-1 : Les différents types de maintenance 4
Figure I-2 : Maintenance corrective 5
Figure I-3 :Maintenance préventive systématique 5
Figure I-4 : Maintenance préventive conditionnelle 6
Figure I-5 : Évolution de la stratégie de maintenance (1930-2020) 7
Figure I-6 : Les différentes méthodes de diagnostic des machines tournantes 8
Figure I-7 : Une vibration peut être comparée à un système masse-ressort 9
Figure I-8 : Caractéristique d'une vibration (cas d'un signal sinusoïdal) 9
Figure I-9 : Grandeurs utilisées pour mesurer les vibrations et les plages de défauts associées 10
Figure I-10 : Exemple de signal dans le domaine temporel 11
Figure I-11 : Spectre d'amplitude du signal x(t) résultant du calcul de la DFT 12
Figure I-12 : Les sources de défaillances les plus rependues dans les moteurs 13
Figure I-13 : Structure mécanique d'un roulement à billes rigide 13
Figure I-14 : Défaut d'usure sur un roulement à billes 14
Figure I-15 : Défaut de dentelure sur la bague interne d'un roulement à bille 15
Figure I-16 : Défaut de bavure sur un roulement à bille 15
Figure I-17 : Défaut d'écaillage sur la bague interne d'un roulement à bille 16
Figure I-18 : Corrosion sur la bague externe d'un roulement à billes 16
Figure I-19 : Fissure sur la bague interne d'un roulement à billes 17
Figure I-20 : Choc électrique qui a endommagé un élément roulant 17
Figure I-21 : Défaut au niveau de la bague externe d'un roulement et le spectre d'amplitude associé 18
Figure I-22 : Défaut au niveau de la bague interne d'un roulement et le spectre d'amplitude associé 19
Figure I-23 : Défaut au niveau de la bille d'un roulement et le spectre d'amplitude associé 19

Figure II-1 : Les domaines de l'intelligence artificielle 21
Figure II-2 : Apprentissage supervisé 22
Figure II-3 : Apprentissage non-supervisé 23
Figure II-4 : Structure d'un neurone biologique 23

Figure II-5 : Architecture d'un réseau de neurones artificiels 24
Figure II-6 : Neurone artificiel simple 'Le perceptron' 25
Figure II-7 : Forme des fonctions d'activations les plus utilisées 27
Figure II-8 : Architecture d'un réseau CNN 28
Figure II-9 : Opération de convolution 29
Figure II-10 : Application de la fonction d'activation ReLU 29
Figure II-11 : Opération de MaxPooling 30
Figure II-12 : Rôle de la fonction d'activation Softmax dans un réseau CNN 31
Figure II-13 : Introduction du Dropout dans un réseau ANN, la moitié des neurones sont désactivés 32
Figure II-14 : Exemple de kurtogramme utilisé pour le filtrage d'un signal vibratoire 32
Figure II-15 : Exemple de spectrogramme 33
Figure II-16 : Représentation temporelle et fréquentielle de la fenêtre de Hamming 34
Figure II-17 : Banc de filtres de Mel (exemple de 20 filtres) 37
Figure II-18 : Étapes de calcul de la matrice MFCC 38
Figure II-19 : Banc d'essai des défauts des roulements 38
Figure II-20 : La longueur d'un segment permet de contenir au moins 3 impulsions 40
Figure II-21 : Conversion du signal vibratoire (8192 points) en image mfcc(20 lignes, 13 colonnes) 41
Figure II-22 : Partage des images du dataset des images MFCC 41
Figure II-23 : Répartition des images utilisées pour entraîner le modèle CNN dans les différentes classes 42
Figure II-24 : modèle CNN proposé 43
Figure II-25 : Résultat du modèle CNN proposé 44
Figure II-26 : Résultat de prédiction sur un signal vibratoire d'un défaut de bague interne 45

Figure III-1 : Schéma bloc du système réalisé 47
Figure III-2 : Schéma bloc simplifié du SoC Zynq 49
Figure III-3 : Couches du cadre de travail Pynq 50
Figure III-4 : Exemple d'un script python sur jupyter notebook 52
Figure III-5 : Photographie de la carte FPGA, PYNQ-Z2 52
Figure III-6 : Clé Intel Neural Compute Stick 2 53
Figure III-7 : Clé Intel NCS2 connectée avec la carte PYNQ-Z2 à travers l'interface USB 54
Figure III-8 : Conversion du modèle CNN en image IR pour la clé NCS2 55

Figure III-9 : Etapes de démarrage d'un nouveau projet Vivado HLS.....57
Figure III-10 : Test de l'algorithme à implémenter avec un testbench C/C++.....57
Figure III-11 : Ajout des directives pour orienter HLS durant la synthèse de l'IP-core.....58
Figure III-12 : Vérification des ressources matérielles et du temps d'exécution estimé.....59
Figure III-13 : IP-core résultant «Add » implémenté dans Vivado Design suite.....59
Figure III-14 : Test de l'IP-core « add » sur la carte FPGA.....60
Figure III-15 : Organigramme de développement d'un IP-core sous Vivado HLS.....60
Figure III-16 : Architecture du système de prétraitement implémenté sur Vivado Design Suite.....61
Figure III-17 : Éléments de la hiérarchie Spectrogram.....62
Figure III-18 : Impacte du choix d'interface sur le temps de traitement, la donnée suit.....62
Figure III-19 : Calcul du spectre de puissance de x[n] sur la carte PYNQ-Z2.....63
Figure III-20 : IP-core « hamming_win » et ses différentes interfaces de connection.....64
Figure III-21 : IP-core fft V9.1 de Xilinx.....64
Figure III-22 : IP-core « fft_abs » et ses différentes interfaces.....65
Figure III-23 : IP-core « mel-filter » et ses différentes interfaces.....65
Figure III-24 : IP-core « dct » et ses différentes interfaces.....66
Figure III-25 : Organigramme de calcul de la matrice MFCC à l'aide des IP-cores sur PYNQ-Z2.....66
Figure III-26 : Comparaison entre le traitement mfcc du même signal vibratoire.....67
Figure III-27 : Résultat du modèle CNN sur le système embarqué.....69
Figure III-28 : Résultat de prédiction sur un signal vibratoire d'un défaut de bague externe.....70

Liste des tableaux

Tableau II-1 : Comparaison entre la DFT et la FFT en terme de complexité de calcul 36
Tableau II-2 : Fréquences des défauts de roulements.....39

Tableau III-1 : Résultats de synthèse et d'implémentation.....68
Tableau III-2 : Comparaison des temps d'exécution µC / IP-core 69

Introduction générale

Les machines tournantes sont des éléments clés dans le développement des industries. Cependant pour assurer la qualité, la fiabilité et la sécurité de ces machines, il est nécessaire d'appliquer une stratégie de maintenance adéquate.

La maintenance préventive est la solution incorporée dans les systèmes de gestion qui a fait ces preuves en réduisant le temps d'immobilisation, optimiser la fabrication, les ressources et la rentabilité de production. Mais ce type de stratégie a l'inconvénient d'avoir des enjeux économiques énormes en raison du suivi quotidien de l'état des machines par les équipes de maintenance.

L'analyse vibratoire est l'outil le plus utilisé pour traquer les défauts dans les machines tournantes. Les vibrations perçues peuvent être analysées automatiquement par des systèmes électroniques pré-entrainés à reconnaitre le moindre signe de défaut. Ceci permet de définir le juste nécessaire au niveau des interventions de maintenance, la machine est alors révisée seulement si son état le nécessite, ce qui a pour effet de réduire considérablement les temps d'arrêts de production les coûts de maintenance.

Le but de ce projet de fin d'étude est d'implémenter une méthode d'extraction des informations utiles d'un signal vibratoire sur une plateforme embarquée de type FPGA, puis d'utiliser ces informations pour déduire l'origine du défaut dans une machine tournante. Pour parvenir à cet objectif, notre travail a été structuré en trois chapitres :

Le premier chapitre donne un aperçu sur les différentes stratégies de maintenance et les méthodes de détection des défauts et de prévision des pannes des machines tournantes, en mettant le plus d'attention sur les défauts issus des roulements à billes.

Le second chapitre est consacré à la recherche d'un prétraitement qui a pour but de transformer l'information vibratoire temporelle, en autre représentation de taille plus réduite sous forme d'image. Cette dernière servira comme d'entrée à un réseau de neurone à convolution chargé de prédire le type de défaut.

Dans le dernier chapitre, nous présenterons l'implémentation la méthode de prétraitement proposée, et le modèle d'intelligence artificielle conçu au cours du deuxième chapitre sur une plateforme embarquée composée d'une carte FPGA et d'un accélérateur matériel de type NCS2 d'Intel. Enfin nous présenterons des tests effectués pour valider l'efficacité du traitement proposé.

Chapitre I: Diagnostic des machines tournantes par analyse vibratoire

I.1 Introduction

Les machines tournantes sont des éléments essentiels des applications industrielles et de la production d'énergie. Le dysfonctionnent prématuré de ces machines peut entraîner des pertes économiques majeures, en raison d'arrêts non planifiés et d'éventuels dommages sur les autres équipements.

Les entreprises apportent de plus en plus d'importance à la maintenance car cette dernière permet de réduire les temps d'arrêt de production, de diminuer les coûts et d'allonger la durée de vie des équipements. Les défauts des machines tournantes apparaissent généralement sous forme de bruits, de surchauffes et de vibrations. Ces grandeurs physiques permettent de surveiller et de donner des indications propres sur l'état des machines.

Les machines tournantes comportent plusieurs éléments dans leur construction, et les roulements à billes sont sans doute les éléments les plus amenés à tomber en panne dans à cause des importantes charges qu'ils supportent et des très hautes températures de fonctionnement qu'ils subissent. Par conséquent, il est très intéressant de développer des systèmes de diagnostic pour détecter, de manière anticipée, les défauts pouvant surgir dans ces éléments roulants.

Dans ce chapitre nous présentons la détection des défauts des machines tournantes par l'analyse vibratoire en se focalisant sur les défauts des roulements à billes.

I.2 Stratégie de maintenance

I.2.1 Définition de la maintenance

La norme Afnor NF X 60-010 définie la maintenance comme suit : « La maintenance est l'ensemble des actions permettant de maintenir ou de rétablir un bien dans un état spécifié ou en mesure d'assurer un service déterminé » [1].

Dans une entreprise, maintenir, c'est donc effectuer des opérations (dépannage, réparation, graissage, contrôle, etc) qui permettent de conserver le potentiel du matériel pour assurer la production avec efficacité et qualité [2].

I.2.2 Objectif de la maintenance

L'objectif de la maintenance se résume dans les points suivants :

- Optimisation des coûts de production
- Réduction des temps d'arrêt des machines
- Augmentation de la durée de vie des équipements
- Minimiser les coûts de réparation

- Amélioration de la qualité des produits
- Garantir la sécurité des personnes et des biens

I.2.3 Rôle de la maintenance dans une entreprise

La stratégie de maintenance, qui résulte de la politique de maintenance, impose des choix pour atteindre, voir dépasser, les objectifs fixés. Ces choix sont effectués pour [1]:

- Développer, adapter ou mettre en place des méthodes de maintenance ;
- Élaborer et optimiser les gammes de maintenance ;
- Organiser les équipes de maintenance ;
- Internaliser et/ou externaliser partiellement ou totalement les tâches de maintenance ;
- Définir, gérer et optimiser les stocks de pièces de rechanges et de consommables ;
- Étudier l'impact économique (temps de retour sur investissement) de la modernisation ou de l'amélioration de l'outil de production en matière de productivité et de maintenabilité.
- La stratégie de maintenance implique la mise en œuvre d'un plan de maintenance avec des objectifs chiffrés et des indicateurs mesurables.

I.2.4 Les types de maintenance

La présence ou l'absence de panne sur le matériel permet de scinder la maintenance en deux types principaux (figure I-1). La maintenance corrective, qui s'applique après la défaillance de la machine, et la maintenance préventive qui est effectuée avant que la panne ne se manifeste.

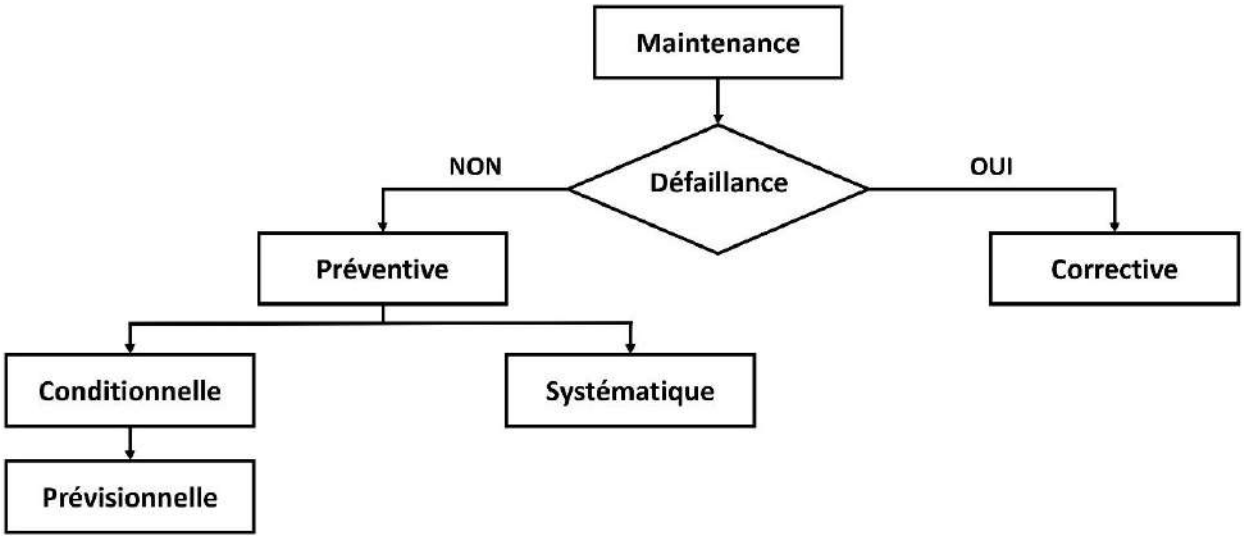


Figure I-1 : Les différents types de maintenance

I.2.4.1 Maintenance corrective

La maintenance corrective, anciennement connue sous le nom de maintenance curative, est historiquement la première génération de maintenance. Elle est exécutée après la détection d'une panne et est destinée à remettre un bien dans un état dans lequel il peut accomplir une fonction requise [1] (figure I-2). Elle peut être différée ou retardée si elle n'est pas exécutée immédiatement après la détection de la panne, ou effectuée d'une façon urgente sans délais après déclaration de la panne, ceci afin d'éviter des conséquences inacceptables.

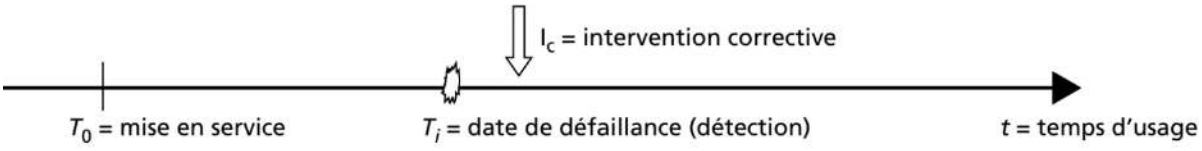


Figure I-2 : Maintenance corrective [3]

Ce type de maintenance convient seulement pour les petites installations qui peuvent être réparées de manière aisée et dont les pièces de rechange sont facilement conservables en stock. Cette stratégie est aujourd'hui délaissée car elle cause des pertes de profit à cause des mises en arrêt imprévues et de la dégradation rapide des machines au point où elles doivent être complètement remplacées.

I.2.4.2 Maintenance préventive

Maintenance exécutée à des intervalles prédéterminés ou selon des critères prescrits. Elle est destinée à réduire la probabilité des défaillances ou la dégradation du fonctionnement d'un bien [1]. La maintenance préventive se divise en :

I.2.4.2.1 Maintenance systématique

Historiquement elle représente la seconde génération de maintenance, c'est un type de maintenance préventive qui est exécuté à des intervalles de temps préétablis ou selon un nombre défini d'unités d'usage mais sans contrôle préalable de l'état du bien (figure I-3).

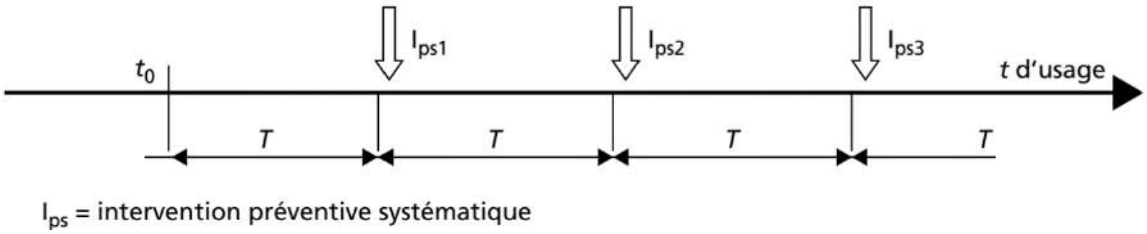


Figure I-3 :Maintenance préventive systématique [3]

Ce type de maintenance crée un compromis entre la préservation de l'état des équipements dans le temps, si les opérations de maintenance sont exécutées de manière trop

fréquente, et une hausse des coûts des opérations dans le cas contraire. Une énorme responsabilité repose alors sur les équipes de maintenance qui ont pour rôle de décider d'un intervalle de temps idéal pour chaque machine.

I.2.4.2.2 Maintenance conditionnelle

Cette dernière rpésente la troisième génération de maintenance. C'est une catégorie de maintenance préventive basée sur une surveillance du fonctionnement du bien et/ou des paramètres significatifs de ce fonctionnement et intégrant les actions qui en découlent [1] (figure I-4).

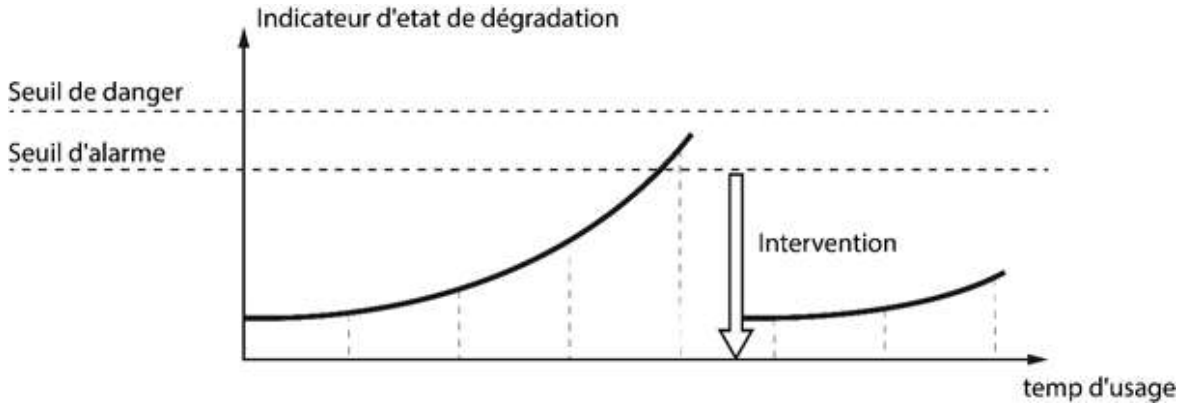


Figure I-4 : Maintenance préventive conditionnelle

Contrairement à la maintenance systématique, la maintenance conditionnelle est uniquement appliquée après que l'état de l'équipement ait dépassé un certain seuil de dégradation. L'entretien est effectué selon les besoins de la machine, ce qui permet d'allonger l'intervalle de temps idéal d'intervention.

I.2.4.2.3 Maintenance prévisionnelle

La maintenance prévisionnelle est une variante de la maintenance conditionnelle, elle est exécutée en suivant les prévisions extrapolées de l'analyse et de l'évaluation de paramètres significatifs de la dégradation du bien [1].

I.3 Implication de l'IA et l'IOT dans la maintenance

Dans son histoire, l'industrie est passée par plusieurs points majeurs pour arriver à la forme que l'on connaît aujourd'hui. La première révolution industrielle est apparue au Royaume-Uni dans le dernier quart du 18^{e} siècle et s'est caractérisée par l'introduction des machines hydrauliques et à vapeur. La seconde révolution à la moitié du 19^{e} siècle était marquée par des chaînes de production de masse qui utilisaient l'électricité comme source d'énergie principale. L'introduction de l'électronique numérique et des ordinateurs au début des années

1970, a permis, quant à elle, de donner naissance à une troisième révolution industrielle qui s'est distinguée par l'utilisation intensive des dispositifs automatisés.

Dans le futur nous assisterons à l'apparition d'une nouvelle révolution industrielle ou industrie 4.0. Cette dernière se démarque des générations précédentes par l'introduction de systèmes cyber-physiques et de l'internet des objets pour la gestion virtuelle des parcs de production, des stocks en matière première et des outils de transport, mais aussi d'une grande dépendance entre les besoins des clients et la cadence de fonctionnement des machines.

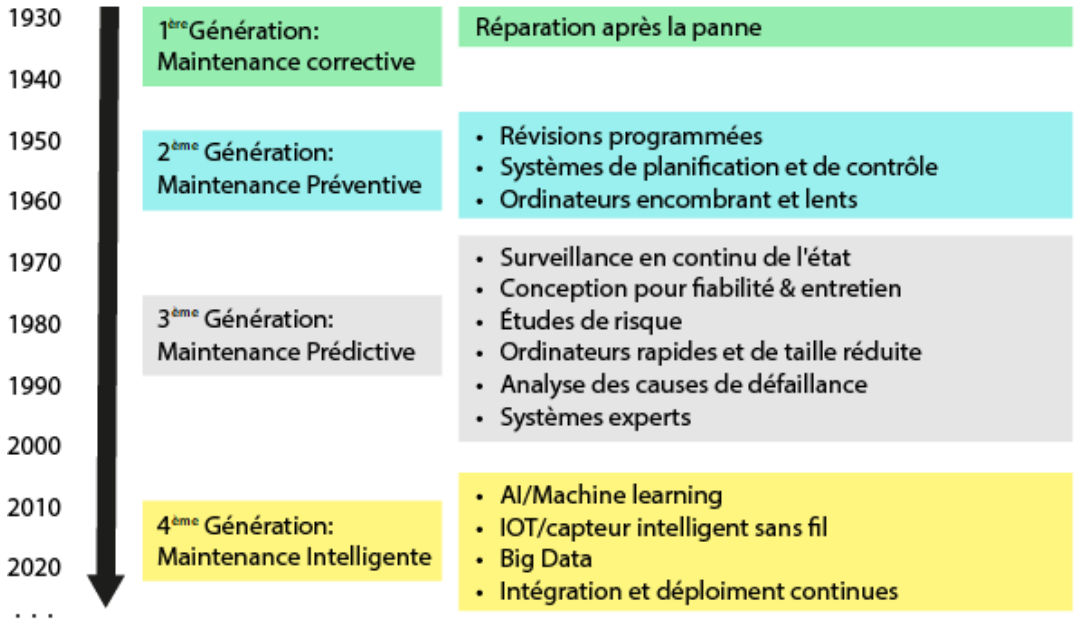


Figure I-5 : Évolution de la stratégie de maintenance (1930-2020) [4]

Pour assurer le bon fonctionnement des machines dans l'usine du futur, la maintenance du matériel de production devra être adaptée et optimisée. Cette nouvelle stratégie de maintenance, appelée maintenance 4.0 repose essentiellement sur l'introduction de l'internet des objets (IoT) et l'intelligence artificielle (IA) dans le monde de l'industrie (figure I-5). L'utilisation de l'IoT permet d'optimiser la maintenance des équipements industriels en temps réel, ce nouveau mode de surveillance à l'avantage de permettre aux équipes techniques d'éviter les déplacements inutiles et de réduire le temps et les délais d'intervention. Afin de traiter les grandes quantités de données perçus par les capteurs, une IA est chargée d'analyser les données les plus significatives dans le but de détecter les anomalies de manière plus rapide et d'anticiper les pannes. L'association de l'IoT et IA dans le domaine de la maintenance permettra donc aux industriels une gestion optimale des opérations d'entretiens et des stocks de pièces de rechange.

I.4 Analyse vibratoire des machines tournantes

Durant leur fonctionnement, les machines tournantes génèrent des efforts divers tels que des secousses, des chocs ou des frottements qui causent l'usure des différents éléments qui les composent. On peut alors juger de l'état de dégradation de la machine en s'appuyant sur plusieurs indicateurs tels que la chaleur dégagée, l'analyse de débris dans l'huile ou encore les bruits et vibrations émis au cours du fonctionnement.

Les vibrations dépendent directement des forces générées par les différents éléments internes en mouvement, pondérées par la fonction de transfert des liaisons (roulements, film d'huile, film d'air, boulons, supports élastiques, ressorts), et présentent des grands avantages en raison de la richesse des informations qu'elles véhiculent, de leur transmission quasi instantanée et de leur rapport direct avec les mouvements de la machine. Ainsi les vibrations occupent une place privilégiée parmi les paramètres à prendre en considération pour assurer une surveillance efficace du bon état de fonctionnement des machines tournantes. C'est pourquoi, la plupart des méthodes modernes de maintenance des machines font appel à l'analyse du comportement vibratoire (figure I-6), qui s'est aujourd'hui très fortement répandue dans l'industrie [5].

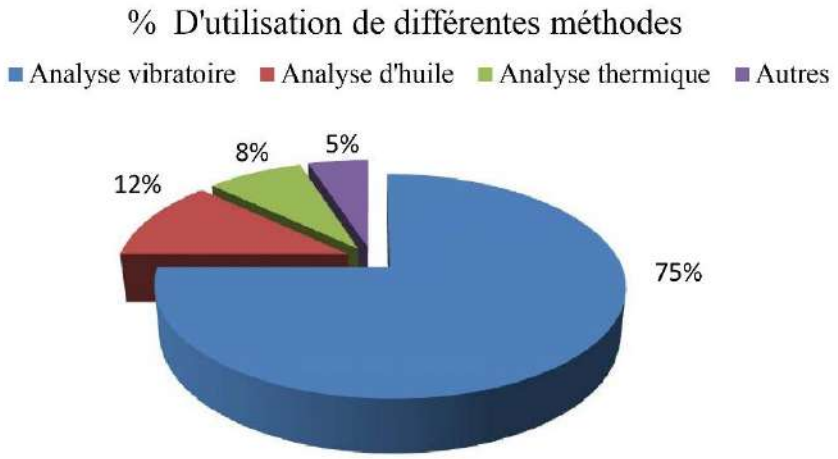


Figure I-6 : Les différentes méthodes de diagnostic des machines tournantes [6]

I.4.1 Définition d'une vibration

Un système mécanique est dit en vibration lorsqu'il est animé d'un mouvement de va-et-vient rapide autour d'une position moyenne appelée « position d'équilibre » [7], (figure I-7).

La transmission du mouvement vibratoire peut se faire par deux moyens : soit par un gaz ou un fluide, généralement de l'air, dans ce cas on parle de « son » ou signal acoustique. Ou bien à travers d'un solide, comme la structure de la machine, et on parle alors de vibration dite « solidienne » ou plus couramment de signal vibratoire.

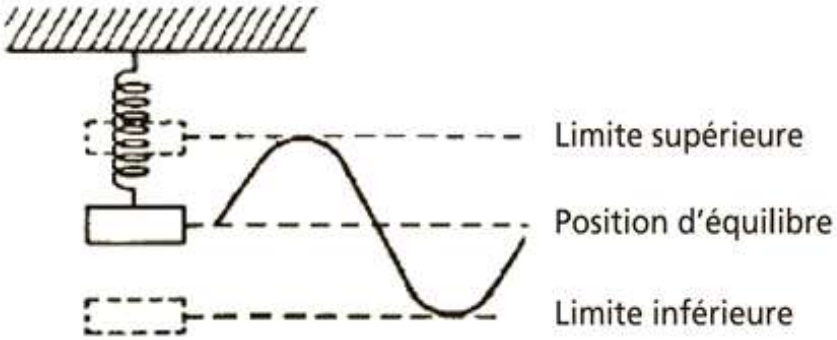


Figure I-7 : Une vibration peut être comparée à un système masse-ressort [7]

I.4.2 Caractéristique d'une vibration

Une vibration est caractérisée par son amplitude et sa fréquence (figure I-8). La fréquence f représente la cadence de répétition d'un phénomène ou le nombre de fois qu'il se reproduit en un temps donné. Lorsque l'unité de temps choisie est la seconde, la fréquence s'exprime en Hertz (Hz) [7]. La période T est l'intervalle de temps qui sépare deux motifs identiques, elle est liée à la fréquence par une relation d'inversion $T = 1/f$. On appelle amplitude d'un mouvement vibratoire la valeur de ses écarts par rapport à sa position d'équilibre [7], elle est généralement notée A.

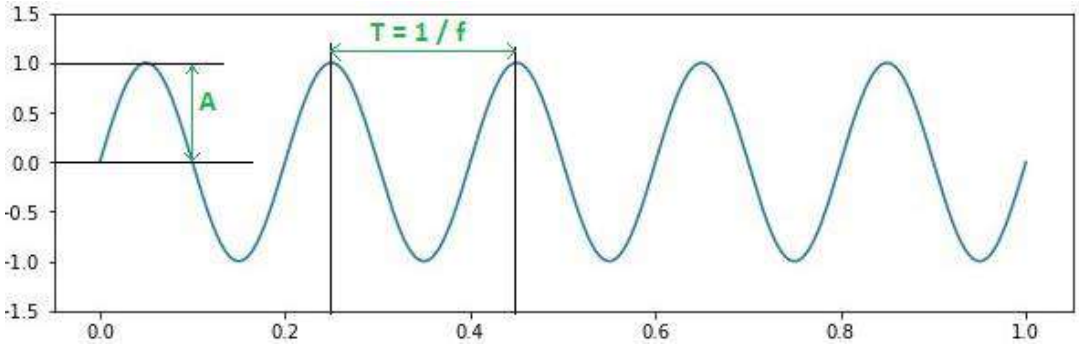


Figure I-8 : Caractéristique d'une vibration (cas d'un signal sinusoïdal)

I.4.3 Grandeurs associées

Une vibration se caractérise par trois grandeurs cinématiques : le déplacement exprimé en [µm], la vitesse en [mm/s] et l'accélération en [m/s²]. A ces grandeurs sont associées une période temporelle et une phase. Ainsi, dans le cas d'une machine tournante, une vibration élémentaire peut se mettre sous la forme :

$$x(t) = A.\sin(2\pi ft + \theta) \qquad \text{Éq 1}$$

Une propriété fondamentale relie les trois représentations : déplacement D, vitesse V et accélération a :

$$D = \frac{V}{2\pi f} = \frac{a}{(2\pi f)^2} \qquad \text{Éq 2}$$

Ces relations mettent en lumière l'importance du choix de la grandeur physique à mesurer pour la surveillance vibratoire d'une machine tournante. Ainsi, la mesure du déplacement favorise les composantes basses fréquences tandis que celle de l'accélération conduira à privilégier les composantes hautes fréquences (figure I-9). En d'autres termes, un phénomène vibratoire de faibles fréquences se traduira par un déplacement significatif. Par contre, la mesure de l'accélération permettra de mettre en évidence des phénomènes de fréquences élevées [8].

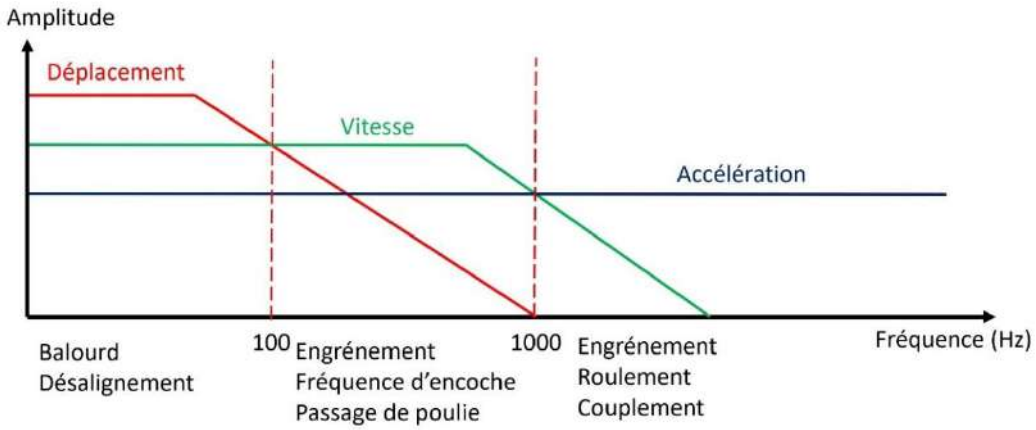


Figure I-9 : Grandeurs utilisées pour mesurer les vibrations et les plages de défauts associées [9]

I.4.4 Analyse temporelle et analyse fréquentielle

Une méthode appropriée doit être choisie pour analyser les signaux vibratoires qui proviennent des machines durant leurs fonctionnements. Les plus utilisées sont l'analyse temporelle et l'analyse fréquentielle.

I.4.4.1 Analyse temporelle

Dans cette méthode, l'analyse ou l'interprétation du signal vibratoire est perçu au travers de divers paramètres ou indicateurs [10], les plus connus sont :

- La valeur efficace du signal (RMS) :

$$RMS = \sqrt{\frac{1}{N}\sum_{i=1}^{N} x_i^2} \qquad \text{Éq 3}$$

- Le facteur de crête :

$$CRF = \frac{|x|_{peak}}{RMS} \qquad \text{Éq 4}$$

- Le facteur d'impulsion :

$$IMF = \frac{|x|_{peak}}{\frac{1}{N}\sum_{i=1}^{N}|x_i|} \qquad \text{Éq 5}$$

- Le kurtosis :

$$K = \frac{\frac{1}{N}\sum_{i=1}^{N}(x_i - \bar{x})^4}{RMS^4} \qquad \text{Éq 6}$$

Où :

x : valeurs successives du signal x(t) échantillonné

$|x|_{peak}$: valeur pique du signal x

$\bar{x}$: valeur moyenne du signal x, avec $\bar{x} = \frac{1}{N}\sum_{i=1}^{N} x_i$

La figure I-10 présente un exemple de signal temporel représenté par l'équation :

$$x(t) = \frac{1}{2}\sin(2.\pi.t.500) + \sin(2.\pi.t.1000) + \frac{1}{2}\sin(2.\pi.t.2000) \qquad \text{Éq 7}$$

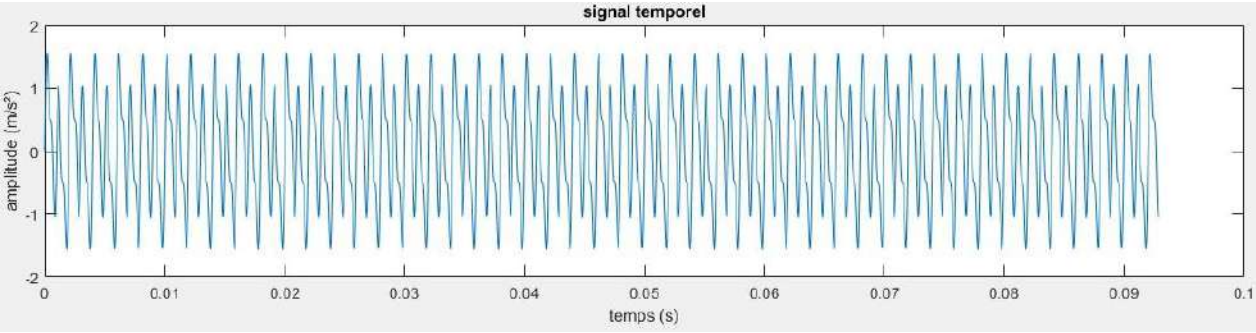


Figure I-10 : Exemple de signal dans le domaine temporel

Ce signal est caractérisé par $RMS = 0.86602, CRF = 1.7989, IMF = 2.0608, K = 1.9166$

I.4.4.2 Analyse fréquentielle

On peut aussi représenter l'information contenue dans un signal dans le domaine fréquentiel, il s'agit de représenter les fréquences ou harmoniques contenues dans le signal. Ce type de représentation est appelé spectre.

Le spectre d'un signal est obtenu en effectuant la transformée de Fourier du signal temporel [11]:

Dans le cas des signaux continus (TF) :

$$X(f) = \int_{-\infty}^{+\infty} x(t)\, e^{-2j\pi ft}.dt \qquad \text{Éq 8}$$

Où :

- t : représente la variable temporelle
- f : représente la variable fréquentielle

Et dans le cas des signaux discrets (DFT) :

$$X(k) = \sum_{n=0}^{N-1} x[n].e^{-2j\pi k\frac{n}{N}}, k \in \{0, 1, \dots, N-1\} \qquad \text{Éq 9}$$

Où :

- N : est le nombre de points temporels (i.e nombre d'échantillons)
- n : indice de temps
- k : indice de fréquence

En effectuant cette transformée sur le même signal précédant on obtient le spectre représenté sur la figure I-11.

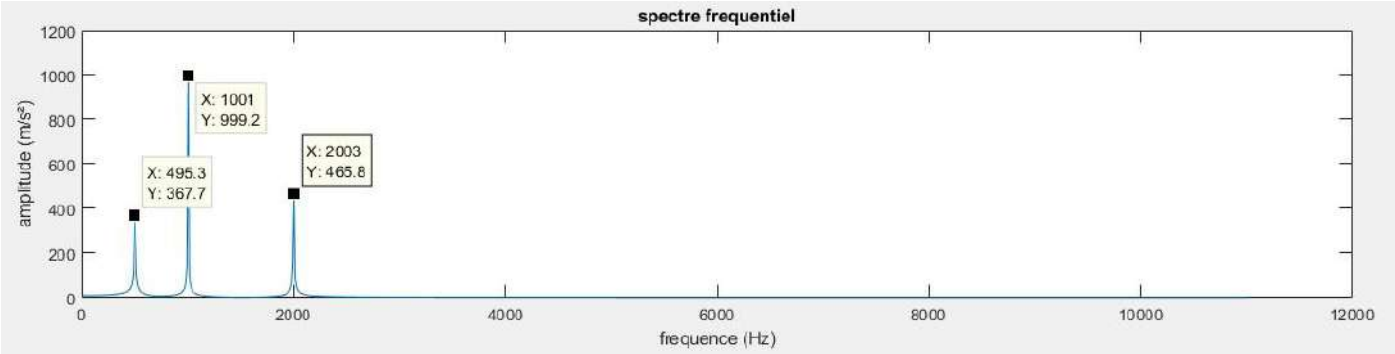


Figure I-11 : Spectre d'amplitude du signal x(t) résultant du calcul de la DFT

I.5 Les défauts des roulements

Les études statistiques montrent que dans les machines tournantes, les roulements sont les éléments qui tombent en panne le plus fréquemment. On a également observé une nette augmentation des défauts de roulements en comparant deux études, l'une établie en 1999 et l'autre en 2008, (figure I-12).

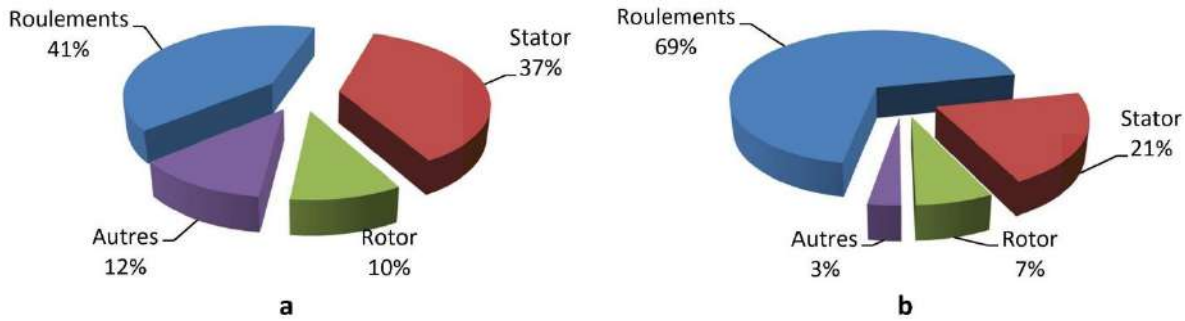


Figure I-12 : Les sources de défaillances les plus rependues dans les moteurs [12]

a : étude établies en 1999, b : étude établie en 2008

I.5.1 Structure mécanique d'un roulement

Le roulement à billes rigide, (appelé aussi roulement à billes à contact radial) est n composant mécanique formé par deux bagues concentriques appelées bagues intérieure et extérieure comportant des chemins de roulement entre lesquelles existent des corps roulants uniformément espacés par une cage. Cette cage est placée entre les bagues et a pour rôle de réduire les frottements en empêchant les éléments de se frotter les uns contre les autres (figure I-13-a/b).

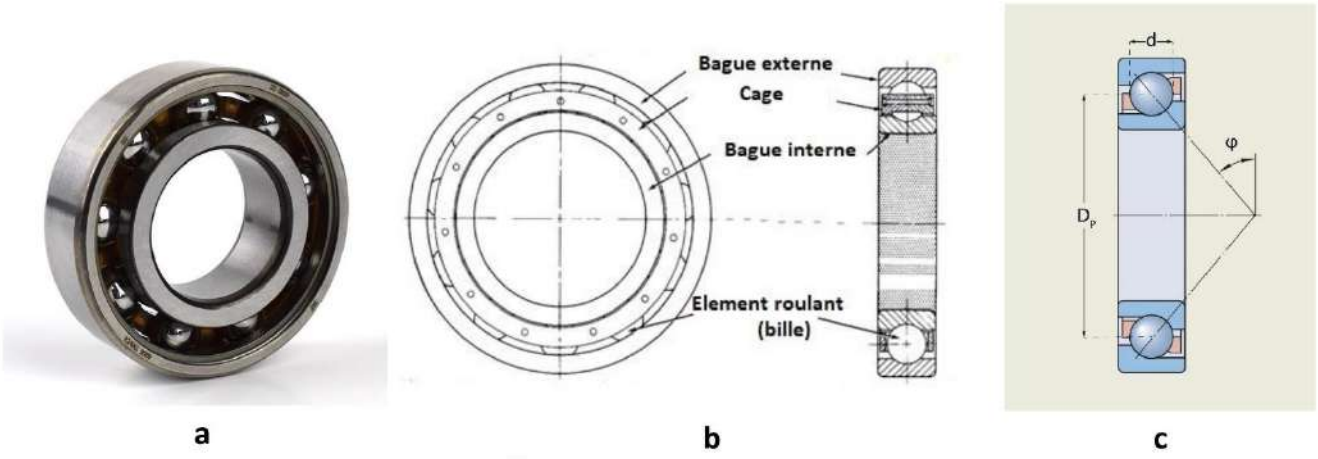


Figure I-13 : Structure mécanique d'un roulement à billes rigide

a : photographie d'un roulement réel, b : dessin technique d'un roulement et ses composants, c : caractéristiques géométriques d'un roulement

Les dimensions et les spécifications d'un roulement à billes peuvent être visualisées sur la figure I-13-c, où :

- d : est le diamètre de l'élément roulant (bille)
- D_p : est le diamètre moyen de rotation (moyenne des diamètres de la bague interne et externe)

- φ : est l'angle de contact

Un autre paramètre géométrique important est le nombre d'éléments qui compose le roulement à billes, et qui est généralement noté N.

I.5.2 Causes des défauts des roulements à billes

Dans la plupart des cas, les défauts des roulements commencent par une perte locale de matériau sur une surface mate.

I.5.2.1 L'usure

L'usure provient généralement de trois causes principales. L'usure peut provenir de particules qui pénètrent entre le corps roulant et les bagues intérieure et extérieure. Cela se manifeste généralement lorsqu'il y a un manque de propreté lors de l'assemblage du roulement ou si le lubrifiant est contaminé par des particules. Aussi l'usure peut encore se manifester à cause d'une lubrification légère ou manquante. La dernière cause qui est responsable d'une telle usure provient d'une vibration excessive qui atteint le roulement sans que celui-ci ne tourne [13] (figure I-14).



Figure I-14 : Défaut d'usure sur un roulement à billes [14]

I.5.2.2 Dentelure

Ce défaut apparaît généralement lorsque le roulement n'est pas correctement monté ou lorsque le roulement est soumis à une grande charge lorsqu'il est immobile. Une dentelure apparaît, dont la distance entre les fissures est égale à la distance entre les corps roulants, (figure I-15) [13].

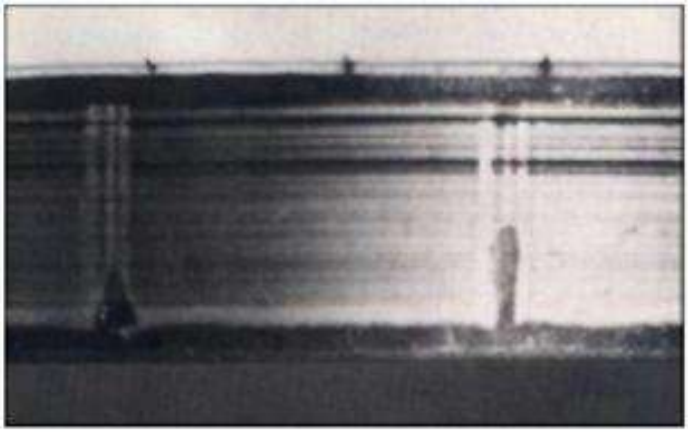


Figure I-15 : Défaut de dentelure sur la bague interne d'un roulement à billes [13]

I.5.2.3 Bavure

Ce phénomène apparaît sous forme d'un transfert de matière, d'une surface à une autre. Cela peut être causé par plusieurs facteurs : un manque de lubrification, la pénétration du corps roulant dans la zone chargée, une charge légère par rapport à la vitesse de rotation. Chaque cas doit être traité de façon à éliminer la cause. Le développement de la bavure conduit à une cassure totale de la bague ou du corps roulant [13], (figure I-16).



Figure I-16 : Défaut de bavure sur un roulement à billes [13]

I.5.2.4 Ecaillage

Il se caractérise par des traces de fissuration et d'arrachement de fragments de matière. Il apparaît normalement à cause du vieillissement du roulement. Mais l'écaillage arrive aussi à cause d'une grande différence de température entre la bague intérieure et extérieure. Également, si le boitier qui porte le roulement possède une forme ovale, le même résultat est observé, il faut alors changer le boîtier immédiatement. D'autres causes de l'écaillage, peuvent être dues à un montage incorrect du roulement ou une liberté axiale légère qui n'est pas suffisante pour le mouvement de la dilatation [13], (figure I-17).



Figure I-17 : Défaut d'écaillage sur la bague interne d'un roulement à billes [13]

I.5.2.5 Corrosion

Elle se manifeste par des taches d'oxydation sur l'alésage, le diamètre extérieur et les faces d'appui du roulement. Elle apparaît après la présence d'eau ou d'humidité pour une longue période dans le roulement. La corrosion peut entraîner la cassure d'une bague ou du corps roulant. Pour la corriger, il faut utiliser un meilleur lubrifiant antirouille de bonne qualité [13] (figure I-18).



Figure I-18 : Corrosion sur la bague externe d'un roulement à billes [13]

I.5.2.6 Coup, fissures, Cassures

Ils se manifestent avec des ruptures des bagues et des traces visibles d'outils ; ils proviennent dans la plupart des cas d'une mauvaise manipulation du roulement, par exemple un coup de marteau au moment du montage. La cassure peut encore provenir d'une corrosion ou d'une bavure [13], (figure I-19).

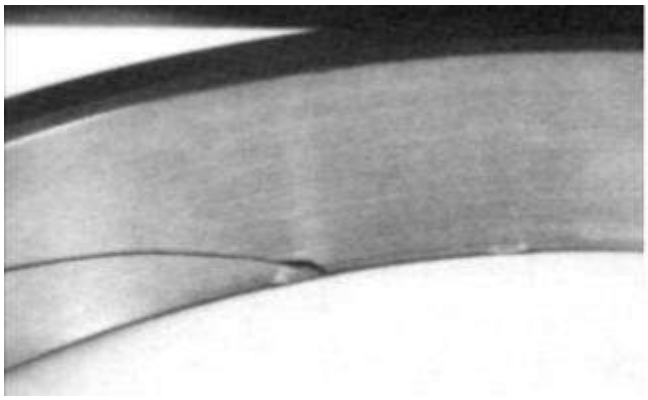


Figure I-19 : Fissure sur la bague interne d'un roulement à billes [13]

I.5.2.7 Chocs électriques

Le passage d'un courant électrique dans un roulement en rotation peut endommager ses différentes parties [13], (figure I-20).

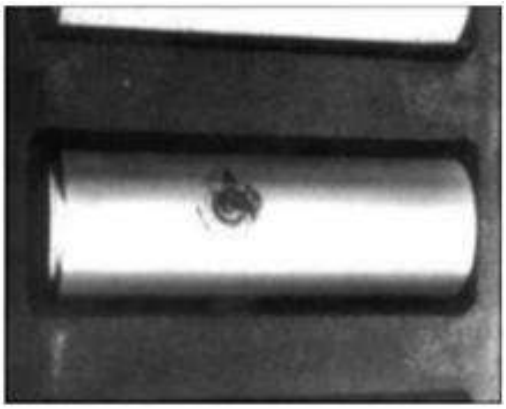


Figure I-20 : Choc électrique qui a endommagé un élément roulant [13]

I.5.3 Les fréquences des défauts

Lorsqu'il y a écaillage ou indentation sur l'un de ces éléments du roulement, le choc se produit à la fréquence correspondante donnée par les formules ci-dessous tirées de la formule de Willis [7]. Ces fréquences dépendent à la fois des caractéristiques géométriques du roulement (d , D_p, φ et N) mais aussi de vitesse de rotation de l'axe qui entraîne la bague interne du roulement F_{rot}.

- La fréquence de passage d'un élément roulant sur un défaut de bague externe (la bague externe étant supposée fixe), F_{b_ext} est donnée par la relation suivante :

$$F_{b_ext} = \frac{N}{2}[1 - \frac{d\cos\varphi}{D_p}]F_{rot} \qquad \text{Éq 10}$$

- La fréquence de passage d'un élément roulant sur un défaut de bague interne (la bague interne étant supposée montée sur l'arbre tournant), F_{b_int}, elle est donnée par la relation suivante :

$$F_{b_int} = \frac{N}{2}[1 + \frac{d\cos\varphi}{D_p}]F_{rot} \qquad \text{Éq 11}$$

- La fréquence de passage de rotation d'un élément roulant sur lui-même, F_{bille} est donnée par l'équation suivante :

$$F_{bille} = \frac{D_p}{2d}[1 - \frac{d^2(\cos\varphi)^2}{D_p^2}]F_{rot} \qquad \text{Éq 12}$$

Mais comme le choc dû au défaut se produit à chaque rotation de l'élément altéré, à la fois sur la bague interne et sur la bague externe, la fréquence du défaut d'élément roulant correspond généralement au double de leur fréquence de rotation sur eux-mêmes.

I.5.4 Les types de défauts

On distingue les types de défauts selon l'arrachement de matériau sur les éléments du roulement.

I.5.4.1 Défaut au niveau de la bague externe

Un défaut de type écaillage affectant la bague externe d'un roulement a pour image vibratoire un peigne de raies dont le pas correspond à la fréquence du défaut (figure I-21). À chaque composante de ce peigne, est associée une paire de bandes latérales espacées de la fréquence de rotation, en cas de charge dynamique importante [15].

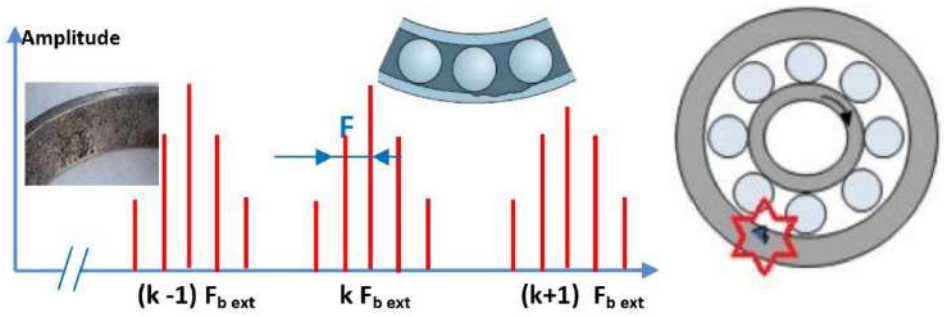


Figure I-21 : Défaut au niveau de la bague externe d'un roulement et le spectre d'amplitude associé [15]

I.5.4.2 Défaut au niveau de la bague interne

Un défaut de type écaillage affectant la bague interne de roulement a pour image vibratoire un peigne de raies (figure I-22). A chaque composante de ce peigne, sont associées plusieurs paires de bandes latérales espacées de la fréquence de rotation [15].

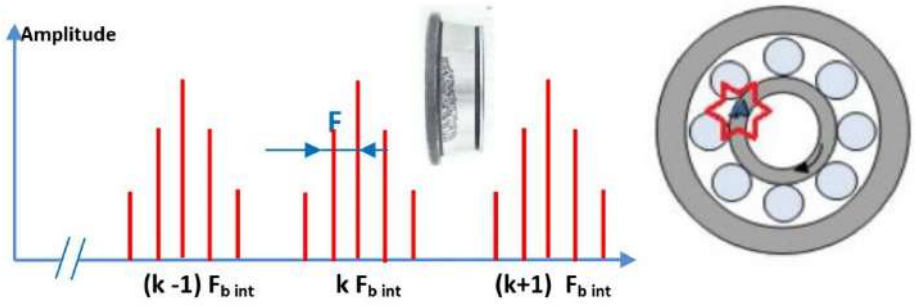


Figure I-22 : Défaut au niveau de la bague interne d'un roulement et le spectre d'amplitude associé [15]

I.5.4.3 Défaut au niveau de l'élément roulant

Un défaut de type écaillage sur un élément roulant (bille, rouleau ou aiguille) a pour image vibratoire un peigne de raies (figure I-23). A chaque composante de ce peigne, sont associées plusieurs paires de bandes latérales espacées de la fréquence de la cage [15].

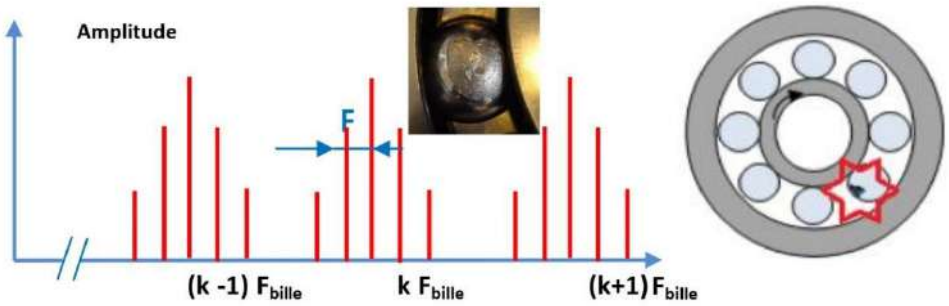


Figure I-23 : Défaut au niveau de la bille d'un roulement et le spectre d'amplitude associé [15]

I.6 Conclusion

Dans ce chapitre nous avons exposé les différentes stratégies de maintenance qui sont adoptées dans le domaine industriel, nous avons aussi présenté l'utilisation de l'analyse vibratoire et son rôle dans la détection des défauts dans les machines tournantes. Et nous nous sommes finalement penchés sur l'utilisation de cette méthode pour se focaliser sur le diagnostic des défauts de roulements.

Chapitre II: Classification des signaux vibratoires

II.1 Introduction

Les données que les machines informatiques traitent aujourd'hui sont de plus en plus complexes, il est très difficile voire impossible d'écrire manuellement des programmes informatiques qui permettent d'extraire l'information utile de ces données. L'émergence de méthode d'apprentissage automatique ML et d'apprentissage profond DL donnent la possibilité d'éduquer les ordinateurs à partir d'un grand ensemble d'exemples et d'appliquer leurs connaissances acquises dans des domaines très pointues.

Les modèles les plus sophistiqués du Deep Learning comme par exemple Yolo-V3 et LeNet5 sont composées de plusieurs couches de profondeur et de plusieurs milliers de paramètres par couche. L'application de ces modèles n'est pas un problème pour des ordinateurs de bureau ou personnels qui comportent aujourd'hui plusieurs gigaoctets de mémoire dynamique et des processeurs très performants, mais constitus un challenge pour les applications embarquées.

Dans ce chapitre, nous allons essayer de trouver une méthode qui soit simple et économe en ressources, en vue de l'appliquer sur une plateforme embarquée pour classifier les défauts des roulements à billes à partir des signaux vibratoires.

II.2 Deep learning

Le Deep Learning est un sous-domaine de Machine Learning, qui est, à son tour, un sous-domaine de l'intelligence artificielle (figure II-1), l'objectif principal de l'intelligence artificielle et de fournir un ensemble d'algorithmes et de techniques qui peuvent être utilisés pour résoudre les problèmes que les humains exécutent intuitivement et presque automatiquement [16].

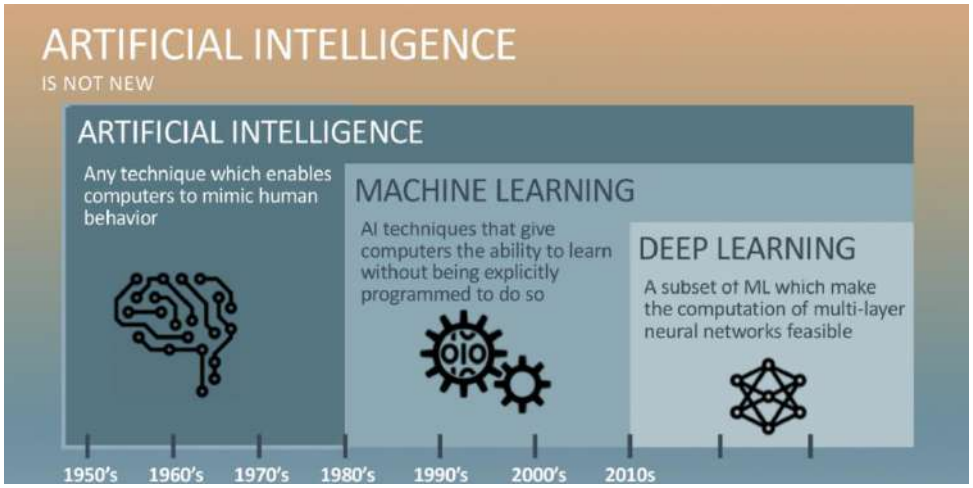


Figure II-1 : Les domaines de l'intelligence artificielle

II.3 Détection des anomalies

La détection des anomalies dans la donnée est une habilité importante à apprendre aux machines. Les hommes peuvent détecter les anomalies présentent dans leurs environnements pour se protéger d'un possible danger ou pour découvrir de nouvelles connaissances. La détection des anomalies et leurs classifications est appliquée aujourd'hui dans le domaine de l'IA pour détecter des fraudes de cartes de crédit, des cyber intrusions ou bien dans la maintenance prédictive des équipements industriels. Cette tâche est une variante de la classification où l'objectif est de dire si l'information présentée à la machine relève d'une situation ordinaire ou bien d'une situation irrégulière et peu même aller plus loin en nous informant sur la nature de cette irrégularité [17].

II.4 Type d'apprentissage

Les algorithmes de Deep Learning se répartissent en deux types principaux, apprentissage supervisé et apprentissage non-supervisé [16].

II.4.1 Apprentissage supervisé

Pour ce type d'apprentissage, l'algorithme de Deep Learning reçoit à la fois un ensemble d'entrées et leurs sorties. Il essaie ensuite d'apprendre des modèles qui peuvent être utilisés pour associer automatiquement le point de données d'entrées sur leur sortie cible correcte (figure II-2).

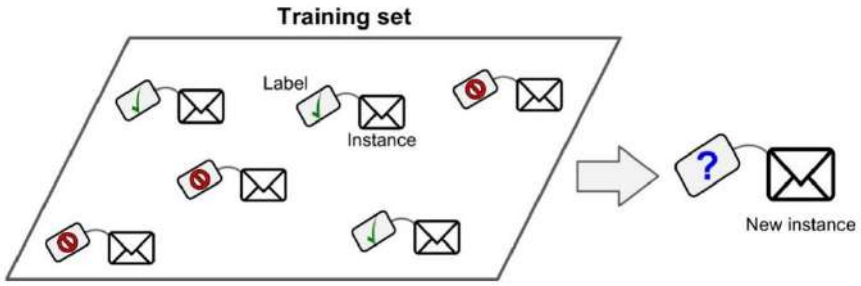


Figure II-2 : Apprentissage supervisé [18], *le modèle est entraîné sur un couple d'entrées/étiquettes et il est ensuite utilisé pour prédire l'étiquette (classe) d'une entrée inconnue*

L'apprentissage supervisé est similaire à celui d'un enseignant qui vous regarde passer un test. Compte tenu de vos connaissances antérieures, vous faites de votre mieux pour choisir la bonne réponse à votre examen ; Cependant, si vous vous trompez, votre enseignant vous guidera vers une supposition meilleure et plus éclairée la prochaine fois.

II.4.2 Apprentissage non-supervisé

Dans un cas d'un apprentissage non-supervisé, les algorithmes de Deep learning tentent de découvrir automatiquement les caractéristiques discriminantes sans aucune indication sur les entrées (figure II-3). Dans ce scénario, l'élève essaie de regrouper des questions et des réponses similaires, même s'il ne sait pas quelle est la bonne réponse et que l'enseignant n'est pas là pour lui fournir la vraie réponse.

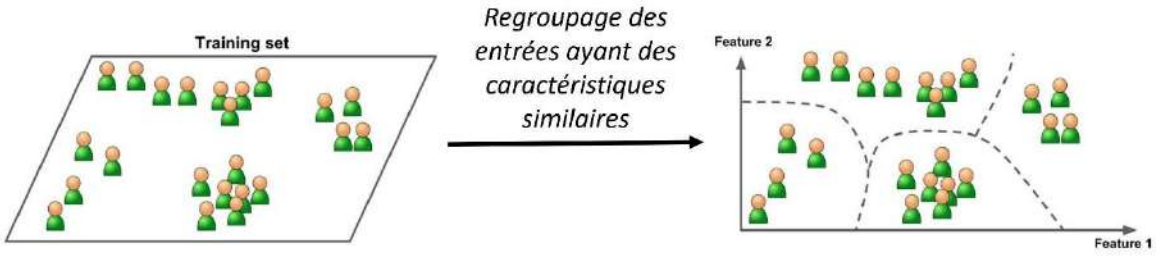


Figure II-3 : Apprentissage non-supervisé [18], *le modèle regroupe tout seul les entrées sans connaitre leurs étiquettes*

II.5 Les réseaux de neurones

II.5.1 Neurone biologique

Comme beaucoup d'invention ont été inspirées par la nature, les réseaux de neurones artificielles ANN inspirés par l'architecture du cerveau humain ou neurone biologique. Notre cerveau est composé d'environ 10 milliards de neurones, chacun connecté à environ 10 000 autres neurones, le neurone est composé de corps cellulaire contenant le noyau, et de nombreuses extensions ramifiées appelées dendrites, plus une très longue extension appelée axone. À l'extrémité de l'axone se trouvent de minuscules structures appelées terminaisons de l'axone, qui sont connectées aux dendrites des autres neurones [18] (figure II-4).

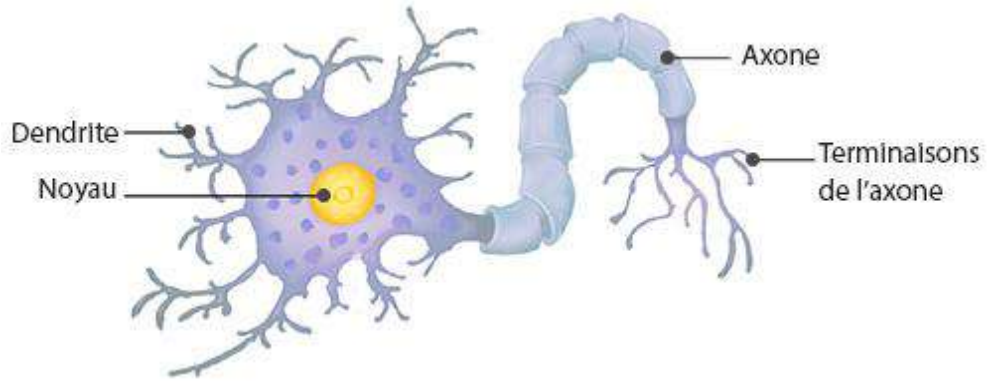


Figure II-4 : Structure d'un neurone biologique [16]

Chaque neurone reçoit des entrées électrochimiques des autres neurones au niveau de leurs dendrites. Si ces entrées électriques sont suffisamment puissantes pour activer le neurone, le neurone est activé et transmet le signal sur son axone, et il le transmet ensuite aux dendrites des autres neurones. Ces neurones attachés peuvent également se déclencher, poursuivant ainsi le processus de transmission du message [16].

II.5.2 Architecture des réseaux de neurones artificiels

Les réseaux de neurones sont des structures complexes constituées de neurones artificiels qui peuvent recevoir plusieurs entrées pour produire une seule sortie. Habituellement, un réseau de neurones se compose d'une couche d'entrée et de sortie avec une ou plusieurs couches cachées à l'intérieur (figure II-5)

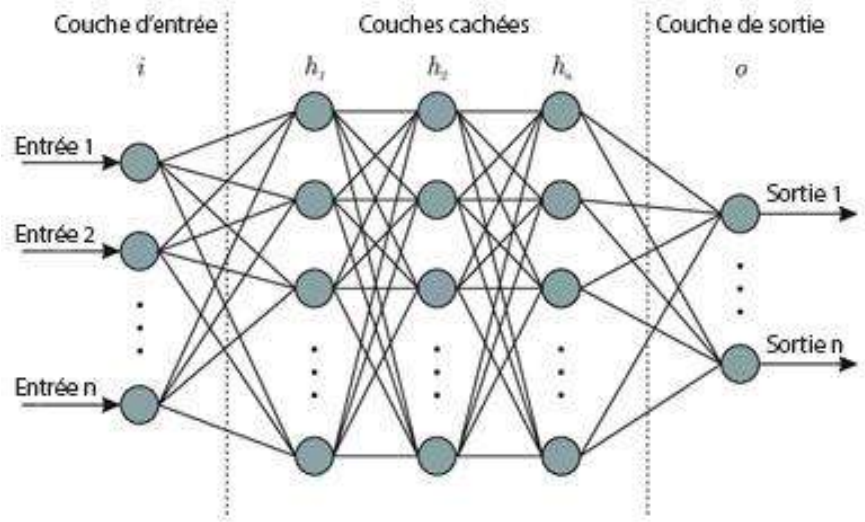


Figure II-5 : Architecture d'un réseau de neurones artificiels

II.5.3 Le perceptron

Le perceptron est une unité élémentaire dans un réseau de neurones artificiels, c'est un neurone artificiel qui effectue une fonction algébrique non linéaire. Le neurone artificiel reçoit une ou plusieurs entrées $\mathbf{x_i}$, et chaque entrée est pondérée séparément par $\mathbf{W_i}$, la somme des entrées est ajoutée au biais de neurone **b**, puis utilisé comme argument d'une fonction d'activation **f** (figure II-6). La prédiction avec un réseau de neurones se fait en appliquant les données comme entrées à la première couche, puis en calculant toutes les couches suivantes jusqu'aux sorties finales. C'est ce qu'on appelle souvent la propagation vers l'avant [16].

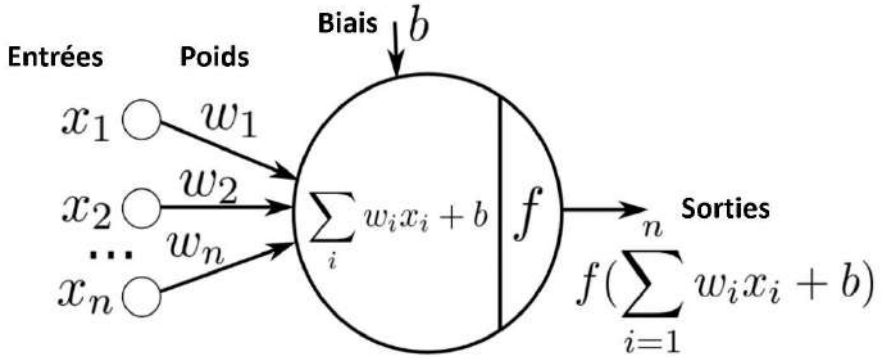


Figure II-6 : Neurone artificiel simple 'Le perceptron' [16]*, qui prend une somme pondérée à l'entrée et détermine si la sortie est active ou non*

II.5.4 Fonction d'activation

Pour pouvoir exprimer des relations non linéaires entre l'entrée et la sortie, des fonctions d'activation sont appliquées. Lorsque la non-linéarité est utilisée, un réseau de neurones devient un approximateur de fonction universel [19]. La sortie du neurone n'a pas de limite, elle varie de $-\infty$ à $+\infty$, le neurone ne connaît donc pas vraiment de limites. Pour dire si un neurone est excité ou non, on introduit une fonction d'activation [16].

II.5.4.1 La fonctions seuil (échelon unité)

C'est la fonction d'activation la plus simple, son équation est donnée par :

$$f(x) = \begin{cases} 1 & si\ x \geq 0 \\ 0 & si\ x < 0 \end{cases} \qquad \text{Éq 13}$$

Comme on peut le voir dans l'équation ci-dessus, si la valeur de la somme pondérée est supérieure à 0 la sortie est 1, sinon la sortie est 0.

II.5.4.2 La fonction sigmoïde

Une fonction d'activation plus couramment utilisée dans l'histoire de la littérature NN est la fonction sigmoïde, qui suit l'équation :

$$f(x) = \frac{1}{1 + e^{-x}} \qquad \text{Éq 14}$$

L'avantage de cette fonction d'activation est quelle est différentiable en tous point. La sortie de cette fonction d'activation sera, en comparaison de la fonction linéaire, toujours dans l'intervalle [0,1].

II.5.4.3 La fonction Tangente Hyperbolique

Une autre fonction d'activation qui est utilisée est la fonction tangente hyperbolique (Tanh), Son équation est donnée par :

$$f(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \qquad \text{Éq 15}$$

La fonction Tangente hyperbolique est centrée sur zéro, donc la sortie de la fonction sera toujours comprise dans l'intervalle [-1, 1].

II.5.4.4 La fonction ReLU

Cette fonction d'activation Rectified Linear Unit (ReLU) est définie comme suit :

$$f(x) = \begin{cases} x & si\ x \geq 0 \\ 0 & si\ x < 0 \end{cases} \qquad \text{Éq 16}$$

ReLU est également appelé fonction rampe, en raison de sa représentation graphique. La sortie de la fonction est nulle pour les entrées négatives, mais augmente linéairement pour les valeurs positives. Empiriquement, la fonction d'activation ReLU a tendance à surpasser à la fois les fonctions sigmoïde et Tanh dans presque toutes les applications [20].

II.5.4.5 La fonction Leaky ReLU

Une variante de ReLU, appelée Leaky ReLU, elle est représentée par la forme algébrique suivante :

$$f(x) = \begin{cases} x & si\ x \geq 0 \\ \alpha x & si\ x < 0 \end{cases} \qquad \text{Éq 17}$$

D'après la formule ci-dessus, et la représentation graphique de cette fonction, nous pouvons voir que la fonction est bien autorisée à prendre une valeur négative, contrairement à ReLU qui met la sortie de la fonction à une valeur nulle.

II.5.4.6 La fonction Unité exponentielle linéaire (ELU)

La fonction Unité Exponentielle Linéaire (ELU) est définie comme suit :

donc :

$$f(x) = \begin{cases} x & si\ x \geq 0 \\ \alpha(e^{x} - 1) & si\ x < 0 \end{cases} \qquad \text{Éq 18}$$

L'hyper-paramètre ELU α contrôle la valeur à laquelle la fonction ELU sature pour les entrées négatives. ELU obtient souvent une précision de classification plus élevée que ReLU.

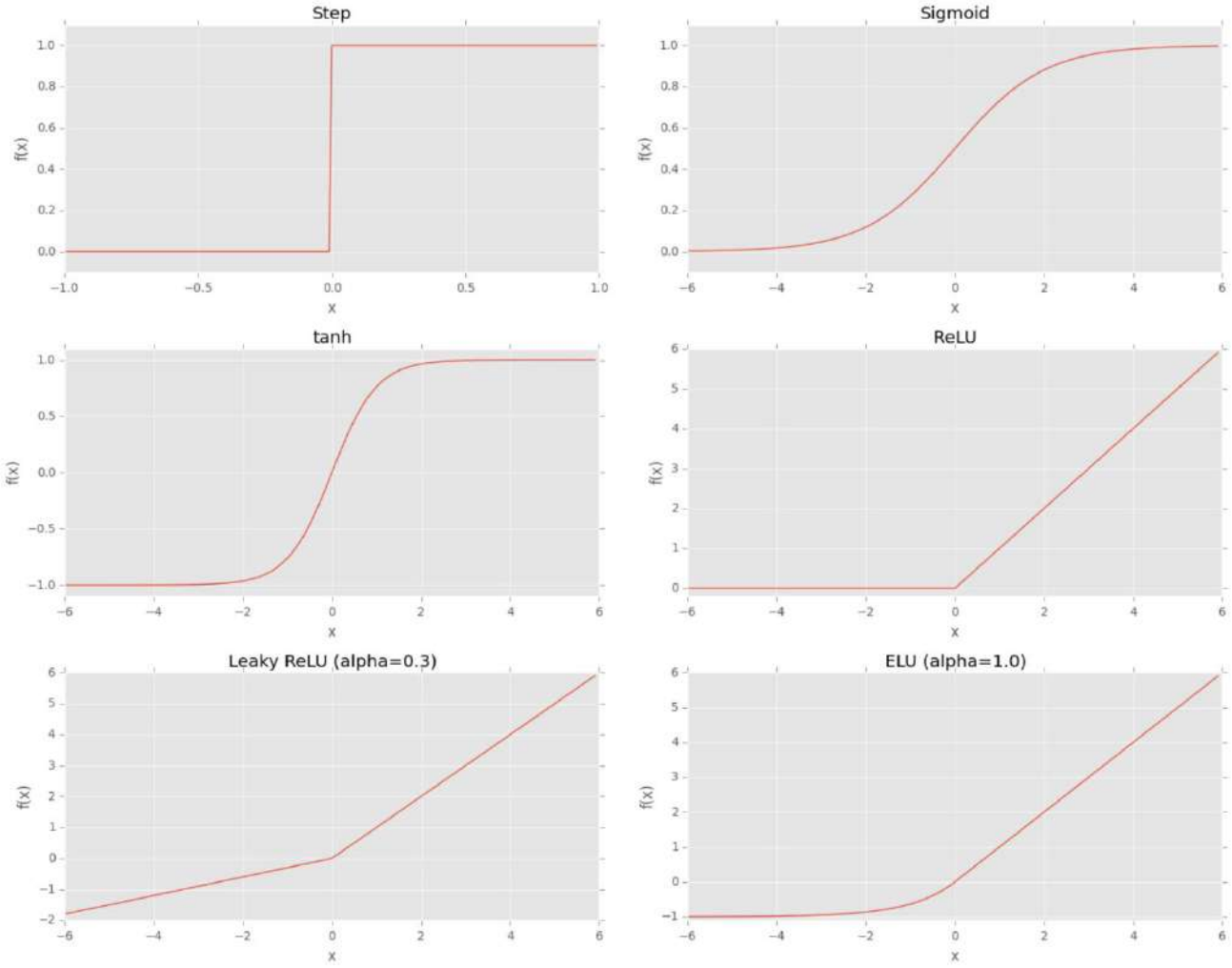


Figure II-7 : Forme des fonctions d'activations les plus utilisées [16]

II.5.5 Types de réseaux de neurones

Plusieurs architectures de réseaux de neurones sont utilisées dans la pratique. Les plus exploitées sont, les réseaux de neurones artificiels (ANN), et les réseaux de neurones à convolution (CNN).

II.5.5.1 Réseaux de neurones artificiels (ANN)

Un seul perceptron (ou neurone) peut être imaginé comme une régression logistique. Le réseau de neurones artificiels, ou ANN, est formé de plusieurs perceptrons dans chaque couche [21] (figure II-5).

Un ANN est également connu sous le nom de réseau de neurones Feed-Forward, car les valeurs des entrées ne circulent que dans le sens direct. Un ANN se compose de 3 types de couches, une couche d'entrée, des couches cachées et une couche de sortie.

II.5.5.2 Réseau de neurones à convolution (CNN)

L'un des réseaux de neurones les plus populaires est le réseau de neurones à convolution (CNN) (figure II-9). Son nom est tiré d'une opération mathématique linéaire entre des matrices appelée convolution. Un réseau CNN a d'excellentes performances dans les résolutions des problèmes d'apprentissage automatique. En particulier les applications traitent des grandes bases de données d'images (ImageNet), la vision artificielle et le traitement du langage naturel [22] .

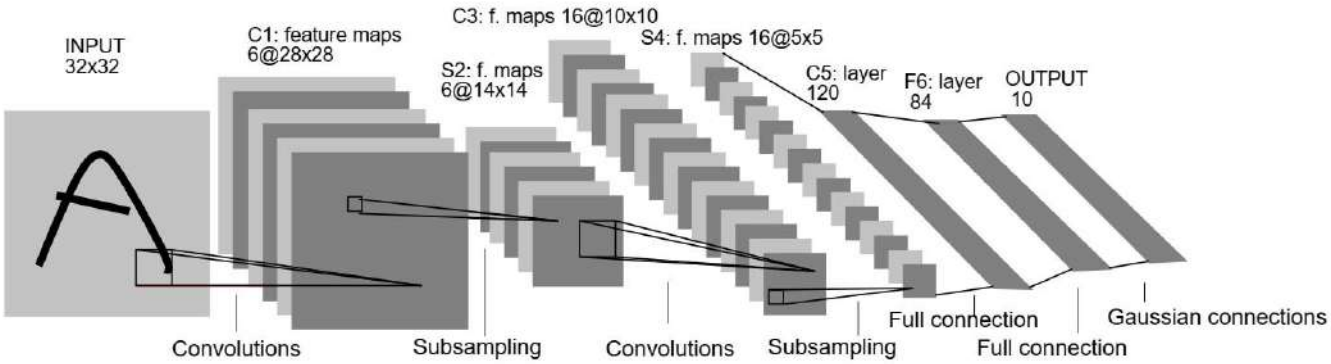


Figure II-8 : Architecture d'un réseau CNN [23]

Il existe de nombreux types de couches utilisées pour constituer des réseaux de neurones convolutifs, mais les plus utilisées sont : convolution, fonction activation, pooling, entièrement connectée, batch normalisation et dropout.

II.5.5.2.1 Couche de convolution (CONV)

La couche CONV est le bloc de construction de base d'un réseau de neurones convolutifs. Les paramètres de la couche CONV consistent en un ensemble de filtres (Noyaux) pouvant être appris, où chaque filtre a une largeur et une hauteur. Dans la plupart des applications, le noyau est une matrice carrée avec des côtés de longueurs impaires.

Pour effectuer une convolution, le filtre de taille fixe balaye la matrice 2D. Pour chaque emplacement, on effectue une multiplication entre les données d'entrée avec les poids du filtre puis on les additionne pour obtenir une valeur unique qui devient la valeur de sortie (figure II-10).

Il convient de mentionner qu'une couche de convolution est spécifiée par les hyper-paramètres suivants :

- La taille du filtre.

- Le nombre de filtre.
- Le pas que les filtres effectuent durant le balayage (Stride)

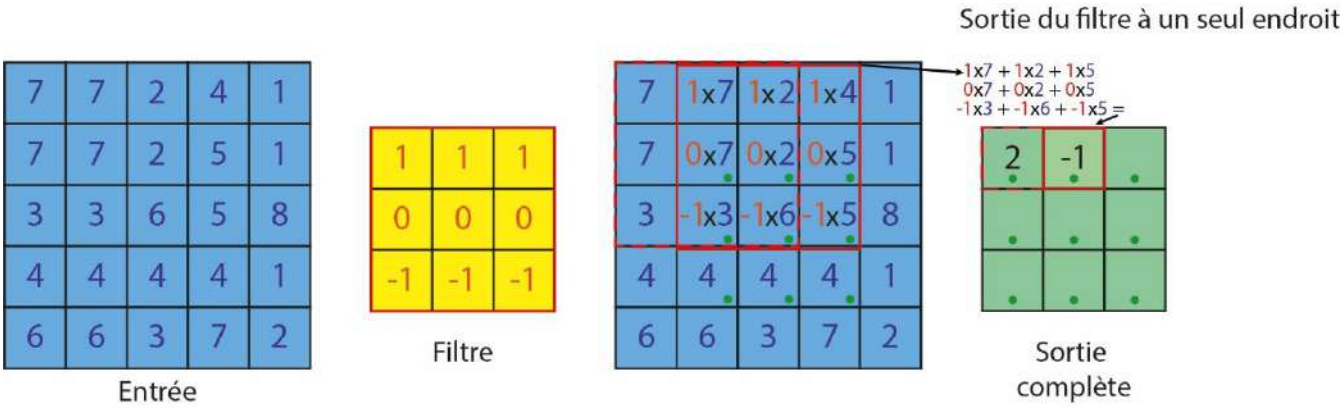


Figure II-9 : Opération de convolution

II.5.5.2.2 Couche de correction (ReLU)

Après chaque couche CONV dans un CNN, une fonction d'activation non linéaire telle que ReLU, ELU ou Leaky ReLU est appliquée [16]. Nous désignons généralement les couches de correction par ReLU dans les architectures de réseaux, car la fonction d'activation ReLU est la plus couramment utilisée. La couche de correction ReLU remplace toutes les valeurs négatives reçues en entrées par des zéros (figure II-11).

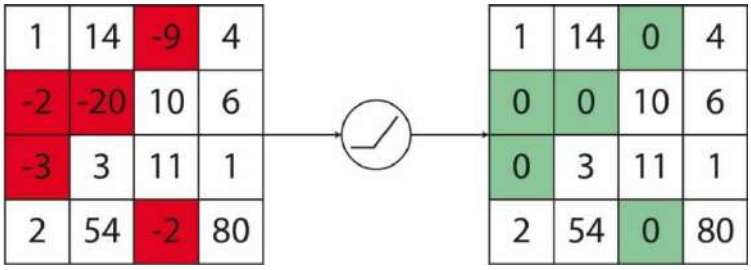


Figure II-10 : Application de la fonction d'activation ReLU

II.5.5.2.3 Couche de Pooling (POOL)

Le rôle principal de la couche POOL est de réduire progressivement la taille (la largeur et la hauteur) des entrées. Cela nous permet de réduire le nombre de paramètres et de calculs dans le réseau. La couche de Pooling nous aide également à contrôler le sur-apprentissage « Overfiting ». La fonction la plus utilisée est la fonction MAX POOLING où l'élément le plus grand est extrait.

En règle générale, nous utiliserons une taille de Pool de 2 × 2, bien que des CNN plus profonds utilisent des images d'entrées plus grandes (> 200 pixels). On choisit une taille de pool de 3 × 3 au début de l'architecture du réseau, puis cette taille est réduite dans les couches les

plus profondes. Nous définissons également le pas de balayage (Stride), qui est généralement de S = 1 ou S = 2. (figure II-12) donne deux exemples d'applications où la fonction MAX POOLING est utilisée une fois avec une taille de Pool 2 × 2 et un stride de S = 1 et l'autre fois avec un stride S = 2 dans l'autre exemple. Dans chaque bloc 2 × 2 le MAX POOLING ne conserve que la valeur supérieure.

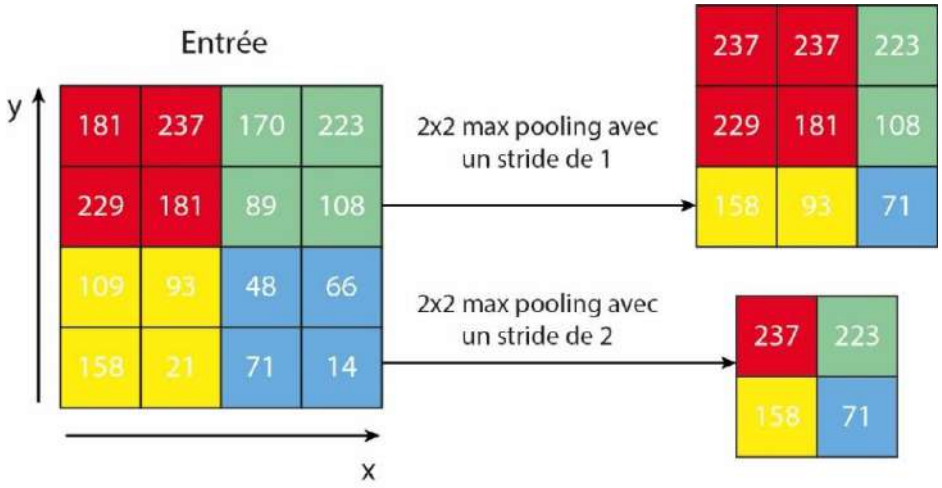


Figure II-11 : Opération de MaxPooling

II.5.5.3 Couche entièrement connectée (FC)

Les neurones des couches FC sont entièrement connectés à toutes les sorties des fonctions d'activation de la couche précédente, comme c'est le cas pour les réseaux de neurones ANN. Les couches FC sont toujours placées à la fin du réseau, c'est-à-dire après la dernière couche de convolution. Il est courant d'utiliser une ou deux couches FC avant d'appliquer le classificateur softmax.

II.5.5.3.1 La fonction d'activation softmax

Softmax est une fonction mathématique qui convertit un vecteur de nombres en un vecteur de probabilités, où les probabilités de chaque valeur sont proportionnelles à l'échelle relative de chaque valeur dans le vecteur [24]. Elle est généralement placée en dernière couche dans le modèle de Deep Learning [21] pour présenter les classes de sorties comme probabilité entre 0 et 1. Mathématiquement, Softmax est défini comme :

$$S(y_i) = \frac{e^{y_i}}{\sum_{j=1}^{n} e^{y_j}} \qquad \text{Éq 19}$$

Où n est le nombre de classes et y_i les éléments de vecteur d'entrée.

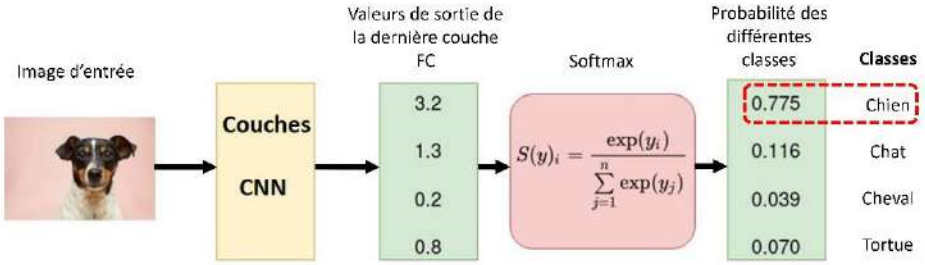


Figure II-12 : Rôle de la fonction d'activation Softmax dans un réseau CNN

II.5.5.4 Normalisation des lots (BN)

Introduites pour la première fois par Ioffe et Szgedy dans leur article de 2015, Batch Normalization (BN) permet d'accélérer l'apprentissage du réseau en réduisant le décalage de covariance interne. Les couches BN, sont utilisées pour normaliser les activations d'un volume d'entrée donné avant de le transmettre à la couche suivante du réseau. Si nous considérons x comme notre mini-lot d'activations, alors nous pouvons calculer le $\hat{x}$ normalisé via l'équation suivante :

$$\hat{x} = \frac{x_i - \mu_\beta}{\sqrt{\sigma_\beta^2 + \varepsilon}} \qquad \text{Éq 20}$$

Pendant l'apprentissage, nous calculons le μ_β et σ_β sur chaque mini-lot β, où :

$$\mu_\beta = \frac{1}{M}\sum_{i=1}^{m} x_i \rightarrow \sigma_\beta^2 = \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_\beta)^2 \qquad \text{Éq 21}$$

Nous mettons la valeur de $\varepsilon = 10^{-7}$, pour éviter de prendre la racine carrée de zéro.

II.5.5.5 Dropout

Le dernier type de couche que nous allons voire est Dropout. Le Dropout est en fait une forme de régularisation qui vise à aider à prévenir le sur-apprentissage en augmentant la précision des tests, au détriment de la précision du réseau au cours de l'apprentissage. Pour chaque mini-lot de notre ensemble d'apprentissage, les couches de Dropout, avec une probabilité p, déconnectent de manière aléatoire les entrées de la couche précédente à la couche suivante dans l'architecture du réseau. (figure II-14) représente exemple d'application d'un Dropout de 50% il est important de remarquer que la moitié des connexions sont supprimées.

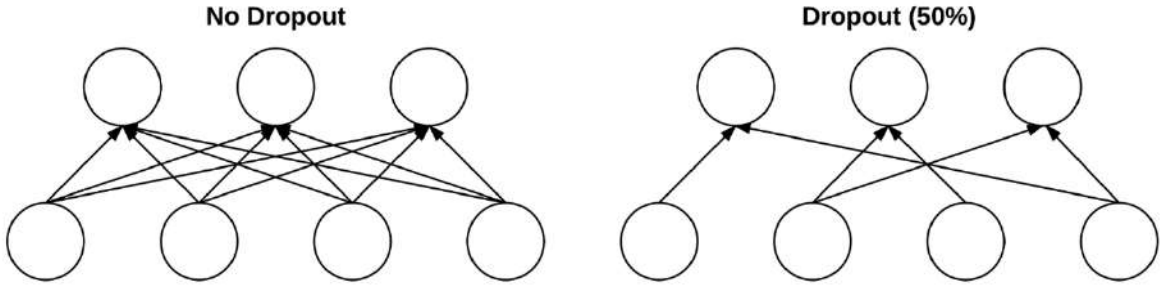


Figure II-13 : Introduction du Dropout dans un réseau ANN, la moitié des neurones sont désactivés

II.6 Extraction des caractéristiques des signaux vibratoires

Étant donné que les réseaux de neurones à convolution (CNN) ont une faible capacité d'extraction de caractéristiques pour les données du format 1D, le réseau CNN est alors combiné à une méthode de traitement du signal pour le diagnostic des défauts [25]. Il existe plusieurs méthodes pour obtenir un format de données 2D à partir de données 1D, parmi les méthodes utilisées on trouve le kurtogramme et le spectrogramme [26].

II.6.1 Kurtogramme

Le kurtogramme (figure II-15) est un outil récent d'analyse spectrale qui permet de détecter dans un signal la présence de structures non-stationnaires, de les localiser en fréquence et, d'une certaine manière, de les caractériser. Le calcul du Kurtogramme dans un calculateur numérique est très coûteux en temps de calcul et difficilement envisageable en pratique. C'est pour cela, Antoni et Randall [27] ont développé un algorithme qui accélérer le calcul du Kurtogramme. Cependant l'implémentation de cet algorithme reste compliquée [27].

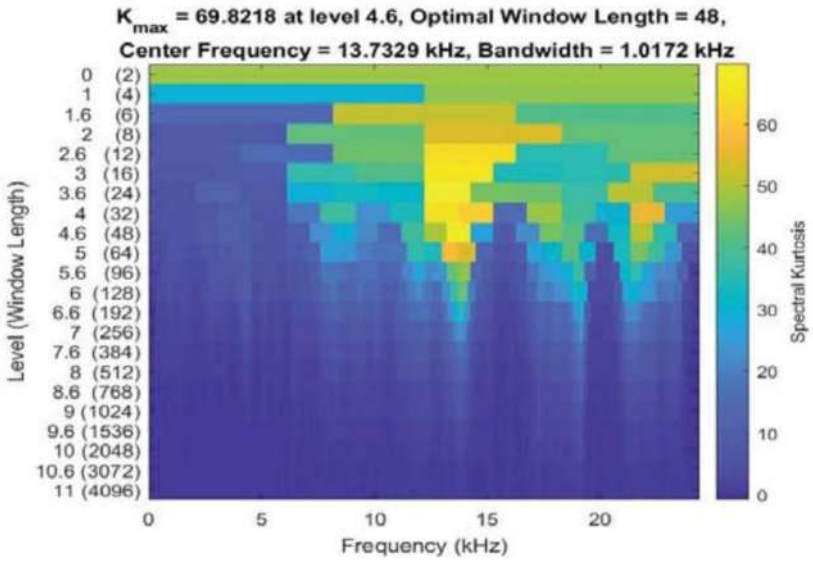


Figure II-14 : Exemple de kurtogramme utilisé pour le filtrage d'un signal vibratoire [28]

II.6.2 Spectrogramme

Un spectrogramme est la représentation visuelle de contenu fréquentiel d'un signal en fonction du temps. Il est représenté par un graphique bidimensionnel dans lequel le temps est indiqué le long de l'axe horizontal, la fréquence est indiquée le long de l'axe vertical et l'amplitude des composantes de fréquence à un moment donné est visuellement indiquée par l'intensité de la couleur de ce point dans le graphique. Les faibles amplitudes sont généralement représentées par des couleurs bleus foncés et les amplitudes plus élevées sont représentées par des couleurs plus vives comme la couleur rouge. Le calcul du spectrogramme est effectué en divisant le signal en segments puis en appliquant une transformée de Fourier au signal vibratoire, ce qui permet de former une représentation temps-fréquence (figure II-16).

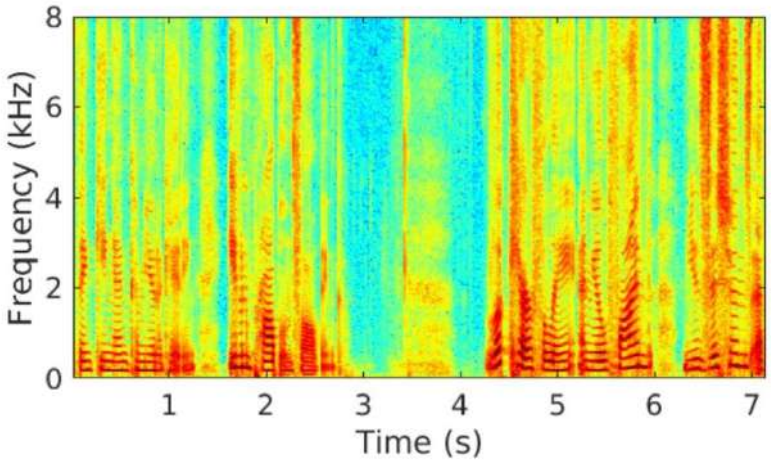


Figure II-15 : Exemple de spectrogramme [29]

II.6.3 MFCC

On a constaté que les méthodes proposées dans la littérature ne conviennent pas pour l'implémentation sur les plateformes embarquées. Plusieurs autres méthodes ont été alors proposées dans l'extraction des caractéristiques, surtout dans le domaine des signaux vocaux et sonores. On a voulu tester l'efficacité de l'une de ces méthodes, le prétraitement MFCC.

Le traitement MFCC (**M**el **F**requency **C**epstral **C**oefficients) est une technique utilisée pour extraire les caractéristiques d'un signal. Elle est utilisée dans plusieurs domaines notamment dans la reconnaissance automatique de la parole. Leur force réside dans leur capacité à effectuer des types de filtrage similaires qui se trouvent dans le système auditif humain et leur capacité à compresser cette information [30]. Les MFCCs contiennent à la fois des informations dans le domaine temporel et fréquentiel du signal, ce qui les rend plus utiles

pour extraire les caractéristiques des signaux vibratoires. Ce traitement est divisé en plusieurs étapes successives.

II.6.3.1 Segmentation

Les méthodes d'analyse du contenu fréquentiel opèrent généralement sur des signaux stationnaires, alors que le signal vibratoire que l'on veut traiter change perpétuellement au cours du temps. La solution est donc d'effectuer l'analyse sur des trames successives de durées relativement courtes où le signal peut être considéré comme quasi-stationnaire. Les trames peuvent aussi être chevauchées pour ne pas perdre l'information contenue aux bordures. Le nombre de trames obtenu par la segmentation du signal $x[n]$ de longueur N en trames de longueur L et d'espacement entre les trames H est de :

$$M = \frac{N - L}{H} + 1 \qquad \text{Éq 22}$$

II.6.3.2 Fenêtrage

Le fenêtrage consiste à multiplier les différentes trames obtenues au cours de la segmentation par une fonction de mise à l'échelle. Le but de cette fonction est de lisser les effets de discontinuité introduite au cours de l'opération de segmentation et qui peuvent faire apparaître des composantes hautes fréquences. L'application des fonctions de fenêtrage aux échantillons réduit donc les modifications apportées au segment pour atténuer les signaux près des bords de la trame qui peuvent avoir des effets sévères après l'application de la DFT. Dans la plupart des applications, la fenêtre la plus utilisée est la fenêtre de Hamming (figure II-17) qui est définie comme suit :

$$w[n] = 0.54 - 0.46\cos(2\pi\frac{l}{N-1}) \qquad \text{Éq 23}$$

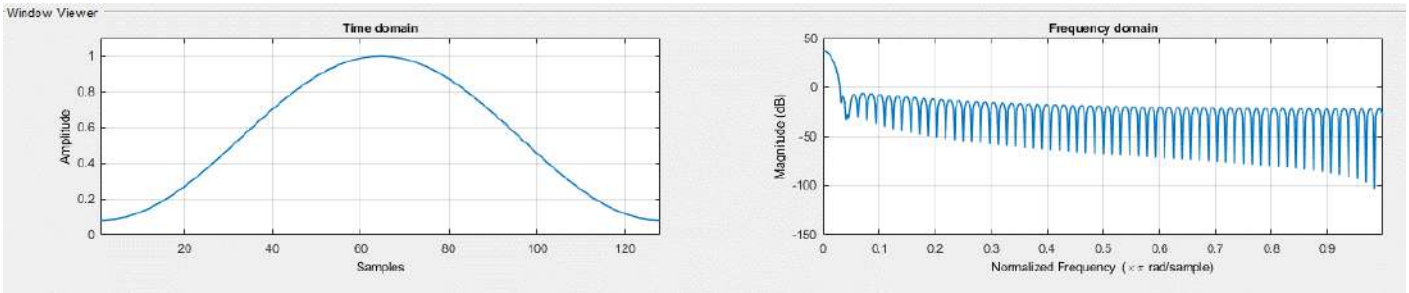


Figure II-16 : Représentation temporelle et fréquentielle de la fenêtre de Hamming

II.6.3.3 Transformée de Fourier

La transformée de Fourier TF permet de représenter un signal temporel à travers ses composantes fréquentielles. La transformée de Fourier dans les cas des signaux continues est définie par :

$$X(f) = \int_{-\infty}^{+\infty} x(t)e^{-j2\pi ft}dt \qquad \text{Éq 24}$$

Où $X(f)$ est le spectre du signal temporel $x(t)$.

La transformée de Fourier Discrète (TFD), est utilisée quand il s'agit de calculer la transformée de Fourier d'une fonction à l'aide d'un calculateur numérique. En effet les machines ne peuvent traiter que des nombres finis car elles sont limitées par la taille de leurs mémoires. Pour aller de la TF à la TFD il faut remplacer d'une part le signal $x(t)$ par des nombres $x(nT)$ qui représentent un échantillonnage de ce signal et d'autre part en limitant l'ensemble des nombres sur lesquels portent les calculs à une valeur finie N [31].

$$X[k] = \sum_{n=0}^{N-1} x(n)e^{-j2\pi f\frac{nk}{N}} \; avec\; k = 0, 1, 2, \dots\; N-1 \qquad \text{Éq 25}$$

Le coût de calcul de la TFD est donc de N^2 multiplications complexes, il a donc une complexité de $O(N^2)$. Le nombre de multiplications augmente alors de façon exponentielle avec l'augmentation de la taille du signal à traiter. Ce nombre peut être grandement réduit en utilisant la transformée de Fourier rapide (FFT).

La FFT est simplement un algorithme permettant de réduire le nombre d'opérations, en particulier le nombre de multiplications. Pour calculer la TFD, il existe différents algorithmes de transformée de Fourier rapide. Le plus connu et le plus utilisé est celui de Cooley-Tukey (appelé aussi « à entrelacement temporel »). Le nombre des multiplications complexes réduit par cette algorithme en $N + N\log_2 N$, ce qui réduit la compléxité à $O(N\log_2 N)$.

Le (tableau II-1) montre l'amélioration des temps de calcul d'une DFT par utilisation d'un algorithme de FFT.

Tableau II-1 : Comparaison entre la DFT et la FFT en terme de complexité de calcul

Nombre d'échantillons	Nombre de multiplications DFT	Nombre de multiplications FFT	Rapport du nombre de multiplications
8	64	24	2.5
128	16384	896	18.95
512	262144	4608	56.88
1024	1048578	10240	102.4
2048	4194304	22528	186.18

II.6.3.4 Puissance spectrale

Le résultat du calcul de la FFT d'un signal est généralement complexe (module et phase), on introduit alors le module élevé au carré pour extraire uniquement l'amplitude du spectre :

$$S[k] = |X[k]|^2 \qquad \text{Éq 26}$$

II.6.3.5 Banc de filtre de Mel

Un banc de filtres Mel est un ensemble de filtres passe-bande qui émulent le système auditif humain. Plutôt que de suivre une échelle linéaire, ces filtres triangulaires agissent de manière logarithmique pour les fréquences hautes, supérieures à 1000Hz et de manière linéaire pour les basses fréquences (figure II-18). Généralement, le nombre de la banque de filtres choisi est de 40 filtres.

Le passage entre Mel-fréquence et fréquence-Mel s'effectue par les formules suivantes :

$$m = 2595 log_{10}\left(1 + \frac{f}{700}\right), f = 700\left(10^{\frac{m}{2595}} - 1\right) \qquad \text{Éq 27}$$

Chacun des filtres produit une sortie qui est la somme pondérée des fréquences spectrales qui correspondent à chaque filtre. Ces valeurs permettent d'associer les fréquences d'entrée dans l'échelle Mel [32], mais aussi de réduire grandement la dimension du signal.

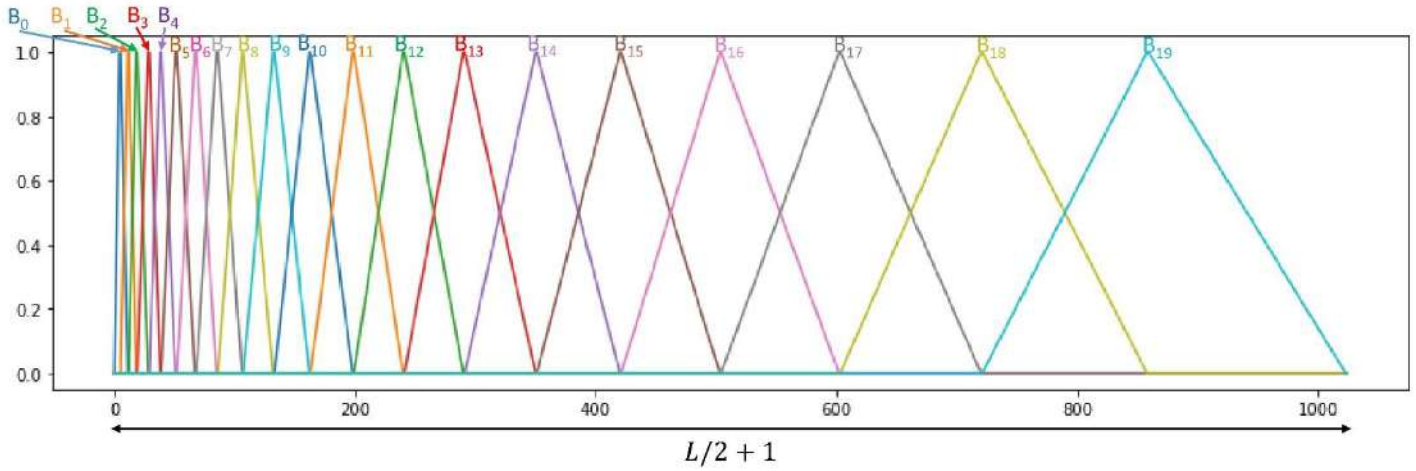


Figure II-17 : Banc de filtres de Mel (exemple de 20 filtres)

II.6.3.6 Conversion en décibel :

L'échelle du décibel qui est obtenue par l'introduction du logarithme permet d'une mise à l'échelle des données numériques sur une très large plage de valeurs de manière compacte, il produit une image des caractéristiques plus claires que l'image.

II.6.3.7 Transformée en cosinus discrète (DCT)

La fonction DCT est similaire à une transformée de Fourier mais n'utilise que des nombres réels. Elle compresse les données d'entrées dans un ensemble de coefficients en cosinus qui décrivent les composantes cepstrale de la fonction. La sortie de cette conversion est appelée MFCC. La DCT possède plusieurs variantes, la plus utilisées est la DCT de type 2 :

$$C_k = \sum_{n=0}^{N-1} x_n \cos\left[\frac{\pi}{N}\left(n + \frac{1}{2}\right)k\right] \qquad \text{Éq 28}$$

II.6.3.8 Déroulement du calcul MFCC

Le calcul des MFCC commence par la division du signal vibratoire en segments de tailles bien déterminés. Ces segments sont traités individuellement sur les opérations mathématiques mentionnées précédemment. Les coefficients MFCC résultants sont copiés colonne par colonne, jusqu'à construire une matrice 2D représentée sous forme d'une image.

L'ensemble des étapes de calcule des MFCC peuvent être résumées sur la (figure II-19).

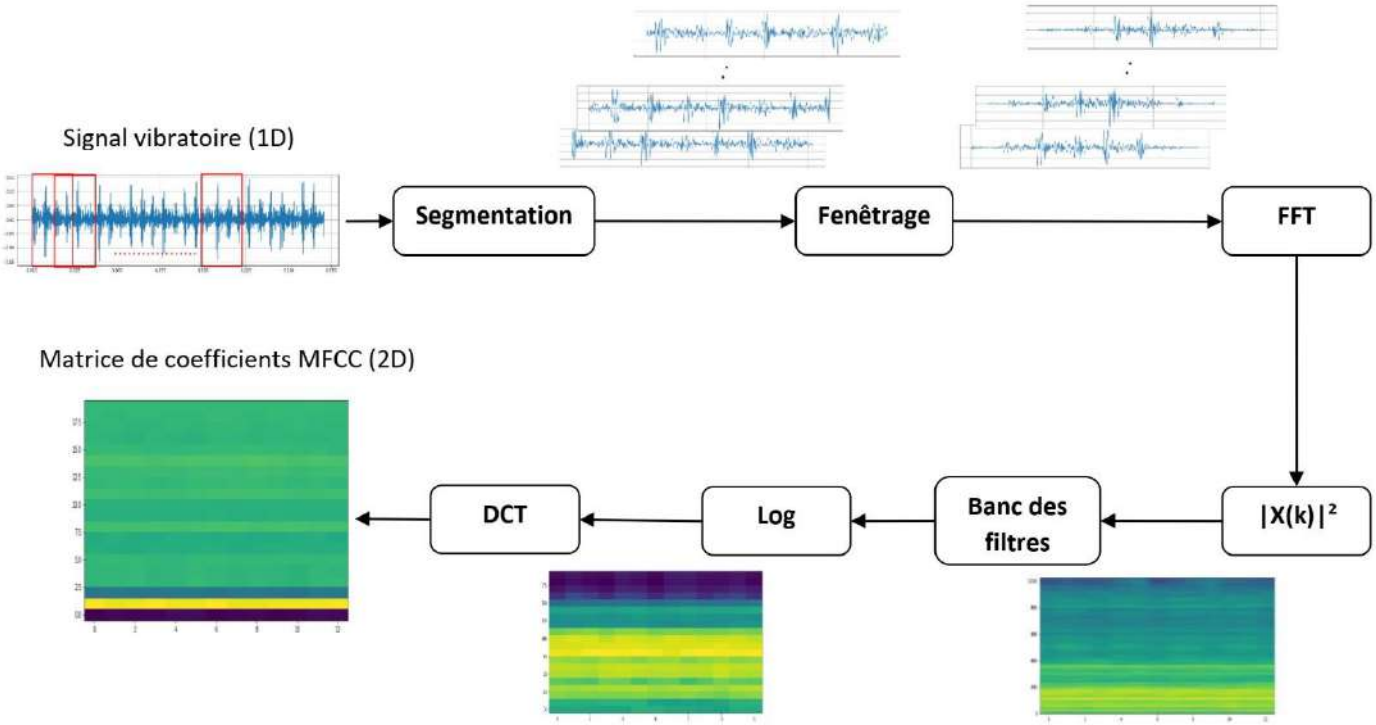


Figure II-18 : Étapes de calcul de la matrice MFCC

II.7 Application de la méthode MFCC sur les signaux vibratoires

II.7.1 Base de données choisie

Pour tester notre méthode de classification des défauts de roulement, nous avons choisi d'utiliser des données expérimentales fournies par le centre de données sur les roulements de l'université de case western (Cleveland, Ohio, États-Unis d'Amérique) [33].

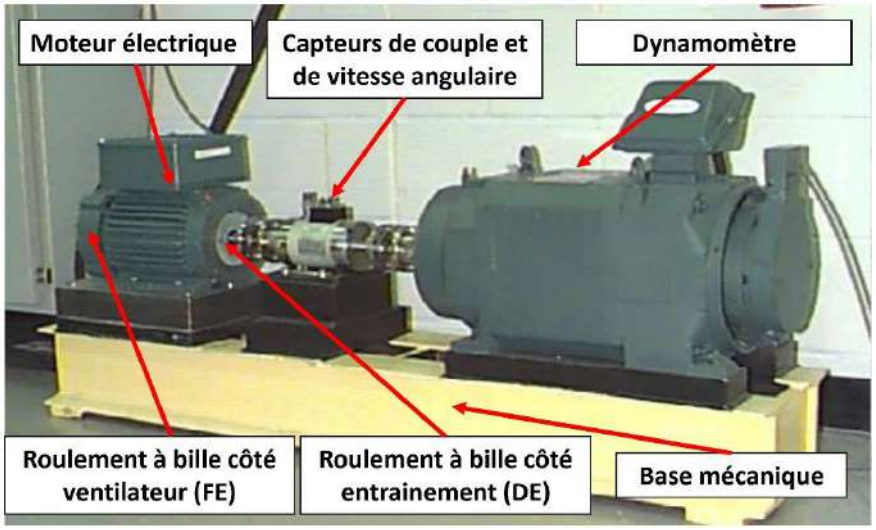


Figure II-19 : Banc d'essai des défauts des roulements

La maquette sur laquelle les tests ont été effectués est représentée sur la (figure II-20). Elle consiste en un moteur électrique de 2 chevaux (HP) Reliance Electric Motor qui entraine un axe relié à un capteur de couple et un capteur de vitesse angulaire. Un couple est appliqué au moteur via un dynamomètre qui joue le rôle d'une charge variable. Pour effectuer les tests, des défauts ont été implantés aux différentes parties des deux roulements, côté entrainement et côté ventilateur, par la méthode d'électroérosion (EDM), le diamètre des défauts introduits varie entre 0.18mm et 0.71mm. La maquette comporte aussi trois accéléromètres qui servent à capturer les signaux vibratoires du banc de test, et sont positionnés au niveau du roulement côté d'entrainement, du roulement côté ventilateur et de la base du banc de test.

Les signaux récoltés sur cette maquette ont été classées sur la base de cinq critères :

- Le roulement sur lequel le défaut a été implanté (côté entrainement ou ventilateur).
- La fréquence d'échantillonnage (48KHz et 12KHz).
- L'emplacement du défaut sur le roulement (aucun défaut, défaut sur la bague extérieure, défaut sur la bague intérieure, défaut sur un élément roulant) et ceci pour le roulement côté entrainement et côté ventilateur.
- Vitesse d'entrainement de l'axe qui dépend de la charge appliquée au moteur par le dynamomètre.
- Diamètre du défaut (0.18mm à 0.71mm avec un pas de 0.18mm).

Les signaux sont délivrés au format Matlab (fichiers .mat) et la plupart sont structurés de la façon suivante :

- **DE** (Drive End) : signal récolté à partir de l'accéléromètre côté entrainement
- **FE** (Fan End) : signal récolté à partir de l'accéléromètre côté ventilateur
- **BA** (Base) : signal récolté depuis la base de la maquette
- **RPM** (Révolutions Per Second) : vitesse de rotation du moteur durant le cas de test.

Tableau II-2 : Fréquences des défauts de roulements

Origine du défaut	**Bague interne**	**Bague externe**	**Élément roulant**
Fréquence du défaut[1] (Hz)	157.91	104.53	137.46

[1] Les fréquences ont été calculées en utilisant une vitesse de rotation de l'axe du moteur de 1750 révolution par minutes, ce qui correspond à une fréquence de rotation de 29.16 Hz

Pour effectuer nos tests, nous avons travaillé sur les signaux vibratoires échantillonnés à 48 KHz pour les accéléromètres côté entrainement DE. Le modèle de roulement utilisé est le SKF 6205, les fréquences des défauts de ce type de roulements sont résumées sur le (tableau II-2).

II.7.2 Traitement MFCC sur les signaux vibratoires

Chacun des signaux vibratoires des défauts de roulements de la base de données choisie est enregistré pendant une période de 10s avec une fréquence d'échantillonnage de 48KHz, ce qui nous donne environ 4800000 échantillons dans chaque structure des fichiers Matlab, qui est un nombre d'échantillons très imposant. Pour utiliser notre prétraitement MFCC et transformer les signaux vibratoires en images, nous avons dû diviser les signaux en trames de 8192 points.

Les trames résultantes sont ensuite converties en images en utilisant la méthode MFCC. Les paramètres choisis pour le prétraitement sont :

- taille des trames N = 8192 points;
- taille des segments L = 2048 points;
- taille des sauts de chevauchements entre chaque segment H = 512 points ;
- nombre de point choisi pour effectuer la FFT L = 2048 points ;
- nombre de filtres de Mel n_mel = 20 ;
- nombre de points pour effectuer la DCT n_dct = 20 ;

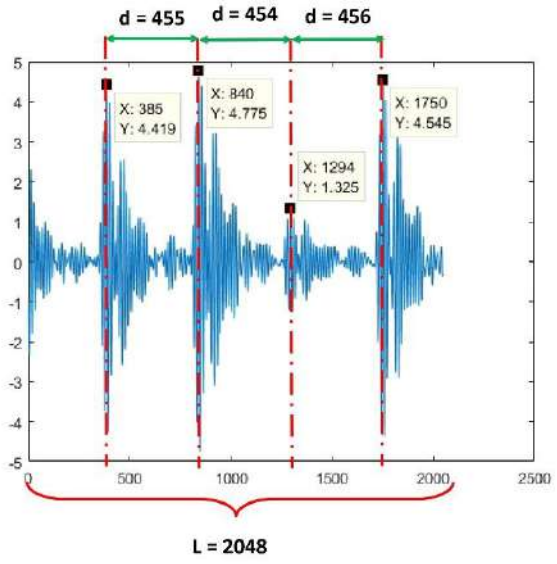


Figure II-20 : La longueur d'un segment permet de contenir au moins 3 impulsions

Les valeurs de N, L et H ont été choisie pour avoir exactement 13 colonnes dans l'image finale. Aussi le nombre de point dans un segment L qui est de 2048 point correspond au regroupement de trois impulsions de défaut au minimum dans un seul segment (figure II-21)

La figure II-22 permet de voir le résultat du traitement MFCC effectué sur la première trame du signal vibratoire d'un défaut issu de la bague interne en utilisant les paramètres cités ultérieurement, l'image ou la matrice résultante à une taille fixe de 20 lignes, et 13 colonnes.

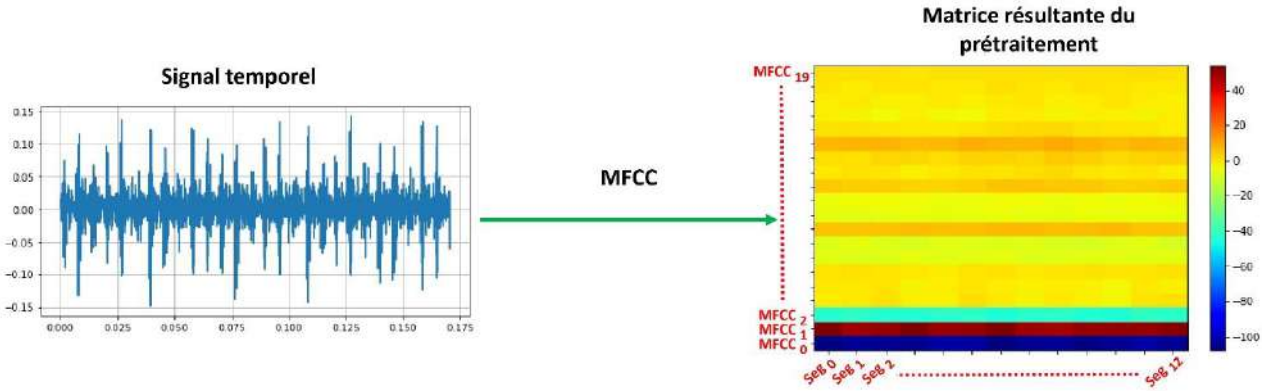


Figure II-21 : Conversion du signal vibratoire (8192 points) en image mfcc(20 lignes, 13 colonnes)

II.7.3 Création d'un dataset MFCC

Après avoir converti 2505 signaux vibratoires du dataset en image MFCC, il en résulte un nouveau dataset intermédiaire composé des images MFCC et des différentes classes de défauts associés (figure II-23). Le reste des signaux vibratoires (232) n'ont pas été converti et servirons pour tester l'efficacité de l'implémentation finale, prétraitement MFCC et classificateur CNN sur ordinateurs.

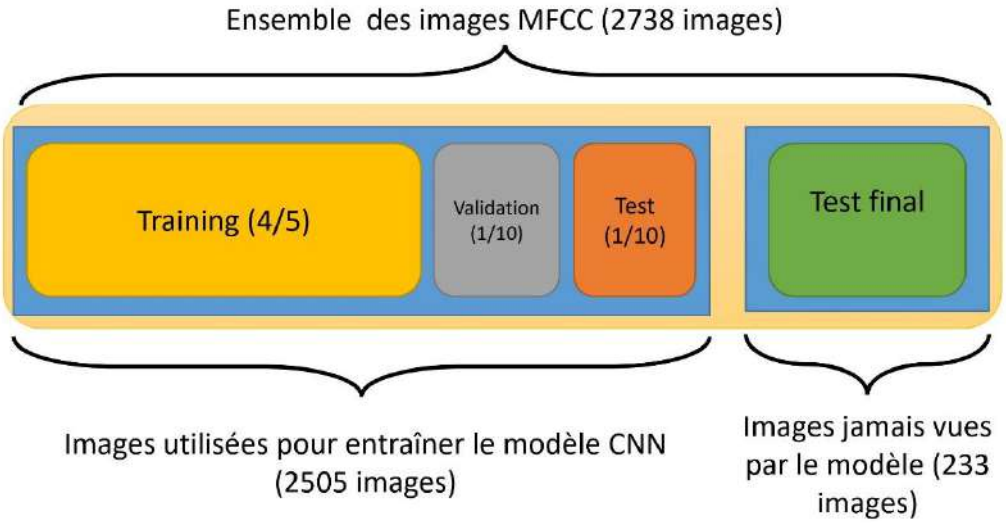


Figure II-22 : Partage des images du dataset des images MFCC

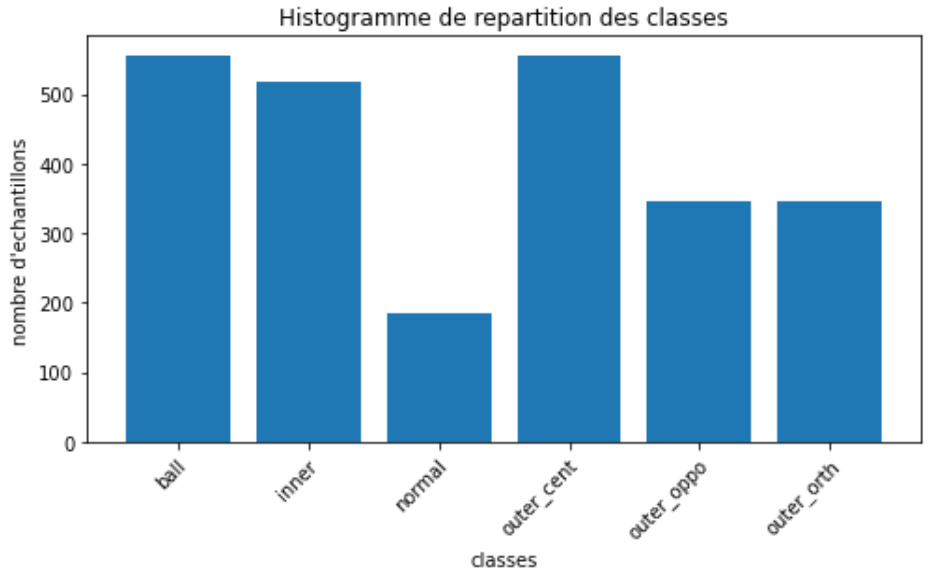


Figure II-23 : Répartition des images utilisées pour entraîner le modèle CNN dans les différentes classes

II.7.4 Le modèle CNN

Pour bénéficier des avantages que nous offre l'utilisation du Deep Learning, nous avons décidé de créer notre propre modèle de réseau CNN. Pour ce faire nous nous sommes tournés vers le langage de programmation Python, et les diverses librairies de Deep Learning qu'il propose.

II.7.4.1 Python

Python est un langage de programmation haut niveau crée par Guido van Rossum en 1989. C'est un langage de programmation multiplateformes haut niveau qui est gratuit et simple à prendre en main. Python est aussi un langage interprété, c'est-à-dire qu'il n'a pas besoin d'être compilé pour être exécuté, contrairement à d'autres langages comme le C ou le C++, ce qui le rends adapté pour le prototypage [34].

La Python Software Fondation est l'association qui organise le développement de Python et anime la communauté de développeurs et d'utilisateurs [34]. La dernière version de Python est la version 3.9.6 mais celle que nous avons utilisée est la version 3.7.9.

Pour installer Python et les différentes librairies que l'on a utilisé au cours de la création du réseau CNN, nous avons utilisé Anaconda pour gérer les environnements virtuels et les librairies associées.

II.7.4.2 Tensorflow et Keras

TensorFlow est une plate-forme Open Source de bout en bout dédiée au Machine Learning. Elle propose un écosystème complet et flexible d'outils, de bibliothèques et de ressources communautaires permettant aux chercheurs d'avancer dans le domaine du machine learning, et aux développeurs de créer et de déployer facilement des applications qui exploitent cette technologie [35]. Tensorflow a aussi pour avantage de pouvoir utiliser une carte graphique (GPU) de la marque Nvidia pour accélérer les phases d'entrainement et de prédictions des modèles des Deep Learning.

Tensoflow apporte un très grand nombre de fonctions pour créer des modèles de Deep Learning, mais leur utilisation reste compliquée, c'est pour cela que Tensorflow est généralement allié à d'autres cadres de travail comme Keras pour faciliter le processus de création. Keras est une API de réseaux de neurones de haut niveau, écrite en Python et interfaçable avec TensorFlow, CNTK et Theano. Elle a été développée avec l'objectif de permettre des expérimentations rapides. Être capable d'aller de l'idée au résultat avec le plus faible délai possible étant la clef d'une recherche efficace [36].

II.7.4.3 Le modèle CNN proposé

Pour classifier les défauts des roulements à billes, nous avons créé un modèle CNN qui comporte deux couches de convolution et deux couches entièrement connecté. Ce modèle est d'une taille très réduite et ne comporte pas beaucoup de paramètres par couches cachée. On justifie le choix de ce modèle par la taille des images en entrées qui sont très petites (20 lignes et 13 colonnes), mais aussi par la volonté d'avoir un modèle qui ne soit pas très volumineux en vue de l'implémenté sur une plateforme embarquée.

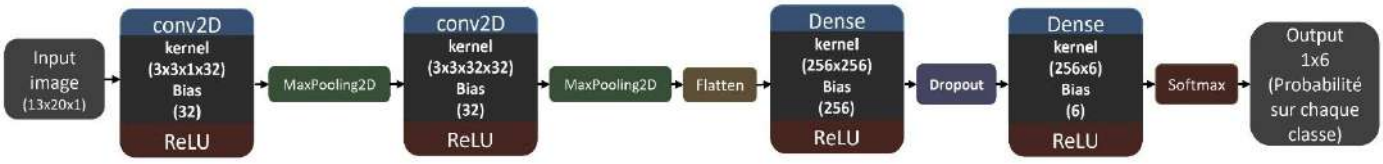


Figure II-24 : modèle CNN proposé

II.7.5 Résultats de classification

Le dataset intermédiaire a été divisé en échantillons d'apprentissage (80%), de tests (10%) et de validation (10%) (figure II-23). Les échantillons d'apprentissage ont été ensuite utilisés pour entrainer le modèle CNN choisi précédemment. Les résultats peuvent être visualisés sur

les (figure II-26). On a relevé une précision de 99.13% pour les signaux de test jamais vues par le modèle.

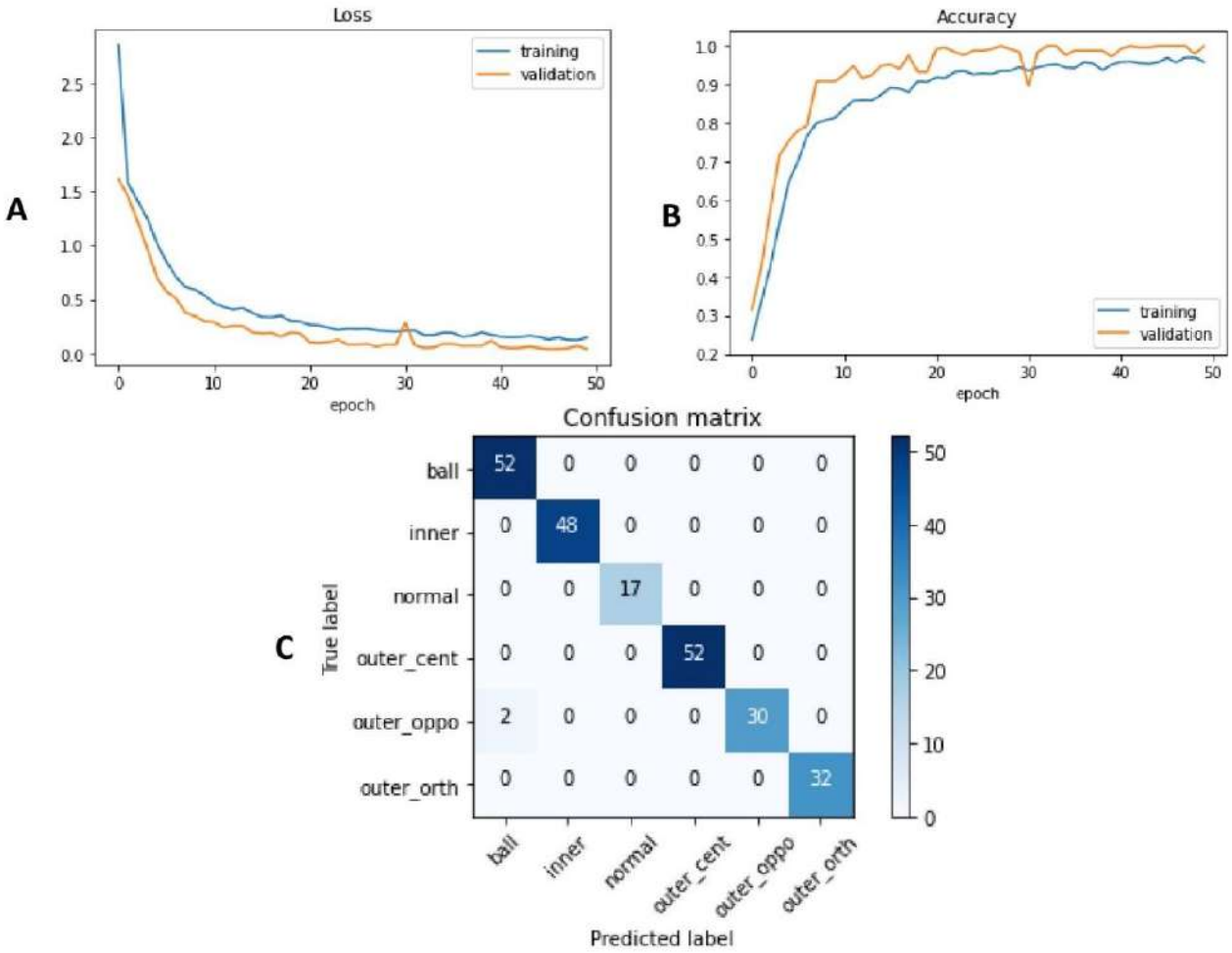


Figure II-25 : Résultat du modèle CNN proposé

A : graphe de l'évolution de l'erreur, B : graphe de l'évolution de la précision

C : résultat de classification des signaux de tests (10% du dataset)

La figure II-27 présente un exemple de résultat d'exécution du prétraitement, du résultat du classificateurs CNN et du temps d'exécution sur l'ordinateur.

```
MFCC compute time on pc : 0.28751 ms
CNN forward propagation time on pc : 0.17953 ms

Predicted classe : inner with 100.00% certenty
True classe is : inner
```

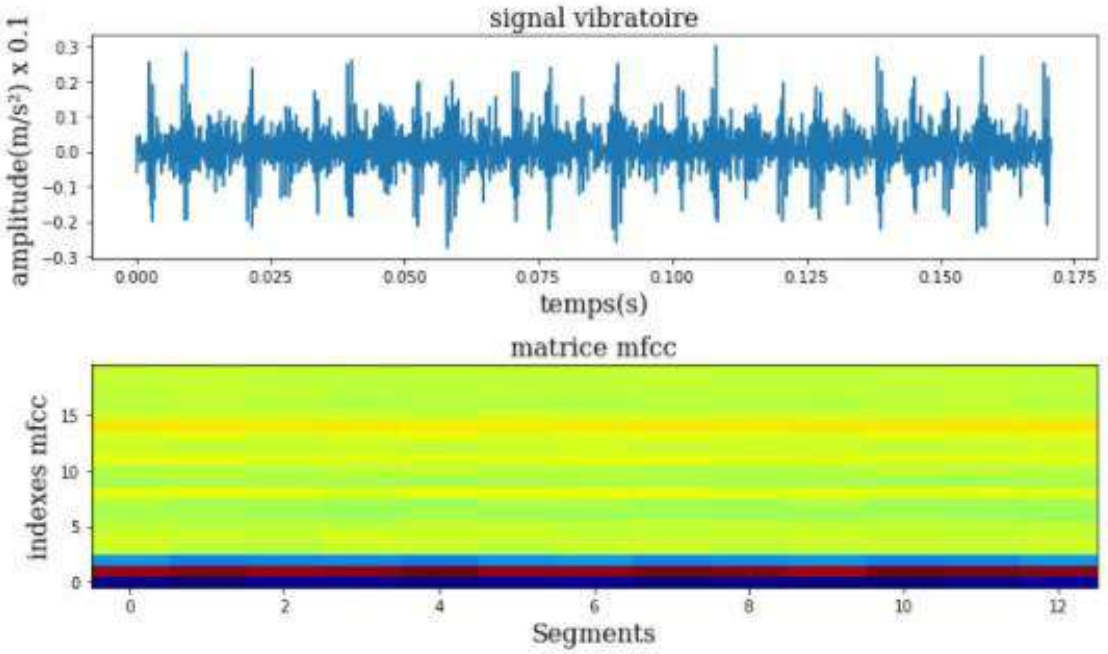


Figure II-26 : Résultat de prédiction sur un signal vibratoire d'un défaut de bague interne et temps d'exécution du prétraitement et de la prédiction sur PC

II.8 Conclusion

Dans ce chapitre, une méthode d'extraction des caractéristiques des signaux vibratoires a été développée pour transformer les données vibratoires unidimensionnelles en image ou matrice à 2 dimensions. On a ensuite exploité cette méthode pour servir d'entrée à un réseau CNN pour l'entraîner à classifier les différentes localisations des défauts des roulements à billes. Les tests sur ordinateur ont montré des résultats concluants.

Chapitre III: Implémentation, tests et résultats

III.1 Introduction

Le chapitre précédant nous a permis de vérifier la validité de la méthode d'extraction des caractéristiques des signaux vibratoires choisie sur ordinateur. Mais on a remarqué que la plupart des applications industrielles utilisent maintenant des systèmes plus performants dans leurs tâches, moins encombrants en espace et qui consomment beaucoup moins d'énergie. Ces systèmes spécialisés sont connus sous le nom de systèmes embarqués.

Un système embarqué est composé typiquement de matériel exécutant du logiciel, il est généralement constitué d'un processeur, de mémoires et éventuellement de circuits dédiés à des applications spécifiques (ASIC) [37]. Le développement de la technologie micro-électronique et de miniaturisation des composants a fait que les systèmes embarqués modernes sont implémentés autours d'un seul circuit intégré, on parle alors d'un système sur circuit (SoC).

Ce chapitre a pour but d'implémenter le système de détection des défauts des roulements sur une plateforme embarquée en utilisant la carte FPGA Pynq-Z2 et la clé Intel NCS2.

III.2 Système implémenté

III.2.1 Schéma synoptique

La figure III-1 représente le schéma synoptique du système que nous avons conçu :

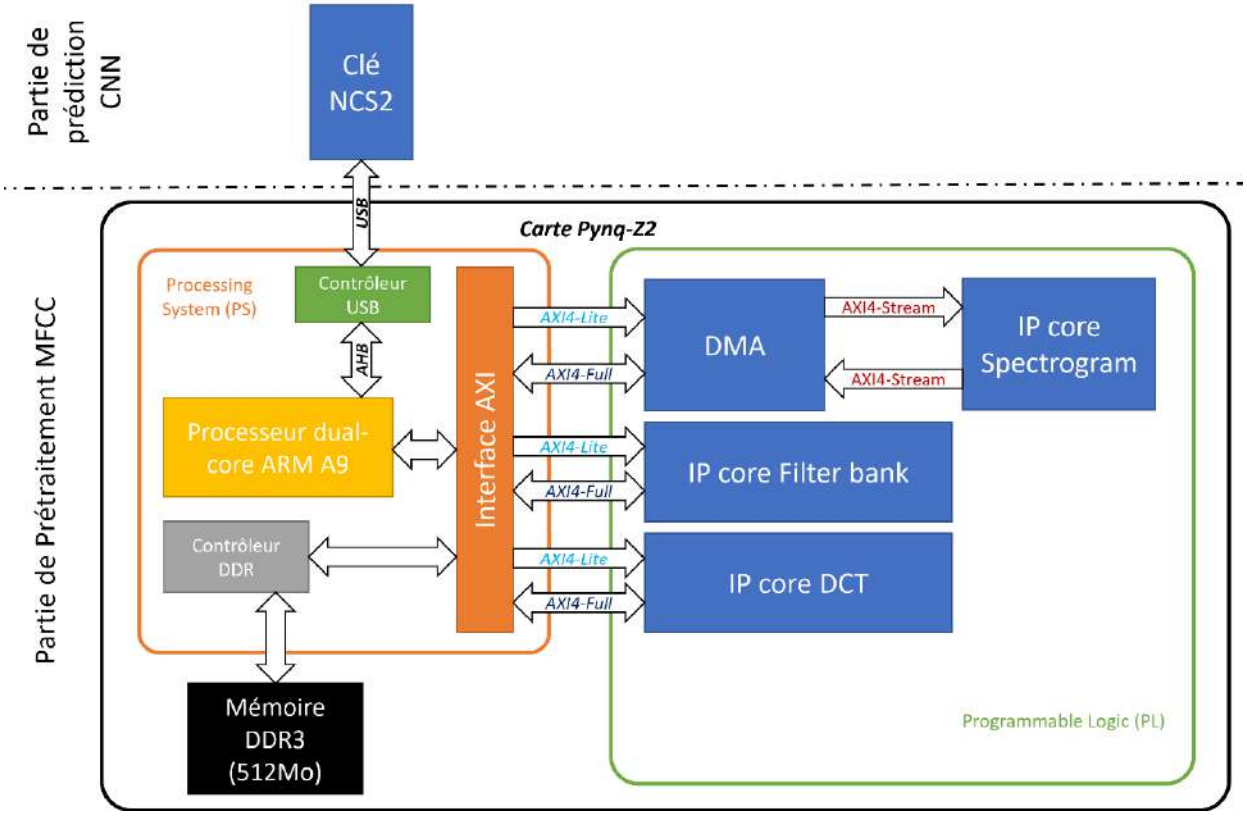


Figure III-1 : Schéma bloc du système réalisé

III.2.2 Description du système

Le système de détection des défauts que nous avons conçu se divise en deux parties : une partie dédiée au prétraitement MFCC, qui est implémentée sur une carte FPGA de type Pynq-Z2 et une seconde partie dédiée à la prédiction de la classe des défauts à l'aide d'un réseaux CNN, ce dernier est implémenté sur la clé Intel NCS2.

La partie de prétraitement est implémentée sur la carte FPGA qui comporte une partie processeur qui a pour rôle de gérer le système d'exploitation et les ressources mémoires, et une partie logique pour implémenter des accélérateurs matériels. La partie logique de la plateforme FPGA est reliée au processeur et à la mémoire par un ensemble de bus d'interconnexions de type AXI. Elle contient un ensemble de noyaux de propriétés intellectuels (IP-cores) qui servirons à calculer la matrice MFCC à partir des signaux vibratoires. Le résultat est alors une image comportant les caractéristiques MFCC.

Les images résultantes sont ensuite transférées vers la clé NCS2 à travers l'interface USB. L'accélérateur NCS2 effectue alors une prédiction sur la nature du défaut en propageant l'image dans le réseau CNN pré-entrainé. La classe résultante est ensuite retransmise vers la carte Pynq-Z2 pour afficher le résultat final du traitement.

III.2.3 Le SoC Zynq-7000

L'algorithme d'extraction des caractéristiques a été totalement implémenté dans un seul et même circuit intégré, le Zynq-7020. Ce dernier fait partie de la famille Zynq-7000 de Xilinx, qui est la première génération de SoC qu'a produit la société. L'architecture du Zynq-7000 comprend deux grandes parties : une partie de traitement figée (PS : Processing system) et une partie logique programmable (PL : Programmable Logic). Ces deux parties sont reliées entre elles avec un ensemble d'interconnexions sous forme de bus AXI [38] (figure III-2).

La partie PS gère généralement le système d'exploitation de l'application embarquée, elle intègre ainsi un ensemble de pilotes matériels de bas niveau pour contrôler l'exécution des accélérateurs matériels de la partie PL et a aussi pour rôle de gérer le transfert des données entre les différents espaces mémoires. La partie PS comprend :

- Un processeur à deux cœurs physiques de type Cortex ARM A9, augmenté d'une mémoire cache de 256 Ko.
- Un ensemble d'interfaces mémoires pour les échanges de données entre les deux parties PS et PL.

- Des périphériques d'entrées/sorties standards comme des contrôleurs USB, UART, I²C et SPI, etc. [38].

La partie PL du SoC constitue la partie FPGA configurable de l'application. Elle peut accueillir un ensemble de structures logiques instanciées par le concepteur du système. Les blocs de traitement de l'application sont délégués à la partie PL qui peuvent travailler indépendamment les uns des autres, ce qui permet d'avoir une architecture hautement parallèle et ainsi d'accélérer le traitement de manière significative. L'architecture matérielle que peut prendre la partie PL est configurée soit à travers l'interface JTAG ou bien directement par la partie PS en utilisant des API spécifiques.

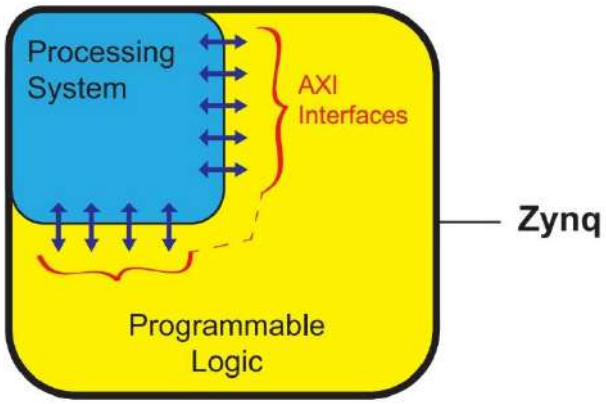


Figure III-2 : Schéma bloc simplifié du SoC Zynq [38]

III.2.4 Interfaces AXI

L'interface AXI (Advanced eXtensible Interface) fait partie de ARM AMBA (ARM, Advanced Microcontroller Bus Architecture), qui est un standard open pour l'interconnexion entre les différents blocs d'un système sur puce (SoC). La version actuellement utilisée est la version AXI4 qui spécifie trois types d'interfaces [39].

Le premier type est l'AXI4-Full. C'est une interface adressable qui permet l'interconnexion entre un IP-core maître et un autre IP-core esclave. Elle est constituée de 5 canaux : deux canaux d'échange d'adresses, deux canaux pour le transfert de données, et un canal réservé aux accusés de réception. Cette interface a pour avantage de permettre l'échange de données en simultané entre maître et esclave, avec des rafales allant de 1 à 256 données par transaction. Elle est idéale pour le transfert de grandes quantités de données entre les différents espaces mémoires du système. Pour assurer le transfert de données entre plusieurs IP-cores maîtres et esclaves qui sont reliés à l'aide de l'AXI4-Full, on utilise généralement l'IP-core AXI Interconnect.

Le second type d'interface est l'AXI4-Lite. Cette interface est en tout point similaire à l'AXI4-Full à la différence qu'elle ne permet pas d'envoyer plus d'une donnée par transaction. Cette interface allégée convient plus pour les traitements qui nécessitent un envoi limité de données entre le maître et l'esclave, ou bien pour la configuration des IP-cores en accédant aux différents registres de contrôle.

L'AXI4-Stream est le dernier type d'interface spécifié. Elle ne comporte pas de lignes d'adresses et n'est donc pas une interface adressable dans l'espace mémoire du SoC. Les données ne peuvent circuler que dans un seul sens et la taille des rafales de données est illimitée. Cette interface est donc idéale pour les traitements qui impliquent des flux de données continues tel que le traitement audio ou vidéo. Le transfert des données entre les différentes mémoires du système et un IP-core qui utilise cette interface est généralement assuré par un contrôleur d'accès directe à la mémoire (DMA).

III.3 La plateforme PYNQ

III.3.1.1 Le cadre de travail PYNQ

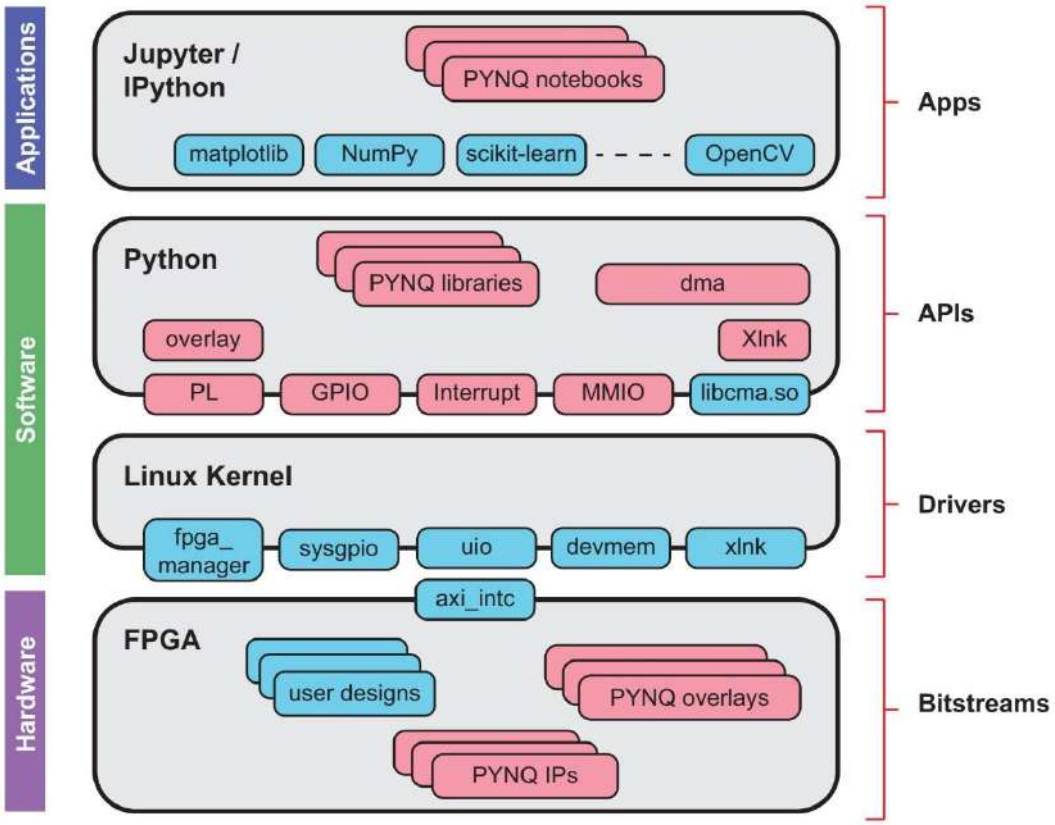


Figure III-3 : Couches du cadre de travail Pynq

L'implémentation d'un système embarqué nécessite un choix judicieux du cadre de développement autour du quel va s'articuler l'ensemble du matériel et logiciel associé au système au cours de son développement. Pour implémenter notre projet, nous avons décidé d'utiliser un cadre de travail PYNQ [38].

PYNQ est un framework à code source ouvert qui est destiné à faciliter la création et l'usage des applications construites autour des plateformes Zynq et Zynq MPSoC. Le nom PYNQ est dérivé de « Python productivity for Zynq » et utilise donc le langage de programmation Python pour simplifier la tâche du co-design en introduisant un ensemble de librairies matérielles et logicielles qui peuvent être utilisées selon les besoins de l'application visée.

Comme tout cadre de travail, PYNQ est constitué de plusieurs couches dépendantes les unes des autres et qui ont chacune un rôle particulier (figure III-3).

La couche inférieure concerne la partie matérielle du projet. Elle implique la conception de l'architecture du circuit que va incarner la partie PL de la plateforme mais aussi la configuration des interfaces de communication avec le PS. Cette partie est généralement créée sur le logiciel Vivado qui génère le bitstream (ficher *.bit) pour configurer la plateforme FPGA. Elle fournit aussi un ensemble d'API pour charger et contrôler l'architecture implémentée.

La couche intermédiaire concerne la partie logicielle de PYNQ. Elle englobe le système d'exploitation LINUX, les pilotes matériels et le langage Python. Cette couche inclue les librairies Python qui sont utilisées pour interagir avec les nombreux éléments du SoC Zynq, comme par exemple la gestion des adresses mémoires avec la classe MMIO et les pilotes du contrôleur d'accès direct à la mémoire DMA. Le système d'exploitation intégré est une variante personnalisée de la distribution Ubuntu 18.04, qui permet d'installer plusieurs accessoires et logiciels pour effectuer des tâches similaires à une machine de bureau ordinaire.

La partie supérieure touche l'application utilisateur. Cette dernière est facilitée par l'intégration d'un environnement de développement appelé Jupyter Notebook. C'est un environnement basé sur IPython (Interactive Python) qui permet de créer et d'écrire des codes sources interactifs en utilisant le langage de programmation Python dans un serveur web hébergé dans la carte. Il permet entre autres, d'écrire et d'exécuter le programme partie par partie en les organisant sous forme de cellules, de créer et d'intégrer des graphes de données traitées au cours de l'exécution du programme, et de l'illustrer en ajoutant des images et des formules mathématiques explicatives (figure III-4).

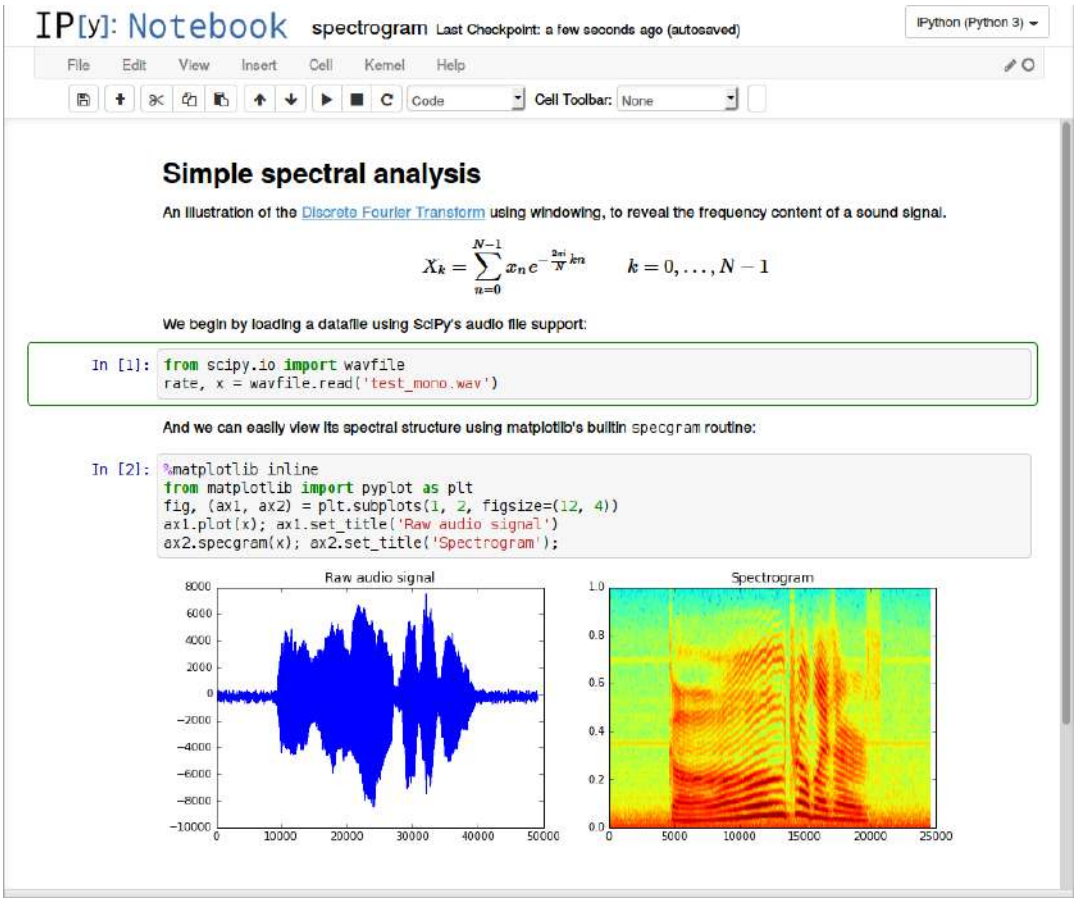


Figure III-4 : Exemple d'un script python sur jupyter notebook [40]

III.3.2 La carte Pynq-Z2

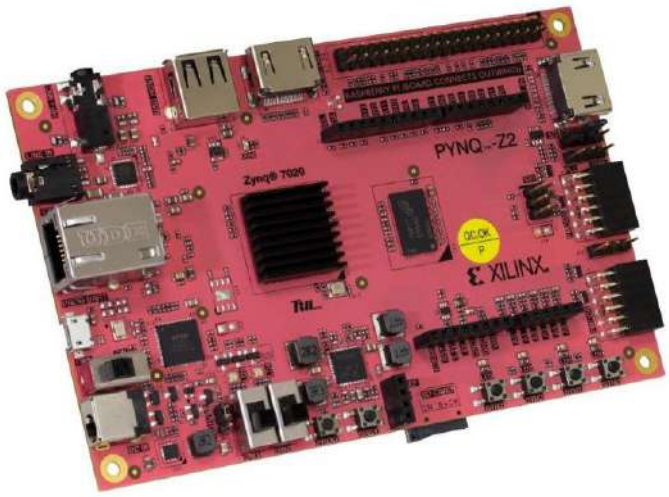


Figure III-5 : Photographie de la carte FPGA, PYNQ-Z2 [41]

L'environnement PYNQ a été créé pour faciliter le développement sur différentes cartes à circuits programmables. Parmi les nombreuses carte FPGA prise en charge on trouve la carte PYNQ-Z2. La carte PYNQ-Z2 (figure III-5) créée par la société Taiwanaises Tul Technology,

est une version revisitée de la première version PYNQ-Z1 de la société Digilent. Elle est construite autour de l'APSoC XC7Z02-1CLG400C de Xilinx. Les caractéristiques techniques de la carte sont résumées dans l'annexe B.

III.4 Implémentation du classificateur CNN

III.4.1 Modèle IA sur plateformes embarquées

Une étape essentielle de l'implémentation de notre méthode de détection des anomalies dans les roulements à bille consiste à effectuer une prédiction sur les images issues du prétraitement des différents IP-core utilisés et à montrer les images résultantes au réseau CNN dont le modèle a été spécifié au cours du chapitre précédant.

D'après les recherches effectuées, la société Xilinx ne propose pas de méthode simple pour transformer des réseaux de neurones créés sous l'environnement Tensorflow en IP-core implémentable sur plateforme FPGA. Aussi, l'exécution d'une prédiction sur le processeur de la carte PYNQ-Z2 serait trop lente.

Pour pallier à ce problème et éviter un travail fastidieux pour implémenter le réseau CNN sur notre plateforme embarquée FPGA, nous avons décidé d'utiliser un accélérateur matériel externe, c'est l'Intel Neural Compute Stick 2.

III.4.2 Présentation de la clé NCS2



Figure III-6 : Clé Intel Neural Compute Stick 2 [42]

Le Neural Compute Stick 2 (NCS2) (figure III-6) est un dispositif qui donne accès à une fonctionnalité de réseau de neurones, sans nécessiter de matériel volumineux et coûteux. Il permet ainsi d'intégrer des algorithmes de vision et d'intelligence artificielles aux plateformes embarquées. L'Intel NCS2 est basée sur le VPU (Vison Processing Unit) Intel Movidius™ Myriad X qui est équipé d'un accélérateur matériel dédié pour effectuer des prédictions sur les réseaux de neurones de type DNN. C'est l'outil parfait pour les développeurs, les analystes

scientifiques de données, les ingénieurs industriels, les ingénieurs en IA et les étudiants universitaires. La communication et l'échange de données entre le matériel hôte et le NCS2 s'effectue via le protocole USB (figure III-7).



Figure III-7 : Clé Intel NCS2 connectée avec la carte PYNQ-Z2 à travers l'interface USB

III.4.3 Le cadre de travail OpenVino

Pour bénéficier plainement des avantages qu'offre l'accélérateur NCS2, il faut utiliser un kit d'outils spécifique appelé OpenVino (pour Open Visual Inference and Neural network Optimisation toolkit). OpenVino contient un grand nombre d'outils pour transformer les modèles de réseaux de neurones, et permet aussi de les optimiser pour effectuer la prédiction sur les différents types de matériel que propose la société Intel (CPU, GPU, VPU et FPGA).

III.4.4 Implémentation du réseau de neurones CNN

Pour implémenter notre réseau CNN modélisé durant le chapitre précédant sur le NCS2, nous avons utilisé l'outil Model Optimizer (MO) d'Intel OpenVino et qui sert à transformer les modèles de réseaux de neurones créés sur les environnements Tensorflow, Caffe, ONNX, MXNet et Kaldi en modèles IR prêt à être exécutés sur les accélérateurs matériels que propose Intel, dont le NCS2 (figure III-8).

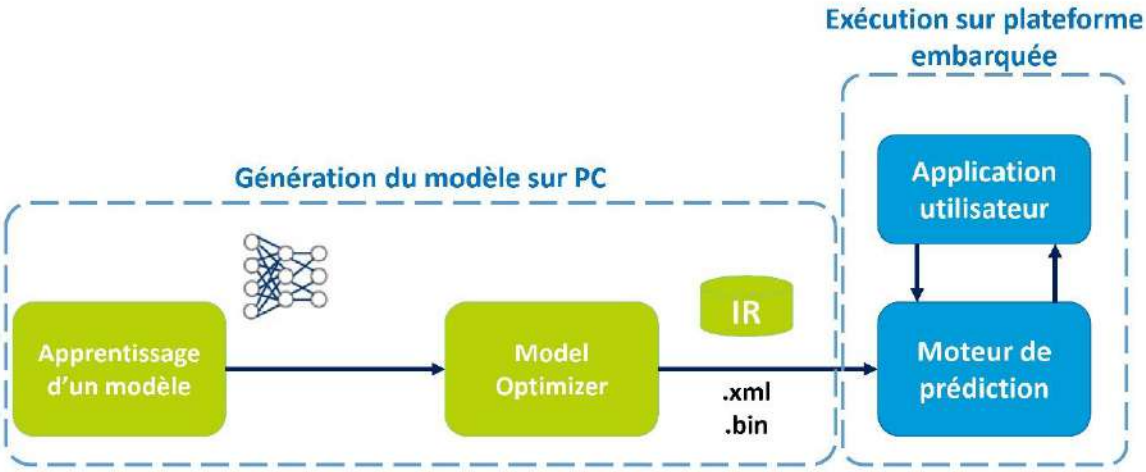


Figure III-8 : Conversion du modèle CNN en image IR pour la clé NCS2 [43]

III.5 Implémentation du prétraitement sur plateforme FPGA

III.5.1 La synthèse haut niveau HLS

Pour intégrer l'ensemble des IP-cores utilisés dans notre projet, il nous a fallu choisir un langage de description matériel qui soit simple et rapide à utiliser.

Le processus de création d'architectures matériels sur les plateformes FPGA a significativement évolué pendant les dernières décennies. Au début les circuits étaient simples, les concepteurs pouvaient facilement manipuler chaque transistor, la manière dont ils étaient reliés entre eux et leurs dispositions dans le circuit intégré, tout le travail était effectué manuellement. Les puces électroniques sont devenues ensuite de plus en plus complexes, au lieu de spécifier chaque transistor, le concepteur pouvait désigner un bloc de circuit numérique confectionné en utilisant des logiciels CAO. Au début des années 1980, sont apparus des langages de description matériel (Verilog, VHDL) qui ont forcé les concepteurs à passer à des langages de programmation de matériel encore plus abstraits, le Register Transfer Level (RTL). Le comportement d'un circuit pouvait alors être défini en termes d'envoi de signaux ou de transferts de données entre registres, et les différentes opérations logiques effectuées sur ces signaux. Ces langages permettaient de concevoir des systèmes beaucoup plus complexes sans se perdre dans les détails.

Une nouvelle étape dans l'abstraction apparaît dans l'utilisation d'un langage de synthèse haut niveau (HLS). Cet outil exploite le langage C/C++ ou system C pour décrire le matériel qui compose le circuit intégré. Fondamentalement, HLS permet d'automatiser des tâches qui

sont habituellement faites manuellement sous RTL, ces tâches peuvent être résumées dans les points suivant [44] :

- HLS analyse et exploite la concurrence dans un algorithme ;
- HLS insère les registres nécessaires pour limiter les chemins critiques et atteindre une fréquence d'horloge souhaitée ;
- HLS génère les circuits de contrôle qui dirigent le chemin des données ;
- HLS implémente des interfaces pour se connecter aux différents éléments du système ;
- HLS localise les données sur les unités de stockage pour équilibrer l'utilisation des ressources et la bande passante ;
- HLS affecte le calcul sur des éléments logiques en effectuant des optimisations automatiques et/ou spécifiées par l'utilisateur pour obtenir l'implémentation adéquate.

III.5.2 Vivado HLS

Vivado HLS est un logiciel de synthèse haut niveau créé par la société Xilinx. Il est basé sur l'environnement de développement Eclipse, il intègre ainsi un ensemble d'outils qui facilite la synthèse des IP-cores. En utilisant les langages C/C++ et système C, Vivado HLS arrive à créer un équivalent en langage HDL, à simuler le circuit synthétisé et enfin à générer la description RTL du circuit pour l'utiliser dans Vivado Design Suite.

III.5.3 Procédure de création d'un IP-core sous Vivado HLS

La procédure de création d'un IP-core sous Vivado HLS impose une méthode de travail qui tient en six étapes. Pour expliquer l'ensemble des étapes et leurs importances, nous présentons ici un exemple simple de création d'un IP-core qui effectue l'addition de deux nombres réels a et b, qui effectue l'opération suivante :

$$c = a + b \text{ où } a, b, c \in \mathbb{R} \qquad \textit{Éq 29}$$

Etape 1 : Création d'un projet et définition de la fonction

Le processus démarre par la création d'un nouveau projet Vivado HLS, le choix de la puce FPGA visée, puis à rédiger le traitement effectué par l'IP-core sous forme d'une fonction principale appelée « Top function » (figure III-9). On peut aussi utiliser plusieurs sous-fonctions dans notre programme et les appeler dans notre fonction principale.

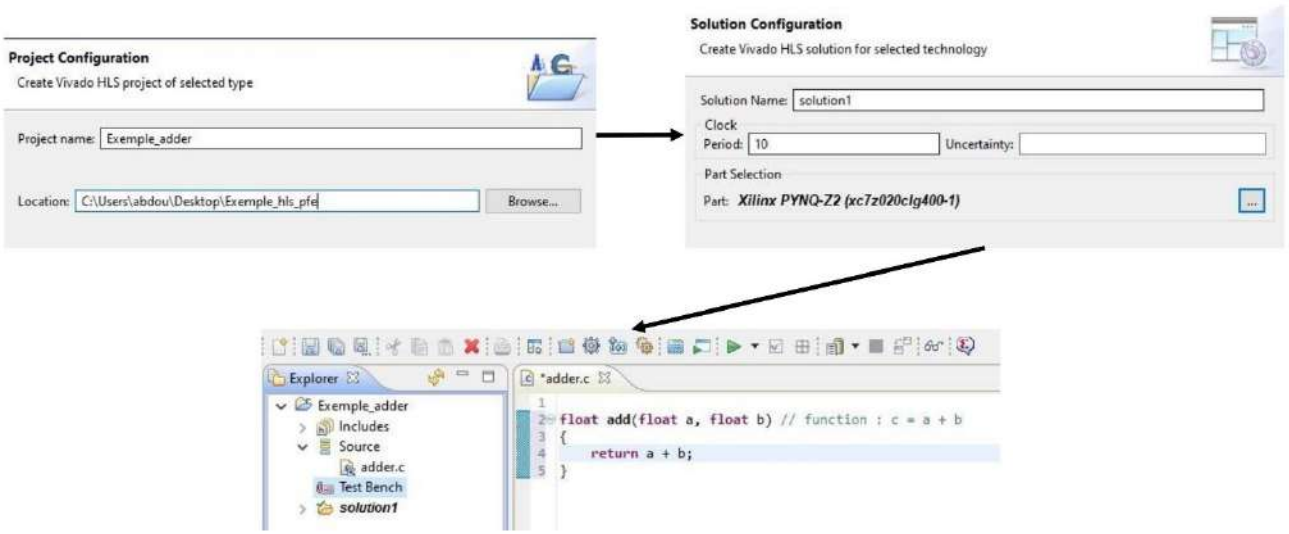


Figure III-9 : Étapes de démarrage d'un nouveau projet Vivado HLS

Etape 2 : Tester la validité de l'algorithme

La seconde étape consiste à vérifier la validité de l'algorithme écrit en langage C/C++ en utilisant un testbench (figure III-10). Le testbench est un programme C/C++ qui va appeler la fonction implémentée durant la première étape. Le testbench n'a aucune incidence sur la synthèse de l'IP-core et n'est pas obligatoire durant le processus de conception.

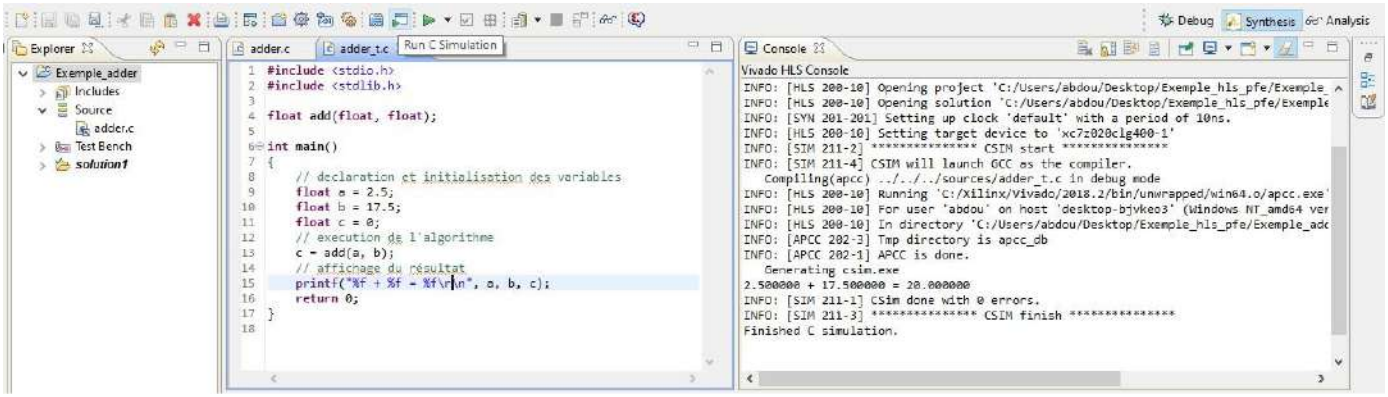


Figure III-10 : Test de l'algorithme à implémenter avec un testbench C/C++

Etape 3 : Synthèse de l'IP-core

Après s'être assuré de la validité de l'algorithme en C/C++, on aborde la synthèse du traitement en RTL en employant diverses directives. Les directives (*#pragmas*) sont utilisées pour guider le logiciel Vivado HLS durant la synthèse de l'IP-core, ce sont des suggestions qu'apporte le développeur pour :

- Décider des interfaces d'entrée/sorties des IP-cores à implémenter

- Définir l'emplacement et le format de stockage des différentes variables.
- Accélérer l'exécution du traitement, par exemple en déroulant les boucles ou en implémentant un pipeline.

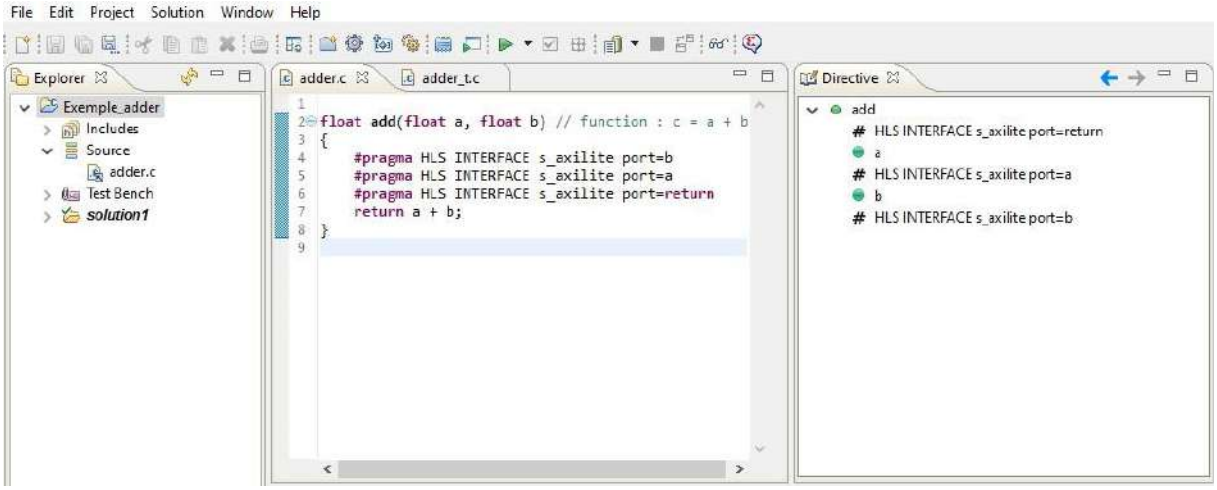


Figure III-11 : Ajout des directives pour orienter HLS durant la synthèse de l'IP-core

Ces décisions sont importantes dans la gestion des ressources que va utiliser l'IP-core dans la conception finale, et influent grandement sur la rapidité d'exécutions du traitement qu'on lui a attribué. Dans le cas général, il faut faire un compromis entre ces deux aspects. Pour notre exemple, les entrées *a* et *b* ainsi que la sortie *c*, sont implémentées en tant que registres accessible à travers l'interface adressable AXI-Lite (figure III-11).

Etape 4 : Aperçu des performances et des ressources

Le processus de synthèse génère ensuite un rapport détaillé de la conception, qui comprend, l'intervalle de temps que met l'IP-core pour démarrer une prochaine exécution, la latence de traitement et une estimation des ressources utilisées (figure III-12).

Etape 5 : Exporter l'IP-core

La dernière étape consiste à empaqueter et à exporter l'implémentation RTL en IP-core afin de l'utiliser dans de futures conceptions sur le logiciel Vivado Design Suite (figure III-13).

Il est aussi à noter qu'on peut effectuer une co-simulation qui transforme le testbench C/C++ en un testbench HDL qui servira ensuite à observer les différents signaux que génèrera l'IP-core durant son exécution.

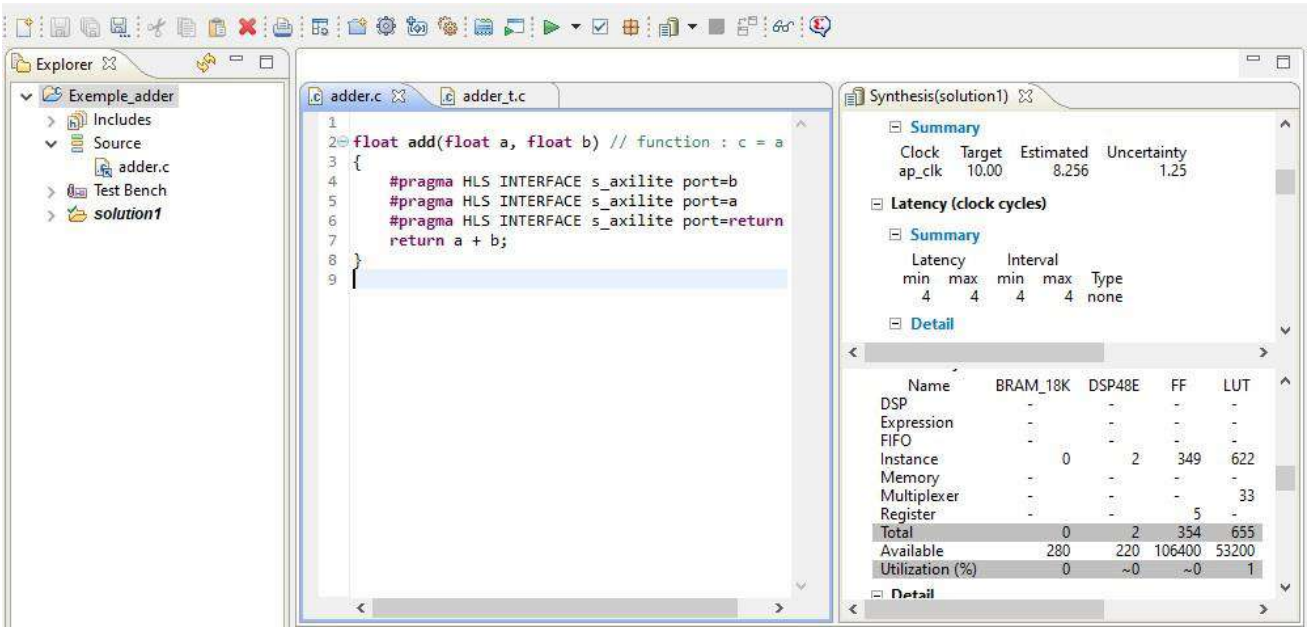


Figure III-12 : Vérification des ressources matérielles et du temps d'exécution estimé

Nous obtenons à la fin un IP-core (nommé add_0 dans notre exemple) prêt à être implémenté dans un projet dans le logiciel Xilinx Software Developement Kit (figure III-14).

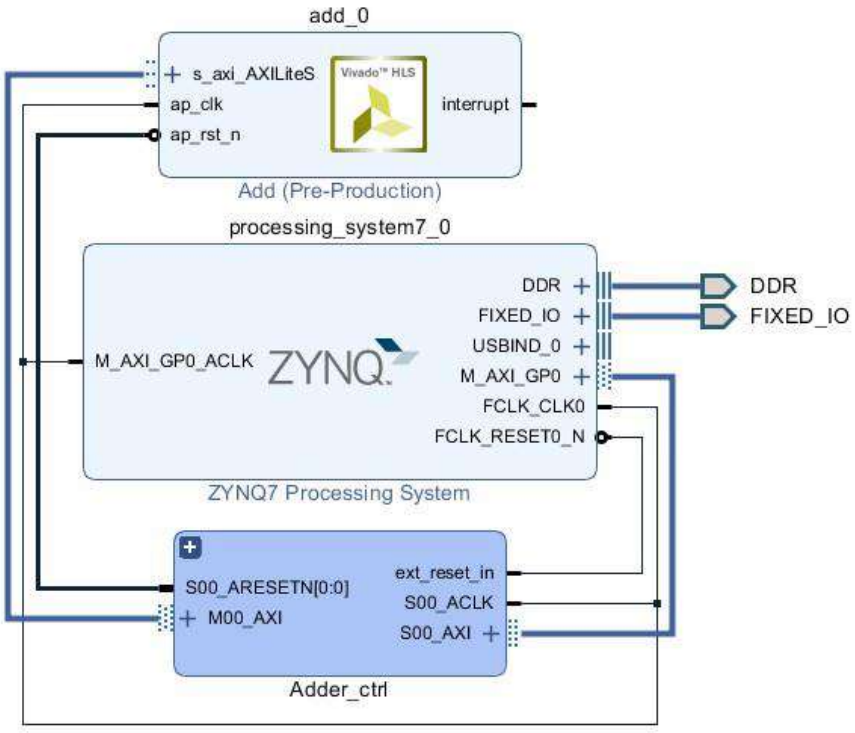


Figure III-13 : IP-core résultant «Add » implémenté dans Vivado Design suite

L'ensemble des étapes et la méthode de travail peuvent être résumés sur l'organigramme (figure III-15).

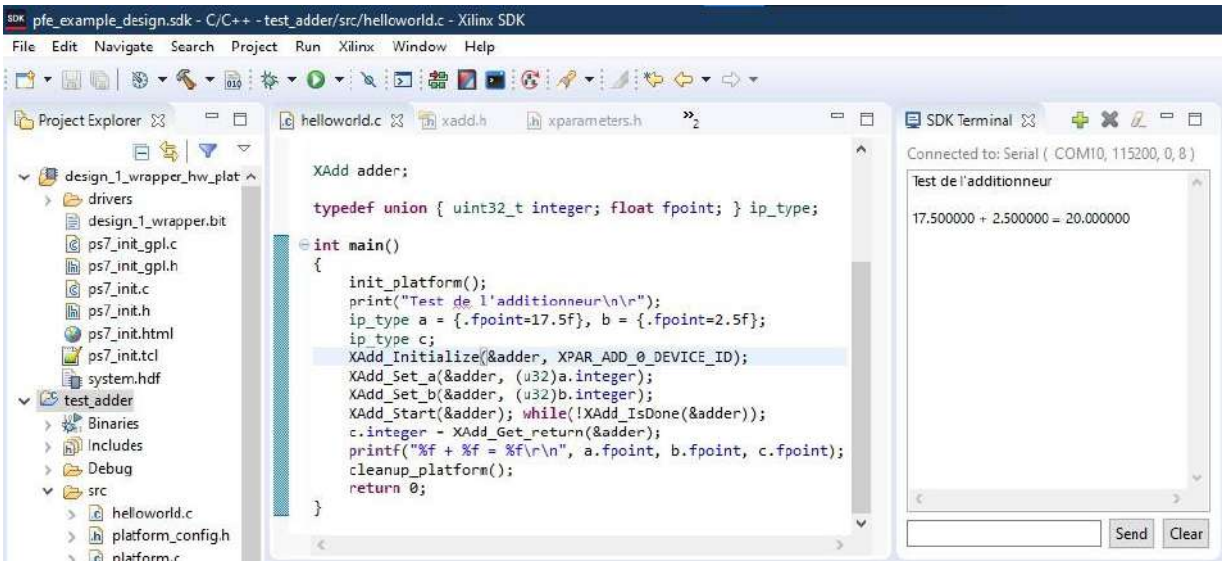


Figure III-14 : Test de l'IP-core « add » sur la carte FPGA

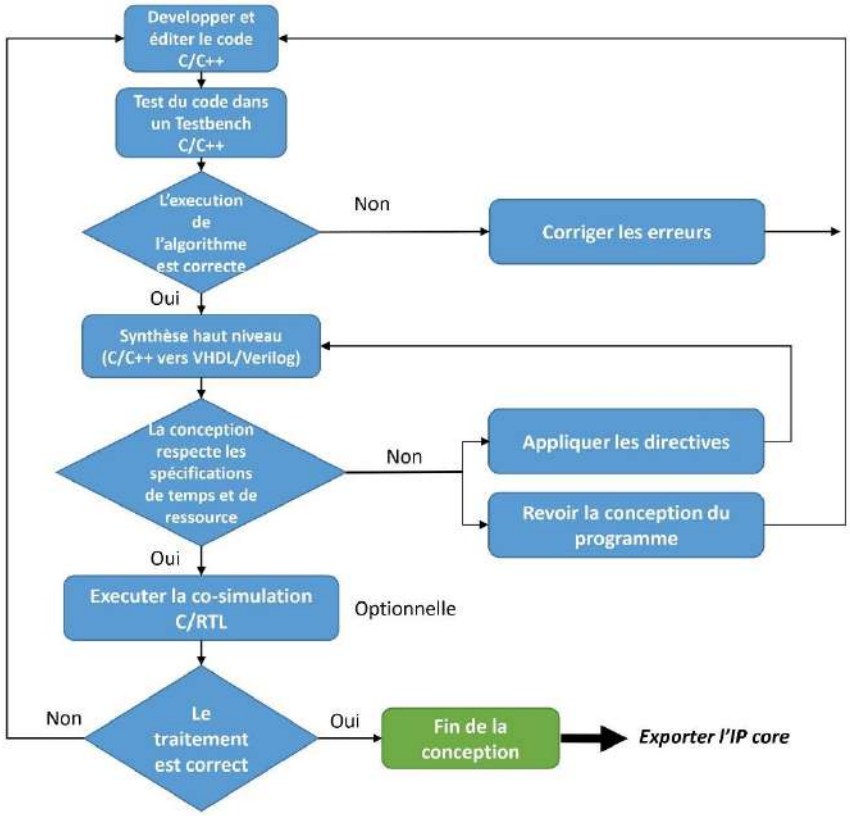


Figure III-15 : Organigramme de développement d'un IP-core sous Vivado HLS [45]

III.5.4 Création des différents IP-cores

La figure II-19 du chapitre 2 explique l'ensemble des étapes de la méthode MFCC appliquée aux signaux vibratoires pour obtenir une matrice de coefficients. Chaque colonne d'une image est le résultat d'un ensemble de traitements mathématiques exécutés selon un ordre

précis. Il est donc logique de diviser chacune des étapes du traitement MFCC sur différents IP-cores. L'architecture réalisée sur vivado est alors présentée sur la (figure III-16).

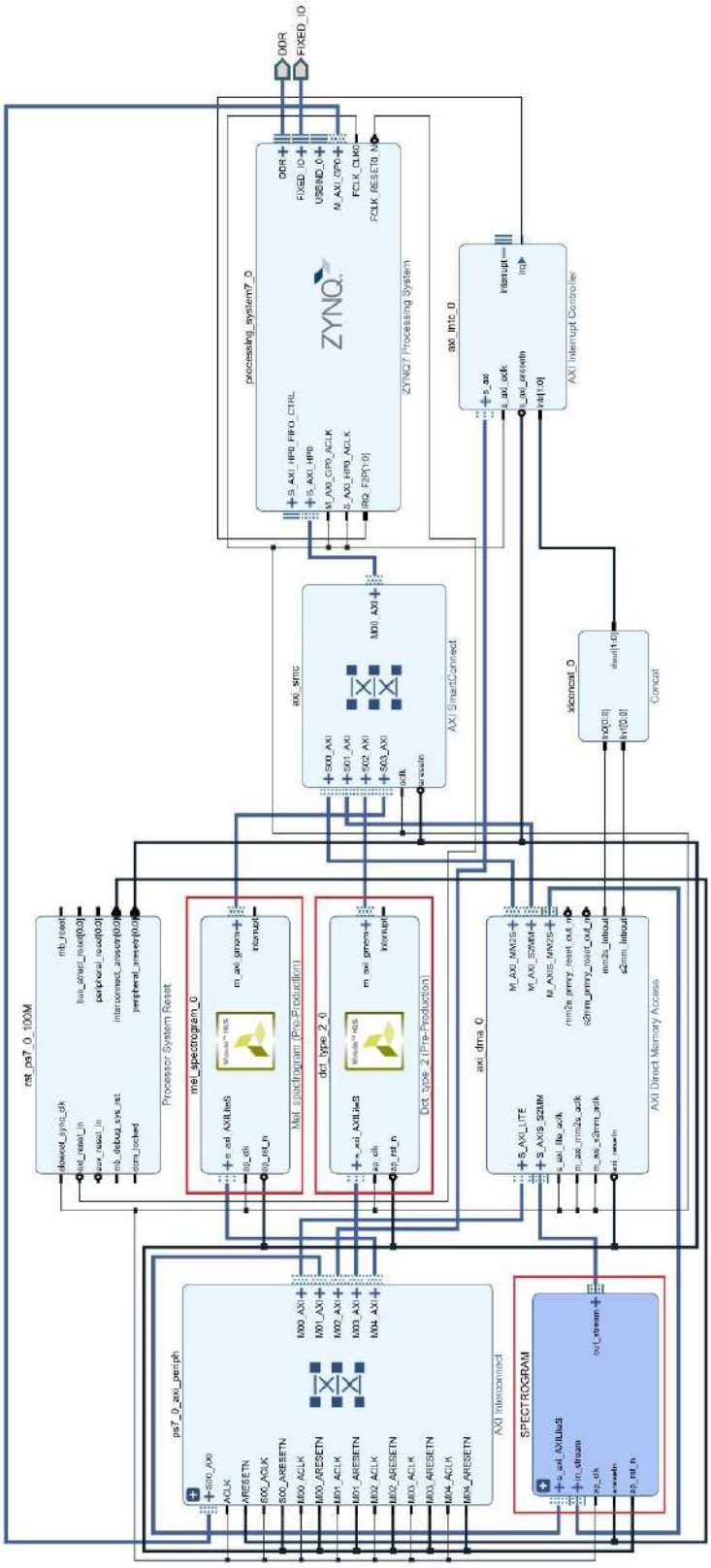


Figure III-16 : Architecture du système de prétraitement implémenté sur Vivado Design Suite

III.5.4.1 Spectrogramme

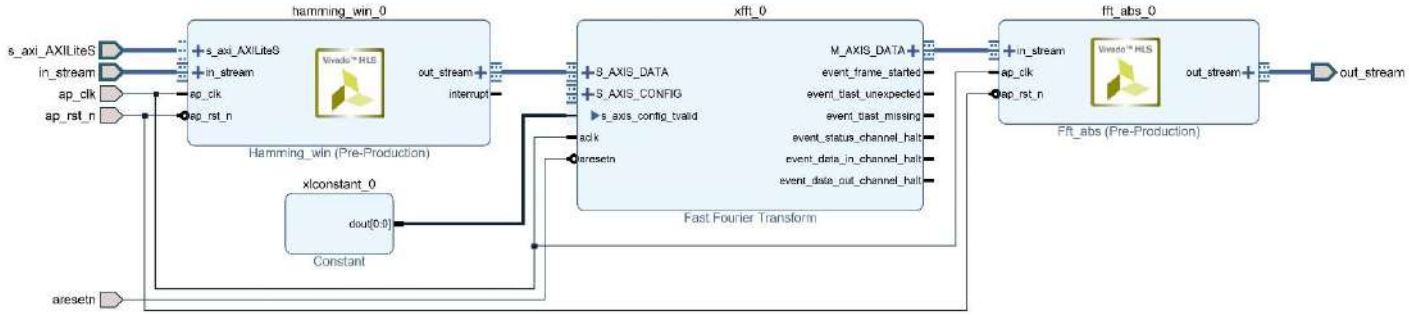


Figure III-17 : Éléments de la hiérarchie Spectrogram

La hiérarchie « SPECTROGRAM » a pour rôle de calculer la puissance spectrale estimée. Elle prend en entrée le signal vibratoire réel et donne en sortie la puissance spectrale de ce signal. Elle est composée de 3 IP-cores (figure III-17)

Cette hiérarchie consomme les données du signal vibratoire fournies par le contrôleur DMA depuis la mémoire DDR, effectue le traitement imposé, puis génère et délivre les données de sortie au même contrôleur pour stocker le résultat dans un autre emplacement de la mémoire DDR.

On a fait le choix de relier les différents IP-cores de cette hiérarchie à travers l'interface AXI-Stream. Ce choix nous a permis d'éviter des lectures et écritures inutiles dans la mémoire DDR qui peuvent mener à une baisse de performances (figure III-18).

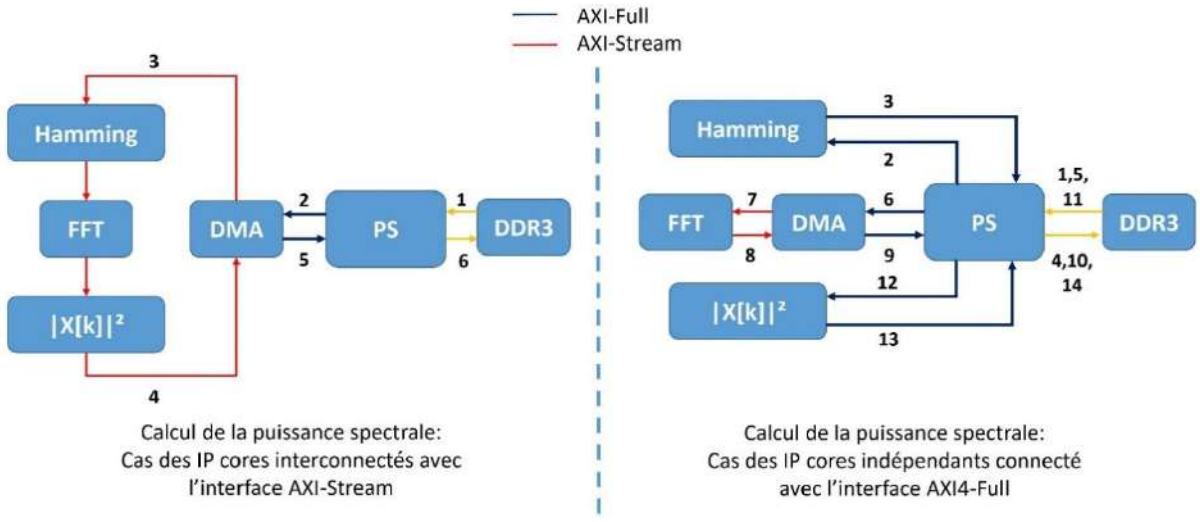


Figure III-18 : Impacte du choix d'interface sur le temps de traitement, la donnée suit un chemin plus court dans le cas où l'on utilise une interface AXI-Stream (la donnée suit le chemin décrit par l'ordre des chiffres)

Le processeur qui gère le système a alors pour seule tâche de configurer les adresses de données d'entrée et de sortie, et de spécifier la taille de données à déverser par le contrôleur DMA. La figure III-19 donne le résultat de calcul de la hiérarchie « Spectrogram » sur un signal x(t) de taille de 2048 points et échantillonné à Fs = 22050 Hz.

$$x(t) = \sin(500.2\pi t) + \sin(3100.2\pi.t) + \sin(6000.2\pi.t) + \sin(9000.2\pi.t) \qquad \text{Éq 30}$$

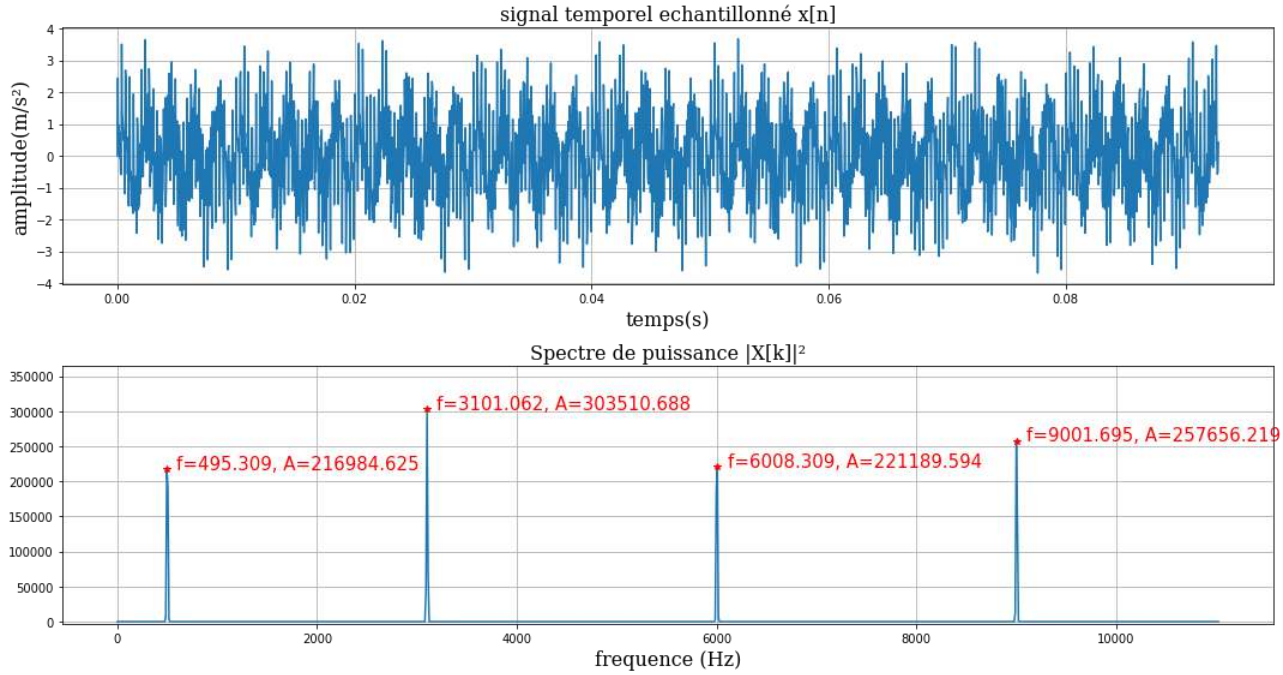


Figure III-19 : Calcul du spectre de puissance de x[n] sur la carte PYNQ-Z2 en utilisant la hiérarchie « Spectrogram »

III.5.4.2 Fenêtre de Hamming

L'IP-core « hamming_win_0 » a pour mission de récupérer le signal envoyé par le contrôleur DMA, d'appliquer un fenêtrage de type Hamming au signal d'entrée et enfin de délivrer ce signal à l'IP-core suivant en utilisant l'interface AXI-Stream (figure III-20). Durant la phase de conception sous HLS, cet IP-core a été décrit pour délivrer les données de manière continue en appliquant la pondération de la fenêtre à chaque échantillon de signal reçu.

Le gestionnaire du système peut lancer l'exécution de cet IP-core à tout moment en accédant au registre CTRL de l'IP-core à travers l'interface AXI-Lite.



Figure III-20 : IP-core « hamming_win » et ses différentes interfaces de connection

III.5.4.3 Transformé de Fourier rapide (FFT)

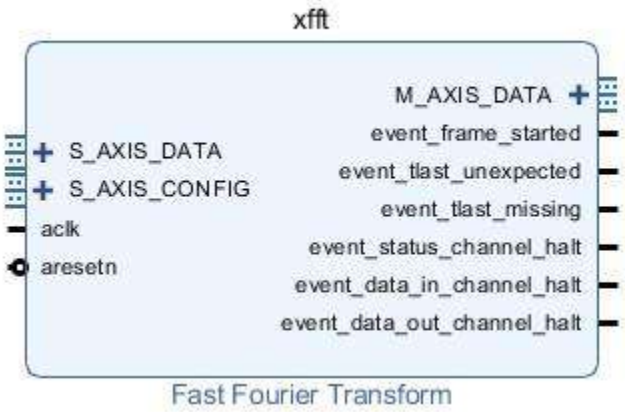


Figure III-21 : IP-core fft V9.1 de Xilinx

La prochaine étape du calcul de la puissance spectrale est la transformé de Fourier rapide. Nous avons décidé de ne pas redéfinir le traitement FFT sur HLS, mais d'utiliser l'IP-core Fast Fourier Transform V9.1 de Xilinx (figure III-21). Cet IP-core est fournie par défaut avec Vivado Design Suite. La tâche de cet IP core est de calculer le spectre complexe du signal d'entrée qui est fourni par l'IP-core « hamming_window_0 ». Cet IP-core a été configuré pour recevoir 2048 point d'entrées en virgule flottante, puis pour délivrer le spectre complexe du signal dans l'ordre naturel en virgule flottante. L'interface d'entrée et de sortie sont toute deux de type AXI-Stream.

III.5.4.4 Puissance spectrale ($|X[k]|^2$)

Cet IP-core sert à calculer la puissance spectrale estimée du signal à partir des données complexes délivrées par l'IP-core « xfft » à travers l'interface AXI-Stream (figure III-22). Les résultats du traitement sont ensuite récupérés par le contrôleur DMA en utilisant le même type d'interface (figure III-17).



Figure III-22 : IP-core « fft_abs » et ses différentes interfaces

III.5.5 Banc de filtres

Cet IP-core a pour rôle de pondérer le spectre d'amplitude par un ensemble de filtres triangulaires. Pour effectuer cette tâche, on a choisi d'utiliser l'interface AXI-Master (figure III-23). Cette interface permet de déléguer, de la même façon que le contrôleur DMA, l'ensemble des opérations de lectures et d'écritures dans la mémoire DDR à l'IP-core, il devient alors maître du bus de données pendant la durée de l'exécution.

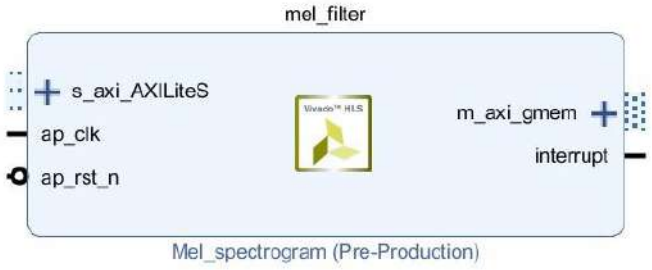


Figure III-23 : IP-core « mel-filter » et ses différentes interfaces

Pour configurer le fonctionnement de cet IP-core, le processeur gestionnaire du système aura seulement à indiquer l'emplacement des données d'entrée et de sortie dans la mémoire DDR en plaçant leurs adresses respectives dans des registres accessibles à travers l'interface adressable AXI-Lite de l'IP-core. Le lancement de l'exécution est, quant à lui, contrôlé par le registre CTRL qui est de même accessible par la même interface AXI-Lite.

III.5.6 Transformé en cosinus discrète

Cet IP-core effectue la transformée en cosinus discrète d'un signal réel (figure III-24). L'interface de cet IP-core est exactement similaire à l'IP-core « mel_filter ».

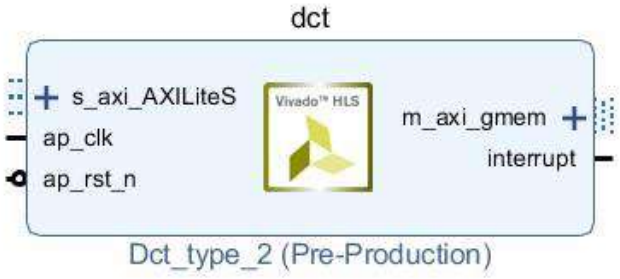


Figure III-24 : IP-core « dct » et ses différentes interfaces

III.5.7 Organigramme

Afin de tester l'architecture implémentée sur la carte PYNQ, on a réalisé un programme qui suit l'organigramme suivant :

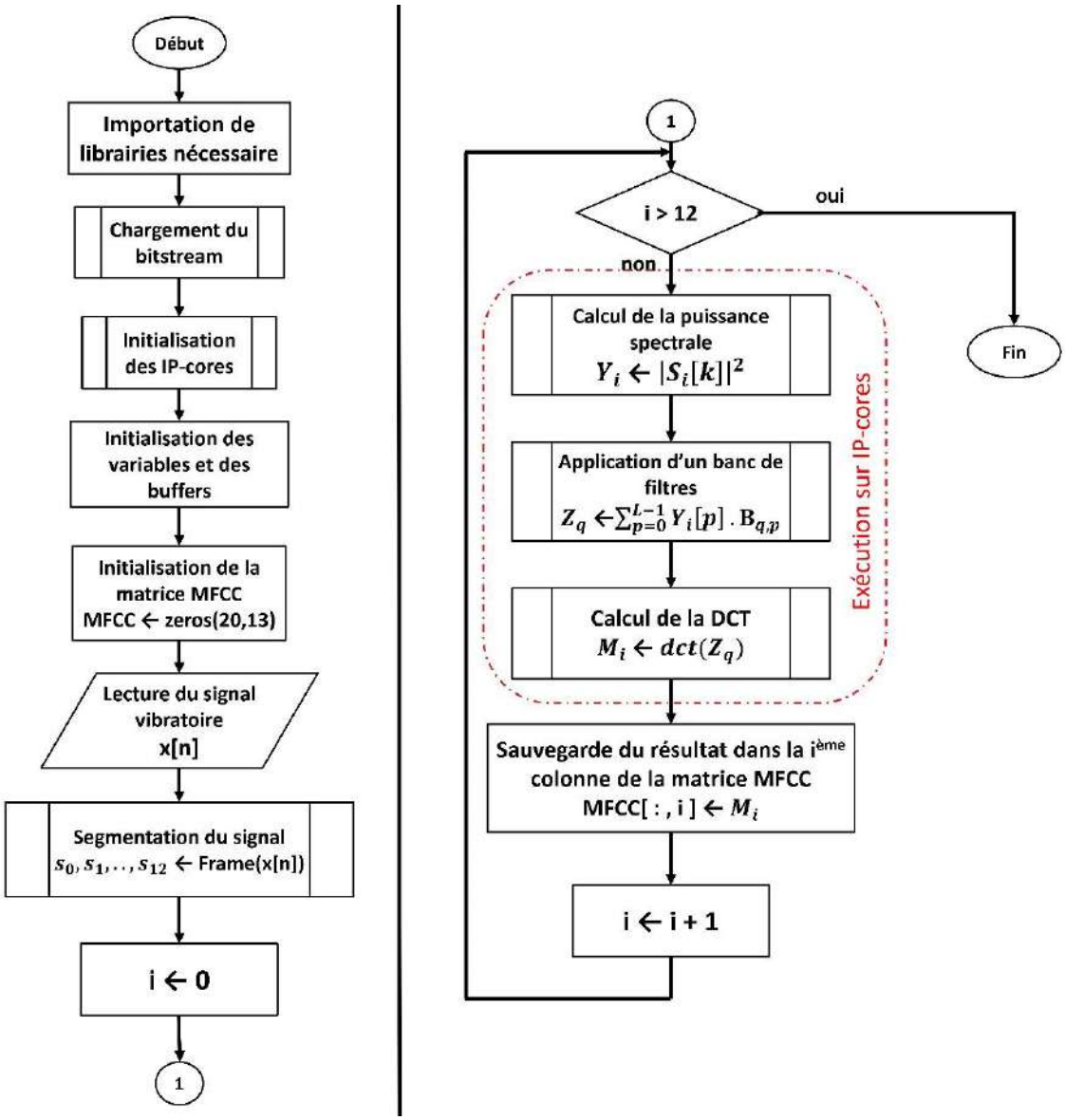


Figure III-25 : Organigramme de calcul de la matrice MFCC à l'aide des IP-cores sur PYNQ-Z2

III.5.8 Résultats de l'implémentation

Durant l'implémentation de la méthode de prétraitement, nous nous sommes assurés que les résultats obtenus par l'implémentation sont corrects (la fonctionnalité), du nombre de ressources utilisées dans la partie PL du système (optimisation côté ressources) et du temps d'execution du traitement sur la plateforme embarquée (optimisation de la vitesse du traitement).

III.5.8.1 Exactitude des résultats

La première chose que l'on a vérifié durant l'implémentation est la validité du traitement effectué par l'enchainement des différents IP-core. Nous nous sommes d'abord assurés de l'exactitude des résultats délivrés par chacun des blocs de traitement effectués par la hiérarchie « Spectrogram », l'IP-core « mel_filter » et l'IP-core « dct ». Puis nous nous sommes penchés sur la justesse du traitement MFCC globale. Pour effectuer cette dernière vérification, nous avons effectué la différence entre une image résultante d'un traitement purement software sur python et une autre image résultante d'un traitement sur les IP-cores. La (figure III-26) est un exemple de vérification, où l'on peut remarquer une similarité presque parfaite entre les deux images, la différence minime est due à l'utilisation d'un algorithme modifié dans l'IP-core « mel_filter » pour réduire l'utilisation des ressources.

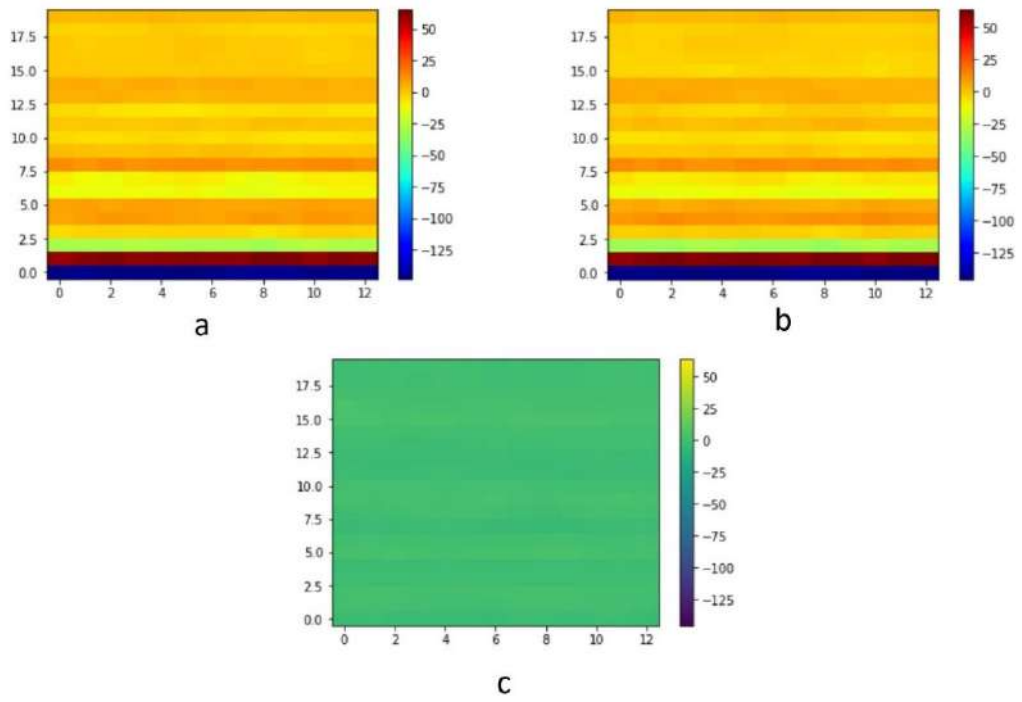


Figure III-26 : Comparaison entre le traitement mfcc du même signal vibratoire
a : traitement sur FPGA, b : traitement sur PC, c : différence entre les deux images

III.5.8.2 Ressources utilisées

Nous avons veillé à utiliser le minimum de ressources possible dans la partie FPGA du système dans le but de laisser un espace assez important pour ajouter d'autres traitements et fonctions à notre SoC. Le tableau III-1 résume les ressources utilisées ainsi que la puissance électrique estimée pour faire fonctionner le système. Le pourcentage d'utilisation des ressources est globalement inférieur à 25%.

Tableau III-1 : Résultats de synthèse et d'implémentation

Ressources	Utilisés	Disponibles	Utilisation (%)
Slice LUT	12304	53200	23.13
LUT RAM	1582	17400	9.09
Bascules (FlipFlop)	16754	106400	15.75
Bloc RAM	24.5	140	17.50
Slices DSP	42	220	19.09
Puissance totale estimée sur puce	1.603 W		
Fréquence maximale des IP cores	100 MHz		

III.5.8.3 Temps d'exécution

Le dernier critère que nous avons vérifié est le temps d'exécution. Ce dernier a été mesuré dans l'environnement Xilinx SDK pour vérifier les performances exactes des différents IP-cores, puis dans l'environnement PYNQ dans le terminal Linux pour évaluer les performances en utilisation réelle.

Pour avoir une mesure relative des performances des traitements effectués, on a comparé les temps de traitement des IP-cores avec une autre plateforme embarquée, le microcontrôleur STM32L433CCT6 [46]. Ce dernier intègre un processeur ARM Cortex M4 avec une unité de calcul en virgule flottante (FPU), il a aussi été cadencé à sa fréquence d'horloge maximale de 80MHz, pour avoir une comparaison juste avec les IP-cores qui sont cadencés à 100 MHz. Pour exploiter la totalité des performances du microcontrôleur, nous avons aussi utilisé la libraire de fonctions mathématiques ARM_MATH qui accélère grandement les fonctions de traitement du signal tel que la multiplication vecteur par vecteur, la FFT et les multiplications matricielles. Le tableau III-2 résume les temps d'exécution des différents blocs de traitement. On remarque alors que l'accumulation des accélérations matériels de la plateforme FPGA nous amènent à un traitement quatre fois plus rapide par rapport au même traitement effectué sur microcontrôleur.

Tableau III-2 : Comparaison des temps d'exécution µC / IP-core

Tâche / Exécution	Hamming FFT $\|X[k]\|^2$ Temps (ms)	Filtrage Temps (ms)	DCT Temps (ms)	Colonne Temps (ms)	Matrice de 13 colonnes Temps (ms)
IP cores seulement	0.409	0.113	0.025	0.545	7.105
Mono cœur Arm M4F (release –O2 + Librairie ARM_MATH)	2.228	0.275	0.049	2.551	33.166
Accélération apportée par l'IP-core	x 5.45	x 2.44	x 1.94	X 4.68	x 4.66

La seconde mesure des performances a été effectuée dans l'environnement PYNQ. Les tests dans le terminal Linux nous montrent une baisse de performance des IP-cores qui est due à l'utilisations des API du cadre de développement (figure III-28).

III.6 Résultats du classificateur CNN

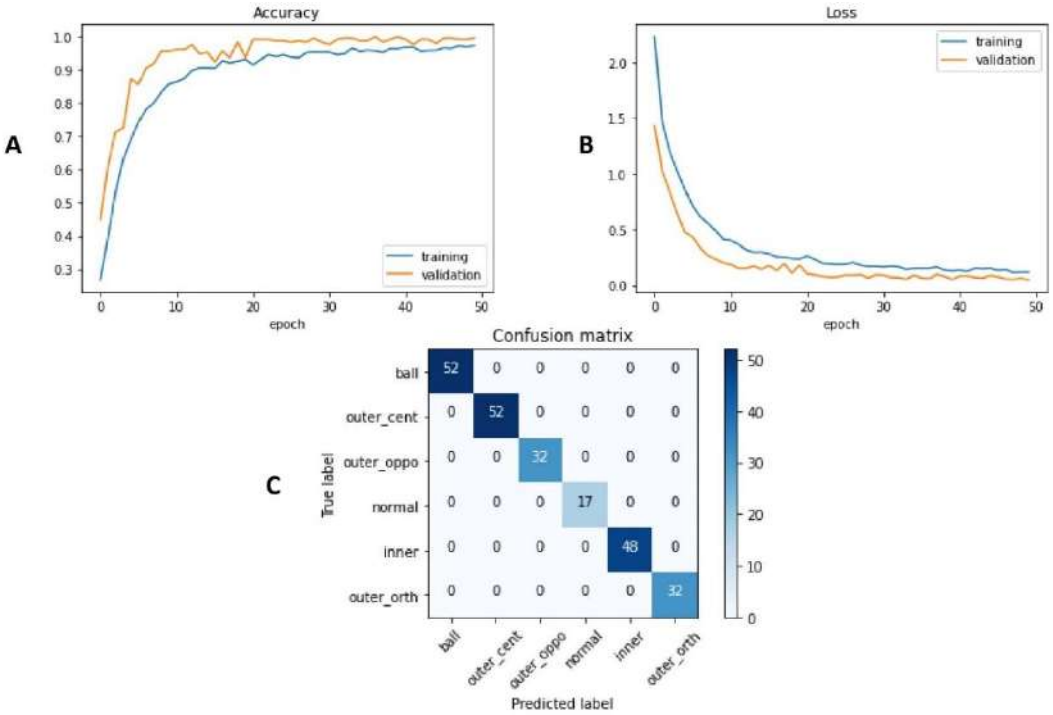


Figure III-27 : Résultat du modèle CNN sur le système embarqué

A : graphe de l'évolution de l'erreur, B : graphe de l'évolution de la précision

C : résultat de classification des signaux de tests (10% du dataset)

Le but de notre projet est d'utiliser les images MFCC résultantes du prétraitement des signaux vibratoires effectué dans les IP-cores pour connaître l'origine du défaut dans les roulements à billes. Pour ce faire, on a regénéré la même base de données du chapitre 2 sur la carte PYNQ en utilisant les IP-cores conçus, puis effectuer un apprentissage du réseau CNN sur PC. Le modèle du réseau CNN résultant a été ensuite transformé en modèle IR implémentable sur la clé NCS2 à l'aide du kit d'outils OpenVino. Les résultats de l'apprentissage et de tests sont résumés sur les (figure III-27).

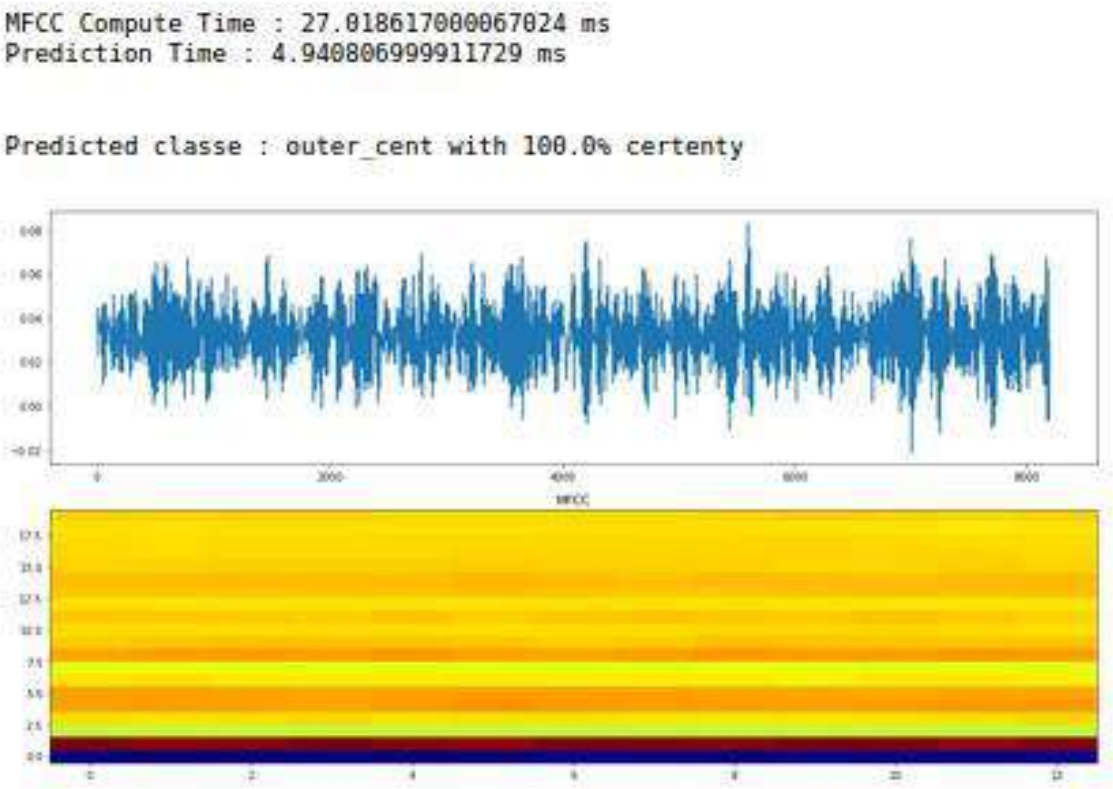


Figure III-28 : Résultat de prédiction sur un signal vibratoire d'un défaut de bague externe et temps d'exécution du prétraitement et de la prédiction sur carte PYNQ-Z2 + NCS2

Les prédictions des signaux de test sont correctes à 100% avec un temps de traitement (extraction des caractéristiques sur IP-core et prédiction sur NCS2) inférieur à 40ms (figure III-28).

III.7 Conclusion

Ce chapitre a été consacré à l'implémentation de notre méthode d'extraction des caractéristiques des signaux vibratoires sur la carte PYNQ-Z2 grâce à l'utilisation d'un langage haut niveau, le langage C/C++, sous le logiciel Vivado HLS. On a constaté une similarité presque parfaite entre les résultats des images issues du traitement MFCC sur ordinateur et celles issues de l'implémentation sur la plateforme embarquée. Le temps d'exécution a

nettement été amélioré par rapport à un système embarqué basé sur un microcontrôleur, mais on a constaté une baisse de performance due à l'utilisation des différentes API de du cadre de développement PYNQ. Les images résultantes ont pu être classifiées correctement en utilisant un accélérateur externe, la clé Intel NCS2.

Conclusion générale

L'objectif de ce travail était d'implémenter sur une carte FPGA, un système de détection automatique des défauts pour les machines tournantes en utilisant les signaux vibratoires générés par les différents éléments qui les composent. On a trouvé que la méthode d'extraction des caractéristiques MFCC était une méthode adaptée pour les applications qui visent à être implémentées sur des systèmes embarqués du fait de la compression des informations apporté par ce prétraitement. Les images résultantes de ce traitement ont ensuite été exposées à un modèle de réseau CNN qui a été entrainé pour effectuer les prédictions sur la nature du défaut sur les machines de production industrielle.

Les signaux vibratoires des roulements à billes issues de la base de données Case Westerne ont été utilisés pour vérifier la validité de notre système de détection des défauts. Les tests ont apporté des résultats concluants en termes de justesse de classification, de ressources matériels utilisées et du temps d'exécution du traitement sur plateforme embarquée FPGA comme sur ordinateur.

On espère finalement que le système de détection des défauts que nous avons développé, puisse trouver un moyen d'être testé et exploité dans un environnement industriel réel. On prévoit alors d'implémenter le système réalisé dans un réseau de capteurs sans fils industriel (IWSN), où la carte PYNQ-Z2 jouera le rôle de base station. Les signaux reçus des différents capteurs positionnés à des emplacements stratégiques des machines pourront alors être traités pour prédire l'origine du défaut. La base station peut aussi être connectée à internet pour prévenir les techniciens d'un éventuel dysfonctionnement d'une machine du parc industriel. Ce système aura pour effet d'alléger la charge de travail sur les équipes de maintenance d'une part et de réduire les coûts d'entretien des machines d'autres parts.

Références

[1] AFNOR, « FD X 60-000: Maintenance industrielle - Fonction maintenance », *Association Française de Normalisation*, vol. 33, n° 0. p. 1-29, 2002, [En ligne]. Disponible sur: http://www.ehpadneuilly.com/cariboost_files/FDX_60-000.pdf.

[2] J. Héng, *Pratique de la maintenance préventive*. Paris, France: Dunod, L'usine nouvelle, 2002.

[3] F. Monchy et J.-P. Vernier, *Maintenance méthodes et organisations*, 3e éd. Paris, France: Dunod, L'usine nouvelle, 2010.

[4] H. Zheng, A. R. Paiva, et C. S. Gurciullo, « Advancing from Predictive Maintenance to Intelligent Maintenance with AI and IIoT », 2020, [En ligne]. Disponible sur: http://arxiv.org/abs/2009.00351.

[5] B. Samira, « Contribution à l'optimisation de la maintenance prédictive par l'utilisation du logiciel ONMITREND (System On Ligne) cas : Cimenterie Hamma Bouziane », UNIVERSITE DE MENTOURI DE CONSTANTINE, 2008.

[6] O. Djebili, « Contribution à la maintenance prédictive par analyse vibratoire des composants mécaniques tournants . Application aux butées à billes soumises à la fatigue de contact de roulement . », UNIVERSITE DE REIMS CHAMPAGNE ARDENNE ECOLE, 2013.

[7] P. C. Boulenger, Alain, *Surveillance des machines tournantes par analyse vibratoire*. Paris, France: Dunod, L'usine nouvelle, 2009.

[8] R. R. Marie Guy, « METHODES NUMERIQUES POUR LA CARACTERISATION VIBRATOIRE DE STRUCTURES COMPLEXES », *HAL*, p. 245, 2011, [En ligne]. Disponible sur: https://hal.archives-ouvertes.fr/tel-00598378/.

[9] B. Slimani, M. A. Baba ali, et M. Boumahdi, « Application du filtrage des signaux vibratoires pour le suivi des défauts de roulements », Yahia Fares Médéa, 2017.

[10] BELOUADAH Abdenaceur, « Amélioration de la fiabilité d'un système électromécanique par l'utilisation des opérations de la maintenance préventive », UNIVERSITE MOHAMED BOUDIAF DE M'SILA, 2016.

[11] S. Kojtych, « Introduction au traitement du signal et à l'analyse fréquentielle par

transformées de Fourier », *HAL*, p. 47, 2019, [En ligne]. Disponible sur: https://hal.archives-ouvertes.fr/hal-02193929.

[12] A. CEBAN, « Méthode globale de diagnostic des machines électriques », Université d'Artois, 2012.

[13] S. Kass, « Diagnostic vibratoire autonome des roulements », INSA de Lyon, 2020.

[14] S. K. (SKF), « Usure et fatigue de surface dans les roulements », 2019. https://evolution.skf.com/fr/usure-et-fatigue-de-surface-dans-les-roulements/ (consulté le juill. 08, 2021).

[15] F. Landolsi, « SURVEILLANCE DES MACHINES PAR ANALYSE VIBRATOIRE », in *COURS DE TECHNIQUES DE SURVEILLANCE PARTIE*, 2015, p. 11-20.

[16] Dr. Adrian Rosebrock, *Deep Learning for Computer Vision with Python: Starter Bundle*. [United States] : PyImageSearch, ©2017, 2017.

[17] L. Bergman et Y. Hoshen, « CLASSIFICATION-BASED ANOMALY DETECTION FOR GENERAL DATA », *Comput. Sci. Mach. Learn.*, 2020, [En ligne]. Disponible sur: https://arxiv.org/abs/2005.02359.

[18] A. Géron, *Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow*, vol. 53, n° 9. 2019.

[19] J. Nordby, « Environmental Sound Classification on Microcontrollers using Convolutional Neural Networks », 2019.

[20] Y. LeCun, Y. Bengio, et G. Hinton, « Deep learning », *Nat. 2015 5217553*, vol. 521, n° 7553, p. 436-444, mai 2015, doi: 10.1038/nature14539.

[21] K. E. Koech, « Softmax Activation Function — How It Actually Works », 2020. .

[22] S. Albawi, T. A. Mohammed, et S. Al-Zawi, « Understanding of a convolutional neural network », *Proc. 2017 Int. Conf. Eng. Technol. ICET 2017*, vol. 2018-Janua, p. 1-6, 2018, doi: 10.1109/ICEngTechnol.2017.8308186.

[23] Y. Lecun, « Object recongnition with gradient-based learning », *AT&T*, 1999, [En ligne]. Disponible sur: http://yann.lecun.com/exdb/publis/pdf/lecun-99.pdf.

[24] J. Brownlee, « Softmax Activation Function with Python », 2020. https://machinelearningmastery.com/softmax-activation-function-with-python/

(consulté le juill. 08, 2021).

[25] J. Wu, « Sensor data-driven bearing fault diagnosis based on deep convolutional neural networks and s-transform », *Sensors (Switzerland)*, vol. 19, n° 12, 2019, doi: 10.3390/s19122750.

[26] J. Wang, Z. Mo, H. Zhang, et Q. Miao, « A deep learning method for bearing fault diagnosis based on time-frequency image », *IEEE Access*, vol. 7, n° c, p. 42373-42383, 2019, doi: 10.1109/ACCESS.2019.2907131.

[27] J. Antoni, « Fast Kurtogram », 2015. https://www.mathworks.com/matlabcentral/fileexchange/48912-fast-kurtogram.

[28] Y. Toumi, S. Lachenani, et M. Ould Zmirli, « analysis and artificial neural network », 2019, doi: 10.1109/ICAEE47123.2019.9015123.

[29] X. Li, L. Girin, S. Gannot, et R. Horaud, « Multichannel Online Dereverberation Based on Spectral Magnitude Inverse Filtering », *IEEE/ACM Trans. Audio Speech Lang. Process.*, vol. 27, n° 9, p. 1365-1377, 2019, doi: 10.1109/TASLP.2019.2919183.

[30] J. Li, L. Deng, R. Haeb-Umbach, et Y. Gong, *Robust automatic speech recognition: A bridge to practical applications*. Massachusetts: Elsevier, 2015.

[31] M. Bellanger, « Traitement numérique du signal - Théorie et pratique », p. 465, 2006.

[32] U. Kamath, J. Liu, et J. Whitaker, *Deep Learning for NLP and Speech Recognition*. 2019.

[33] « case western reserve university bearing data center website ». https://csegroups.case.edu/bearingdatacenter/pages/welcome-case-western-reserve-university-bearing-data-center-website (consulté le juill. 08, 2021).

[34] P. Fuchs et P. Poulain, « Cours de Python », p. 87, 2011, [En ligne]. Disponible sur: https://python.sdv.univ-paris-diderot.fr/cours-python.pdf.

[35] « Why Tensorflow ». https://www.tensorflow.org/?hl=fr (consulté le juill. 08, 2021).

[36] Collaboratif, « Debuter avec keras ». https://www.actuia.com/keras/ (consulté le juill. 08, 2021).

[37] M. Portolan, « Conception d'un Système Embarque Sur et Sécurisé », *Hal*, 2007, [En ligne]. Disponible sur: https://tel.archives-ouvertes.fr/tel-00163980/document.

[38] L. Crockett, D. Northcote, C. Ramsay, F. Robinson, et B. Stewart, *Exploring Zynq*

MPSoC. Glasgow: Xilinx, 2019.

[39] Xilinx, « AXI Reference Guide [UG761 v13.1] », vol. 761, p. 1-80, 2011, [En ligne]. Disponible sur: https://www.xilinx.com/support/documentation/ip_documentation/ug761_axi_reference_guide.pdf.

[40] « 28 Jupyter Notebook Shortcuts », 2016. https://www.dataquest.io/blog/jupyter-notebook-tips-tricks-shortcuts/ (consulté le juill. 08, 2021).

[41] TUL, « TUL PYNQ-Z2 », 2019. https://www.tul.com.tw/productspynq-z2.html (consulté le juill. 08, 2021).

[42] Intel, « Intel Neural Compute Stick 2 ». https://software.intel.com/content/www/us/en/develop/hardware/neural-compute-stick.html (consulté le juill. 08, 2021).

[43] Intel, « Intel Model Optimizer ». https://docs.openvinotoolkit.org/2021.2/openvino_docs_MO_DG_Deep_Learning_Model_Optimizer_DevGuide.html (consulté le juill. 08, 2021).

[44] R. Kastner, J. Matai, et S. Neuendroffer, *Parallel Programming For FPGA The HLS Book*. 2018.

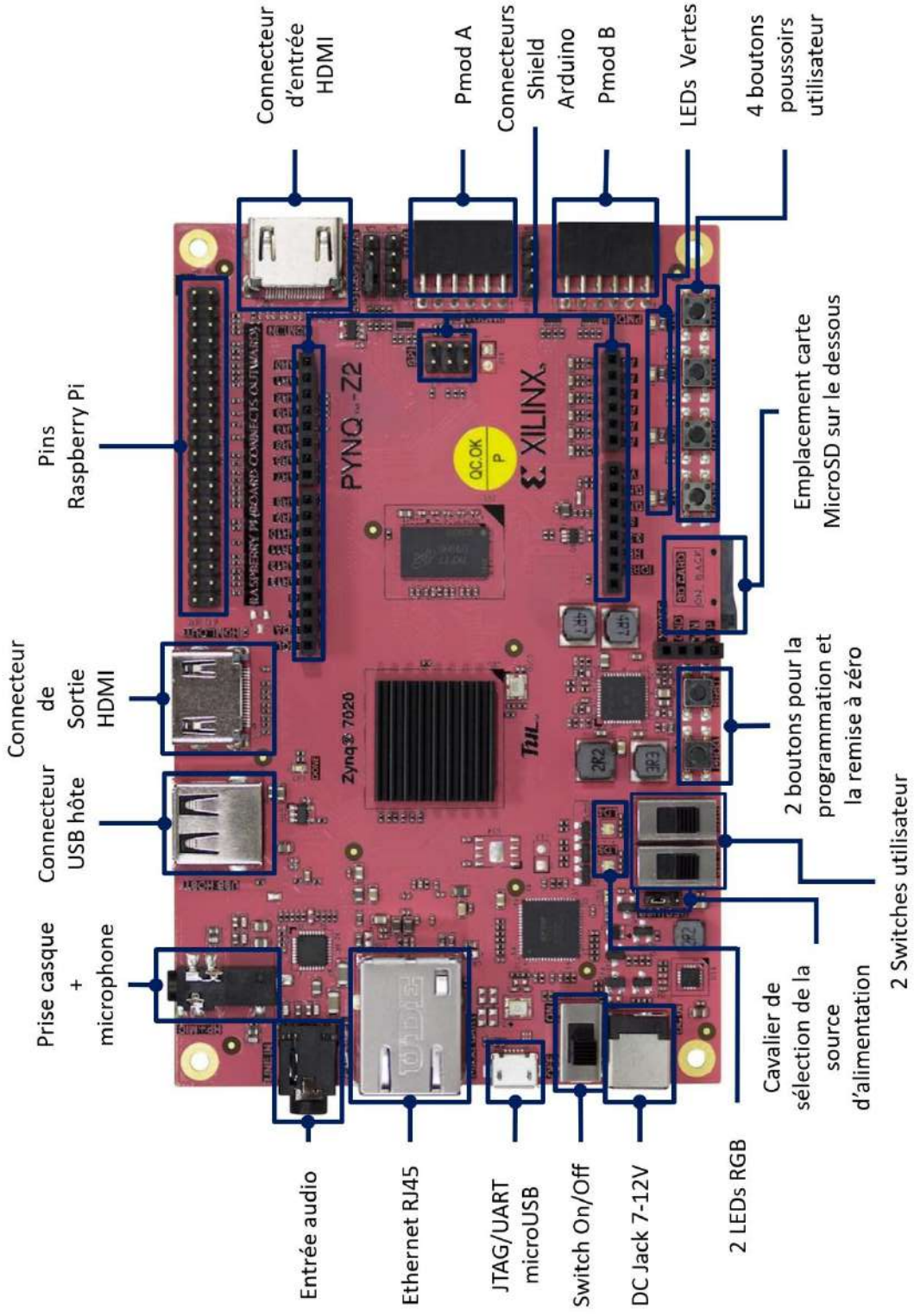
[45] M. Ravi, A. Sewa, S. T.g., et S. S. S. Sanagapati, « FPGA as a Hardware Accelerator for Computation Intensive Maximum Likelihood Expectation Maximization Medical Image Reconstruction Algorithm », *IEEE Access*, vol. 7, p. 111727-111735, 2019, doi: 10.1109/ACCESS.2019.2932647.

[46] STmicro, « STM32L433CC ». https://www.st.com/en/microcontrollers-microprocessors/stm32l433cc.html (consulté le juill. 08, 2021).

Annexes

Annexe A

Interfaces d’entrées/sorties de la carte PYNQ-Z2



Annexe B

Spécifications de la carte FPGA PYNQ-Z2

Zynq XC7Z020-1 CLG400C

- Processeur dual-core Cortex-A9 (650MHz)
- Mémoire DDR3 avec 8 contrôleurs DMA et 4 interfaces esclaves AXI3 hautes performances
- Contrôleurs périphériques haut débits : Ethernet 1Gb, USB 2.0, SDIO
- Contrôleurs bus série : SPI, UART, CAN, I²C
- Carte programmable depuit l'interface JTAG, mémoire flash Quad-SPI ou carte mémoire MicroSD
- Partie FPGA equivalente à Artix-7
    1. 13300 slices de 6 LUTs et 8 Flip-Flops chacun
    2. BRAM 630KB
    3. 4 domaines d'horloge avec une PLL et un contrôleur d'horloge mixte pour chaque domaine
    4. 220 DSP slices
    5. Convertisseur A/N intégré (XADC)

Mémoires

- 512MB DDR3 à 1050Mbps
- 16MB de mémoire flash Quad-SPI
- Emplacement pour carte MicroSD

Alimentation

- Alimentation par USB ou alimentation Externe 7-12V DC

Audio et vidéo

- Une entrée et une sortie HDMI
- Interface I²S avec un DAC de 24bits et un connecteur jack 3.5mm

Switches, boutons et LEDs

- 4 boutons poussoires
- 2 switches
- 4 LEDs vertes
- 2 LEDs RGB

Pins externes

- 2 connecteurs PMOD (dont 8 pins qui sont partagées avec les connecteurs compatibles Raspberry Pi)
- Connecteurs Shields Arduino
  (24 E/S dont 6 pins pour le XADC 0-3.3V)
- Connecteurs Raspberry Pi
  28 pins au total (dont 8 partagées avec le PMOD A).