Authentification par empreinte radio pour l'IoT

Louis Morge-Rollet

COLLEGES BRETAGNE LOIRE
SCIENCES POUR L'INGENIEUR ET LE NUMERIQUE

ENSTA BRETAGNE

THÈSE DE DOCTORAT DE

L'ÉCOLE NATIONALE SUPÉRIEURE
DE TECHNIQUES AVANCÉES BRETAGNE

ÉCOLE DOCTORALE N° 648
Sciences pour l'Ingénieur et le Numérique
Spécialité : *Télécommunications*

Par
Louis MORGE-ROLLET

Authentification par empreinte radio pour l'IoT

Thèse présentée et soutenue à Brest, le 28 septembre 2023
Unité de recherche : Lab-STICC, pôle T2I3, équipe SI3

Rapporteurs avant soutenance :

Abdel-Ouahab BOUDRAA	Professeur des universités, École navale
Arsenia CHORTI	Professeur des universités, ENSEA

Composition du Jury :

Président :	Jean-Pierre CANCES	Professeur des universités, Universités de Limoges
Examinateurs :	Abdel-Ouahab BOUDRAA	Professeur des universités, École navale
	Abdelmalek TOUMI	Enseignant-chercheur, ENSTA Bretagne
Dir. de thèse :	Roland GAUTIER	Professeur des Universités, UBO
Enc. de thèse :	Frédéric LE ROY	Enseignant-chercheur, ENSTA-Bretagne

Invité(s) :

Denis LE JEUNE	Chercheur, Ministère des Armées
Charles CANAFF	Ingénieur de recherche, ENSTA Bretagne

Remerciements

Je tiens tout d'abord à remercier mon directeur de thèse, Roland Gautier, pour m'avoir fait confiance dans cette thèse et de m'avoir guidé tout au long de ces années. Je tiens aussi à remercier chaleureusement mon encadrant, Frédéric Le Roy, qui a été présent au quotidien pour m'aider dans la réalisation de mes travaux. J'ai beaucoup apprécié tes conseils et ta disponibilité ainsi que nos longues discussions sur divers sujets. De plus, je tiens à remercier Denis Le Jeune, qui été initialement encadrant de ma thèse, mais qui a malheureusement dû quitter l'ENSTA Bretagne pendant ma thèse. Tout comme pour Frédéric, j'ai beaucoup apprécié tes conseils et ton expertise. Enfin, je tenais à remercier Charles Canaff qui a pris le rôle d'encadrant (bénévole) lors du départ de Denis. De manière générale, je tiens à vous remercier, vous quatre, pour l'encadrement de cette thèse.

J'adresse aussi mes remerciements aux membres de mon comité de suivi individuel, Stéphane Azou et Jean-Jacques Szkolnik, pour leurs conseils. De plus, je tiens à remercier les membres de jury de mon doctorat. D'une part, je souhaite remercier les rapporteurs de mon manuscrit, Abdel-Ouahab Boudraa et Arsenia Chorti, pour avoir accepté de relire mon manuscrit, ainsi que pour leurs retours. D'autre part, je voudrais remercier les examinateurs de ma soutenance, Jean-Pierre Cances et Abdelmalek Toumi, pour leurs questions et leurs remarques au cours de la soutenance.

D'une manière plus générale, je souhaite remercier mes collègues de l'ENSTA Bretagne. Plus particulièrement, je tiens à remercier mes «co-bureaux», Grégoire De Broglie et Maëlic Louart, pour m'avoir supporté durant toutes ces années. De plus, je souhaiterais remercier mes collègues de l'équipe Circuits et Systèmes (CS), de l'équipe Systèmes Logiciels et Sécurité (SLS) et de l'équipe Mathématiques, Algorithmes et Décision (MAD).

Enfin, je souhaite remercier ma famille et mes amis pour leur soutien indéfectibles pendant toutes les années de mon doctorant. Merci d'avoir été présent durant tous ces moments, qu'ils soient bons ou mauvais.

Je terminerai ces remerciements par ces mots : merci à tous ceux que j'ai cités, mais aussi à ceux que j'aurais pu oublier.

Sommaire

TABLE DES FIGURES

Liste des tableaux

Acronymes

ADS-B	*Automatic Depedant Surveillance-Broadcast.*
CAN	Convertisseurs Analogique-Numérique.
CNA	Convertisseurs Numérique-Analogique.
CNN	réseaux de neurones convolutifs ou *Convolutive Neural Networks.*
CPU	processeur ou *Central Processing Unit.*
CSI	*Channel State Indicator.*
FI	avec fréquence intermédiaire.
FI0	sans fréquence intermédiaire.
FPGA	*Field Programmable Gate Array.*
FSL	apprentissage frugal ou *Few-Shot Learning.*
GMM	mélange de gaussiennes ou *Gaussian Mixture Model.*
GNSS	système de positionnement par satellites ou *Global Navigation Satellite Systems.*
GPU	processeur graphique ou *Graphical Processing.*
I/Q	en phase et en quadrature ou *In-Phase and Quadrature.*
II	*Initiation Interval.*
IMU	Centrale intertielle ou *Inertial Measurement Unit.*
IoT	*Internet of Things.*
LNA	amplificateur à faible bruit ou *Low Noise Amplifier.*

M-PSK	*M-ary Phase Shift Keying*.
MRSE	*Mean Relative Square Error*.
OL	Oscilatteur Local.
PA	amplificateur de puissance ou *Power Amplifier*.
PEA-SD	*Parameters Estimation Algorithm using Structure Dictionary*.
PLA	authentification sur la couche physique ou *Physical Layer Authentication*.
QPSK	modulation en quadrature de phase ou *Quadrature Phase Shift Keying*.
RF	radiofréquence.
RFF	authentification par empreinte radio ou *Radio Frequency Fingerprinting*.
RFMLS	*Radio Frequency Machine Learning Systems*.
RP	Réentrainabilié Partielle.
RSS	*Received Signal Strength*.
SDR	radio logicielle ou *Software-Defined Radio*.
SNR	rapport signal à bruit ou *Signal-to-Noise Ratio*.
SoC	système sur puce ou *System on Chip*.

INTRODUCTION

Le sujet de cette thèse porte sur des méthodes d'authentification non cryptographique, exploitant les imperfections des composants radiofréquence (RF) de l'émetteur, voire du canal de propagation, qui sont considérées comme uniques. L'objectif est alors d'améliorer la sécurité et, notamment, l'authenticité des sources des messages, en exploitant ces imperfections dans le contexte de l'internet des objets ou *Internet of Things* (IoT) [1].

Contexte

Contexte général

Les différents composants d'un émetteur RF peuvent présenter des imperfections dues aux variabilités présentes lors du processus de fabrication et d'assemblage des composants RF [1]. De plus, l'environnement de propagation entre l'émetteur et le récepteur présente de nombreuses imperfections (ou variations), notamment dues aux multiples réflexions de l'onde transmise sur différents éléments de l'environnement ainsi qu'à l'effet Doppler.

L'empreinte radio ou *RF Fingerprint* correspond donc à l'impact de ces imperfections sur le signal transmis. Plus particulièrement, ces imperfections créent des distorsions sur le signal transmis par l'émetteur qui sont considérées comme uniques pour chaque émetteur [1] et agissant comme une empreinte nous permettant de l'authentifier, comme en biométrie. Ainsi, les méthodes d'authentification par empreinte radio ou *Radio Frequency Fingerprinting* (RFF), sur lesquelles porte ce manuscrit, cherchent à exploiter cette empreinte radio pour augmenter la sécurité de l'IoT.

La définition de l'IoT que nous utiliserons dans ce manuscrit est celle d'Oracle [2] : «L'Internet of Things (IoT) décrit le réseau de terminaux physiques, les objets, qui intègrent des capteurs, des logiciels et d'autres technologies en vue de se connecter à d'autres terminaux et systèmes sur Internet et d'échanger des données avec eux». Cependant, les objets ne se limitent pas seulement aux capteurs, mais prennent aussi en compte les

1. Dans ce manuscrit, nous emploierons le terme *Internet of Things* (IoT), à la place d'internet des objets, à cause sa plus grande utilisation dans la littérature.

actionneurs, ainsi que les objets intégrant à la fois des capteurs et des actionneurs [3].

Les applications de l'IoT dans la société sont relativement importantes, allant des technologies portables (*wearables*) aux réseaux et villes intelligentes, en passant par les habitats connectés. En 2022, le nombre d'objets connectés à l'IoT était estimé autour de 13.2 milliards selon le rapport *Ericsson Mobility Report* [4]. Outre leur grand nombre, auquel on se référera dans la suite du manuscrit comme l'échelle de l'IoT, d'autres caractéristiques de l'IoT sont à prendre en compte comme l'interconnectivité, l'hétérogénéité ou la dynamicité [5]. Ces caractéristiques seront notamment réutilisées dans la suite de ce manuscrit pour définir les propriétés nécessaires aux méthodes de RFF pour l'IoT.

Une des différences majeures entre internet «classique» et l'IoT est la possibilité d'interagir avec le monde physique, à l'aide de différents capteurs et actionneurs. Bien que l'IoT vise à simplifier notre vie quotidienne, il augmente aussi le risque de cybermenaces pouvant avoir une influence directe dans le monde réel. Ainsi, un effort important doit être déployé pour s'assurer de la sécurité de ces réseaux. Dans ce manuscrit, nous nous focaliserons sur la couche physique des protocoles sans fil IoT. Cependant, bien que l'authentification par empreinte radio agisse au niveau de la couche physique, il sera parfois nécessaire d'avoir accès à des informations provenant des couches supérieures, comme le format du préambule ou à l'identifiant de l'émetteur d'un message défini au niveau de la couche liaison.

Cadre des travaux

Les travaux présentés dans ce manuscrit ont été réalisés au sein de l'ENSTA Bretagne, en lien avec l'Université de Bretage Occidental (UBO) de Brest. De plus, certains travaux ont été intégrés dans la chaire CyberIoT, ainsi que dans les projets AN DRO et DISPEED décrits dans les paragraphes suivants.

La chaire CyberIoT, soutenue par l'Institut Brestois du Numérique et des Mathématiques (IBNM) au sein de l'UBO, est portée par le Pr. Roland Gautier et a pour thème : «la sécurité de la couche physique - un enjeu indispensable pour la démocratisation de l'Internet des Objets». Plus particulièrement, cette chaire a pour but de renforcer les échanges entre le Lab-STICC et le Laboratoire de Mathématiques de Bretagne Atlantique (LMBA) dans le domaine de la cybersécurité de la couche physique des communications sans fil. Ainsi, l'intégralité des publications présentées dans cette thèse ont été financées à l'aide de cette chaire.

Le projet Analyse Numérique de Signaux de DROnes (AN DRO) est un projet financé

par le fond européen de développement régional (FEDER), qui réunit quatre partenaires : Syrlinks, Elliptica, l'ENSTA Bretagne et l'UBO. Le projet, qui a pris fin en octobre 2022, a été supervisé par le Pr. Roland Gautier pour la partie UBO et par M. Frédéric Le Roy pour la partie ENSTA Bretagne, en collaboration avec M. Denis Le Jeune et M. Charles Canaff. Ce projet avait pour but de développer une solution innovante qui permettrait de détecter au plus tôt, d'identifier et de localiser des drones de différents types, issus du commerce ou réalisés par des tiers à des fins malfaisantes. Plus particulièrement, le but des chercheurs de l'UBO et de l'ENSTA Bretagne ayant pris part à ce projet a été de développer des méthodes de détection et d'identification du type de drones, que ce soit à l'aide de méthodes expertes ou en utilisant de l'apprentissage automatique. Cette collaboration a fait l'objet de plusieurs publications, dont l'une d'elle est présentée en section 2.2 : la modulation Doppler en radiocommunications.

Le projet Détection d'Intrusion, compromis Sécurité/Performance/Énergie, Etude pour les meutes de Drones (DISPEED) est un projet financé par l'Agence de l'Innovation de la Défense (AID) et supervisé par Pr. Jalil Boukhobza, de l'ENSTA Bretagne. L'objectif de ce projet est de proposer un modèle et des stratégies pour l'exécution des systèmes de détection d'intrusion embarqués sur des architectures hétérogènes et qui fournissent un compromis pertinent entre niveau de sécurité, rapidité d'exécution et consommation énergétique. Le doctorant et son encadrant M. Frédéric Le Roy ont pris part au projet en octobre 2022, en travaillant notamment sur les aspects d'implémentation de la phase d'inférence d'algorithmes d'apprentissage automatique sur processeur ou *Central Processing Unit* (CPU) et sur *Field Programmable Gate Array* (FPGA).

Problématique

La problématique de ce manuscrit est la suivante :

«Quelles sont les propriétés nécessaires à l'utilisation du RFF dans un contexte IoT et quelles sont les méthodes (et leurs implémentations) qui en découlent ?»

Au travers de cette problématique, nous nous intéresserons tous d'abord aux propriétés que doivent respecter les méthodes de RFF dans un contexte IoT, que nous appellerons par la suite : les propriétés d'utilisabilité. Comme nous venons de l'évoquer, l'IoT présente des caractéristiques spécifiques comme son échelle, l'hétérogénéité de ses protocoles ou la dynamicité des appareils qui la composent. Ainsi, il semble nécessaire de proposer des propriétés d'utilisabilité dépendantes de ses aspects.

Dans un second temps, notre problématique nous amène à concevoir des méthodes de RFF adaptées au contexte de l'IoT. Généralement, les méthodes de RFF présentes dans la littérature n'ont pas été proposées pour prendre en compte les spécificités de l'IoT, car elles sont conçues pour des réseaux sans fil «classiques». Afin de prendre en compte ces spécificités, il est donc nécessaire de proposer de nouvelles méthodes dédiées au contexte de l'IoT.

Enfin, la problématique de ce manuscrit pose la question des implémentations possibles de ces méthodes. Le domaine du RFF étant proche du traitement du signal, des radiocommunications et de l'apprentissage automatiques, l'aspect implémentation a été peu abordé dans la littérature. Cependant, il s'agit d'un aspect essentiel, qu'il semble nécessaire d'étudier pour l'adoption massive du RFF dans le contexte de la sécurité de l'IoT.

Contributions

La première contribution de cette thèse, présentée en section 3.1, porte sur la proposition de trois propriétés d'utilisalibité : l'adaptabilité, la scalabilité et la complexité. Ces trois propriétés sont issues d'une analyse des caractéristiques et des défis de l'IoT, ainsi que de l'état de l'art des méthodes de RFF pour l'IoT.

La seconde contribution, présentée en section 3.2, porte sur une méthode de RFF basée sur l'utilisation de réseaux siamois [6], [7] pour le RFF. Plus particulièrement, nous avons évalué différentes manières d'entraîner un réseau siamois, ainsi que différentes architectures de réseaux de neurones. Cette contribution se focalise sur les propriétés d'adaptabilité et de scalabilité, même si nous avons aussi évalué la complexité des réseaux siamois pour le RFF.

La troisième contribution, présentée en section 3.3, porte sur une méthode de RFF appelée *RF eigenfingerprints* et inspirée des travaux en reconnaissance faciale sur les *eigenfaces* [8], [9]. À l'inverse de la contribution précédente, cette contribution se focalise majoritairement sur les propriétés d'adaptabilité et de complexité, même si nous avons aussi évalué la scalabilité de cette méthode.

La quatrième contribution, présentée en section 3.4.2, décrit différents cas d'applications du RFF dans un contexte IoT, que nous appellerons configurations : configuration de surveillance, configuration asymétrique et configuration symétrique. Elles exploitent notamment les différents compromis des méthodes proposées pour s'adapter au mieux

aux différents cas d'applications du RFF pour l'IoT.

La cinquième contribution de ce manuscrit, présentée dans le chapitre 4, traite des implémentations logicielles et matérielles des deux méthodes de RFF proposées dans le chapitre 3. Tout d'abord, nous définissons les métriques et les valeurs associées pour évaluer les différentes implémentations, puis nous proposons pour chacune des méthodes plusieurs implémentations logicielles sur CPU et matérielles sur FPGA.

Enfin, la dernière contribution, englobant l'ensemble des travaux présentés dans le chapitre 2, porte sur la modélisation et l'estimation des imperfections dans une chaîne de transmission radio. Ces différents travaux seront notamment réutilisés comme des outils dans les autres contributions, ainsi que plus généralement dans le manuscrit.

Plan de thèse

Le plan de thèse est le suivant :

— Le chapitre 1 présente l'état de l'art, en commençant par le domaine du RFF de manière globale, pour ensuite se focaliser sur les méthodes de RFF pour l'IoT, et notamment, sur les propriétés évoquées dans la littérature. De plus, nous présenterons différentes publications liées à la modélisation d'imperfections en RFF, ainsi qu'à l'implémentation logicielle et matérielle des méthodes de RFF.

— Le chapitre 2 présente nos apports à la modélisation et l'estimation des imperfections dans une chaîne de transmission radio. Nous avons choisi de découper ce chapitre en trois sous-parties distinctes : l'émetteur, le canal de propagation et le récepteur.

— Le chapitre 3 est le chapitre principal de ce manuscrit, puisqu'il regroupe les quatre premières contributions précédemment évoquées. Nous commençons par proposer trois propriétés d'utilisabilité, puis nous présentons nos travaux sur les réseaux siamois pour le RFF et les *RF eigenfingerprints* et nous finissons par une comparaison des méthodes proposées, ainsi qu'une proposition de différents cas d'applications du RFF dans un réseau IoT.

— Le chapitre 4 présente, tout d'abord, les métriques nécessaires à l'évaluation des implémentations logicielles et matérielles. Ensuite, nous proposons pour chaque méthode, plusieurs implémentations sur CPU et FPGA.

— Le chapitre 5 présente les perspectives de ce manuscrit, en s'intéressant d'abord aux perspectives de nos travaux, puis en élargissant à des perspectives plus globales.

Chapitre 1

État de l'art

Dans un premier temps, le domaine de l'authentification par empreinte radio ou *Radio Frequency Fingerprinting* (RFF) sera introduit. Ensuite, nous présenterons les travaux de la littérature en authentification par empreinte radio dans le cadre de l'*Internet of Things* (IoT), ainsi que d'autres sujets connexes nécessaires à ce manuscrit comme la modélisation des imperfections pour le RFF.

1.1 Authentification par empreinte radio (RFF)

La définition que nous donnerons de l'authentification par empreinte radio est la suivante : les techniques permettant d'identifier un émetteur radio grâce aux imperfections de ses composants ainsi que de son environnement de propagation. Le terme anglais habituel pour authentification par empreinte radio est *Radio Frequency Fingerprinting* (RFF), parfois abrégé par *RF Fingerprinting*. Il existe également d'autres termes comme *Physical Layer Identification* [10] et *RF-DNA* [11] entre autres.

Le but du RFF est d'exploiter l'empreinte radio d'un émetteur sans fil, appelée *RF Fingerprint* [12]-[14], pour l'authentifier. Cependant, dans la littérature, le terme *RF Fingerprint* (ou même *fingerprint*) peut aussi faire référence aux descripteurs [1] utilisés pour réaliser l'authentification [15], [16], aussi appelés *RF Fingerprint features* [13], [17]. Dans ce manuscrit, nous utiliserons le terme empreinte radio pour faire référence à l'effet des imperfections sur le signal émis et nous emploierons plutôt le terme signature (voire empreinte) pour parler des descripteurs extraits à partir de l'empreinte radio.

D'une part, le RFF fait partie des techniques de sécurité au niveau de la couche physique [18], [19] qui cherchent à assurer la sécurité des communications en travaillant au niveau de la couche physique et, plus particulièrement, de l'authentification sur la couche physique ou *Physical Layer Authentication* (PLA) [20], [21]. D'autre part, le RFF fait

1. Dans ce manuscrit, le terme descripteurs ou *features* en anglais correspond à un ensemble de variables que l'on va utiliser pour caractériser le signal à analyser.

partie des méthodes d'authentification non cryptographique sans fil [22], aussi appelées *device fingerprinting* [23], qui exploitent les spécificités d'un émetteur sans fil pour l'authentifier. Ainsi, les méthodes de RFF peuvent être vues comme différentes des approches d'authentification cryptographique classiquement utilisées dans les réseaux sans fil IoT [24], [25], car elles exploitent l'empreinte radio au niveau de la couche physique.

1.1.1 Architecture d'un système RFF

Avant toute chose, il est nécessaire d'évoquer la notion de biométrie, sur laquelle l'authentification par empreinte radio s'appuie partiellement [15]. La biométrie se focalise sur «ce que l'on est», ou facteur d'inhérence (*inherence factor*). À la différence d'autres approches d'authentification comme la cryptographie, la biométrie ne donne pas une authentification exacte (comme pour un mot de passe), car elle peut être sujette à des erreurs [26], pouvant amener à des faux positifs comme des faux négatifs.

Un système biométrique est décrit à la figure 1.1 et il est composé d'un capteur biométrique, d'un module d'extraction d'empreinte, d'un module dédié à la correspondance d'empreintes ainsi que d'une base contenant les empreintes des utilisateurs légitimes [27]. De plus, il est composé de deux sous-systèmes :

— Sous-système d'enrôlement : permet d'ajouter l'empreinte d'un nouvel utilisateur légitime à la base d'empreintes.
— Sous-système d'authentification : permet d'authentifier un utilisateur grâce à la base d'empreintes.

Enfin, il existe deux moyens permettant d'authentifier un utilisateur :

— L'identification : permet d'authentifier un utilisateur en comparant son empreinte avec l'ensemble des empreintes de la base d'empreintes.
— La vérification : permet d'authentifier un utilisateur en comparant son empreinte avec l'empreinte d'un utilisateur légitime de la base d'empreintes. En effet, l'utilisateur peut, par exemple, fournir un identifiant permettant de le comparer à un utilisateur légitime spécifique de la base d'empreintes.

Dans le cas du RFF, un système d'authentification par empreinte radio, peut être décrit par la figure 1.2 et il est composé de plusieurs éléments [15] :

— Une antenne (voir de plusieurs),
— Un module d'acquisition du signal,
— Un module d'extraction d'empreintes,
— Une base de données d'empreintes,

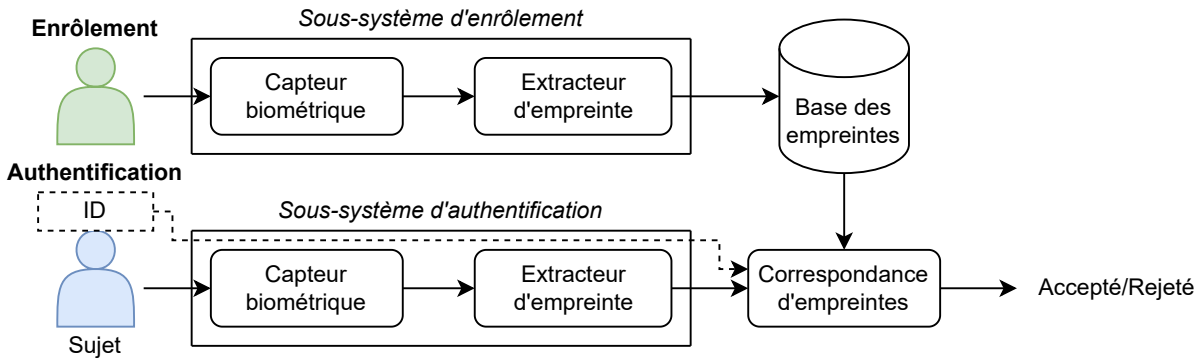


FIGURE 1.1 – Principe de fonctionnement d'un système biométrique

— Un module de correspondance d'empreintes.

De plus, comme dans la figure 1.1, il peut être décomposé en deux sous-systèmes [15] : un sous-système d'enrôlement et un sous-système d'authentification. Enfin, tout comme pour les systèmes biométriques, il est possible lors de l'authentification de réaliser, à partir du signal reçu, soit l'identification d'un émetteur, soit sa vérification (par exemple en exploitant son identifiant défini à la couche liaison, voir dans les couches supérieures).

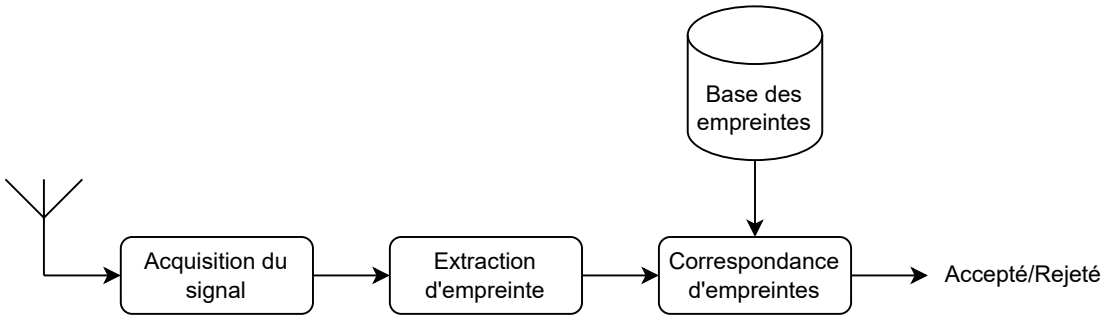


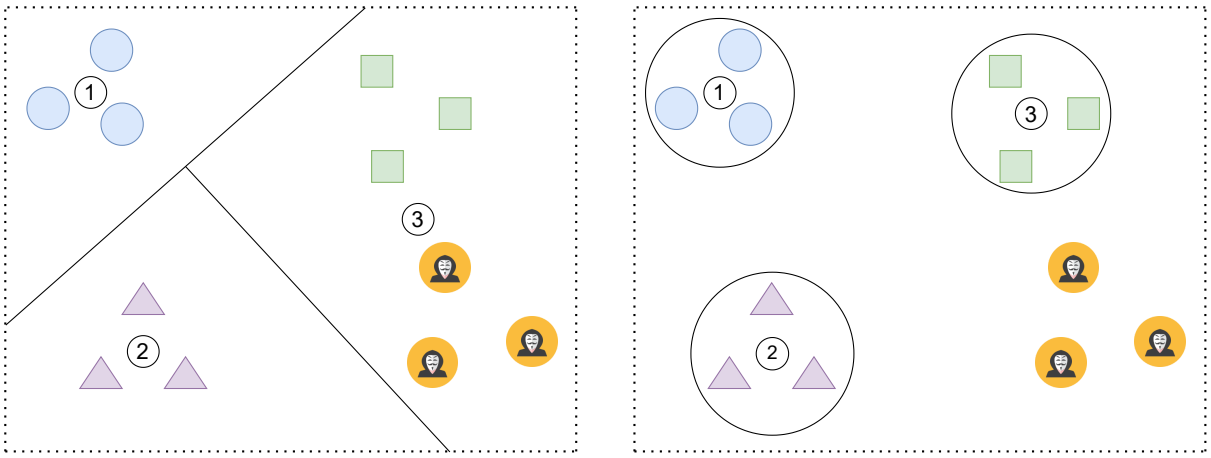
FIGURE 1.2 – Système d'authentification par empreinte radio

Cette vision biométrique du RFF n'en est qu'une parmi tant d'autres, comme nous le montrerons dans la suite de l'état de l'art. En effet, ces dernières années, beaucoup d'auteurs considèrent le RFF comme une tâche d'apprentissage automatique, généralement une tâche de classification en ayant recours à de l'apprentissage profond [28]-[30]. Dans ce manuscrit, nous défendrons une vision plutôt proche de la biométrie, même si nous aurons recours à l'apprentissage automatique ainsi qu'à l'apprentissage profond dans certaines de nos contributions.

1.1.2 Les hypothèses «monde fermé»/«monde ouvert»

Avant de présenter les propriétés du RFF ainsi que ses différents cas d'utilisation, il est nécessaire de clarifier la notion d'hypothèse «monde fermé» et d'hypothèse «monde ouvert».

En effet, le schéma de principe présenté en figure 1.2 est inspiré des systèmes biométriques et, par conséquent, il est basé sur une hypothèse dite «monde ouvert» [27]. Cette hypothèse, dans le cas du RFF, suppose qu'il est possible d'évaluer des signaux provenant d'émetteurs qui ne font pas partie de la base d'empreintes. À l'inverse, l'hypothèse «monde fermé» suppose que les signaux que l'on cherche à identifier appartiennent forcément à un émetteur de la base d'empreintes, que l'on appellera dans la suite du manuscrit : les émetteurs légitimes. Cette dernière hypothèse est acceptable pour un système de classification de chiffres manuscrits comme avec le jeu de données MNIST [31], mais elle l'est beaucoup moins pour un système RFF, puisqu'on ne peut pas connaître tous les émetteurs.



(a) Hypothèse «monde fermé» (b) Hypothèse «monde ouvert»

FIGURE 1.3 – Comparaison entre l'hypothèse «monde fermé» et «monde ouvert»

Comme nous venons de l'évoquer, beaucoup de méthodes de RFF sont basées sur des algorithmes de classification utilisés en apprentissage automatique, appelés classifieurs dans la suite de ce manuscrit et reposant sur une hypothèse «monde fermé» [1], [12], [28]. Selon Gerdes [32], l'utilisation de ce type de méthode peut poser un problème lorsqu'un système RFF est en présence d'un émetteur illégitime (par opposition aux émetteurs légitimes). En effet, comme le montre la figure 1.3, dans le cas d'une hypothèse «monde fermé», les méthodes de RFF vont attribuer l'identité d'un émetteur légitime à un émetteur illégitime, ce qui n'est pas le cas des méthodes basées sur l'hypothèse «monde ouvert».

Dans ce manuscrit, les méthodes proposées prennent en compte l'hypothèse «monde ouvert», car elles permettent de mieux appréhender la problématique des émetteurs illégitimes. Pour cela, nous utiliserons soit des méthodes basées sur la structure présentée en figure 1.2, soit comportant une phase de détection d'anomalie avant la classification et permettant de détecter des empreintes radio inconnues comme dans [33], [34].

1.1.3 Les propriétés de l'empreinte radio

Le module d'extraction d'empreintes nous permet d'obtenir une signature composée d'un ensemble de descripteurs à partir du signal reçu. Ainsi, ces descripteurs doivent respecter certaines propriétés nécessaires au bon fonctionnement du système d'authentification [14], [15], [27] :

- L'universalité : il est possible d'extraire les mêmes descripteurs pour tous les appareils.
- L'unicité : deux appareils ne peuvent pas avoir la même empreinte.
- La permanence : l'empreinte doit être invariante dans le temps.
- La collectabilité : il est possible d'extraire ces descripteurs à partir du signal reçu.

Il existe aussi d'autres propriétés qui ont été proposées dans la littérature :

- La robustesse : l'empreinte doit être robuste, ou du moins être évaluée par rapport aux variations environnementales externes comme les interférences avec d'autres signaux ou aux variations internes de l'appareil comme la température [14], [15].
- L'acceptabilité : en biométrie, cette propriété consiste à considérer que les descripteurs extraits ne sont pas sujets à des objections de la part de la population [27].

En plus, de ces propriétés, il est important de noter que les méthodes de RFF sont majoritairement passives, contrairement à d'autres méthodes de PLA [20], [21]. En effet, les méthodes de RFF ne se contentent que d'écouter, puis de traiter, le signal envoyé par un appareil. De plus, les méthodes d'authentification par empreinte radio peuvent être considérées comme asymétriques, car l'émetteur n'a pas (ou peu) d'effort à fournir alors que le récepteur effectue un certain nombre de traitements pour extraire cette signature du signal reçu [13], [35].

1.1.4 Les cas d'utilisation du RFF

Historiquement, l'authentification par empreinte radio est rattachée au SEI (*Specific Emitter Identification*). Ce domaine est né dans le milieu des années 1960 pour identifier l'émetteur d'un signal, que ce soit pour les radars ou pour les émetteurs sans fils [36], [37]. À notre connaissance, une des premières références faisant mention au RFF est un rapport d'évaluation du Motron TxID-1 qui avait pour but de détecter les émetteurs illégitimes dans les réseaux mobiles aux États-Unis et au Canada [38]. Depuis, de nombreux cas d'application du RFF ont été proposés, notamment pour la sécurité [15]. Plus particulièrement, nous présenterons d'abord les cas d'utilisation liés à l'authentification et au contrôle d'accès, puis à la détection d'attaques et enfin les autres cas d'utilisation qui ne correspondent pas aux deux catégories précédentes.

1.1.4.1 L'authentification

Le premier cas d'utilisation du RFF que nous présentons concerne l'authentification et le contrôle d'accès dans les réseaux sans fil. En effet, dans les réseaux sans fil, des algorithmes d'authentification cryptographique permettant l'authentification des appareils sur le réseau. Cependant, il est possible pour un attaquant d'obtenir le matériel cryptographique d'un appareil (sa clé privée par exemple), lui permettant par la suite de s'authentifier de manière légitime sur le réseau. Ainsi, pour répondre à cette problématique, de nombreux auteurs proposent l'utilisation de l'authentification par empreinte radio pour détecter ce genre d'attaque [23]. Dans [1], Brik et al. mentionnent qu'il s'agit d'un second périmètre de sécurité permettant de détecter une intrusion dans le périmètre principal, qui est quant à lui basé sur des mécanismes cryptographiques. De plus, certains auteurs mentionnent qu'il serait possible d'utiliser le RFF à la place de mécanismes d'authentification cryptographique dans les contextes où la consommation énergétique est réellement problématique [15]. En effet, dans certaines configurations où les émetteurs ont des ressources restreintes, l'exploitation de l'asymétrie du RFF pourrait être profitable, car seul le récepteur consomme de l'énergie pour extraire les descripteurs à partir de l'empreinte radio [13]. Cependant, certains auteurs, comme Robyns et al. préconisent que le RFF devrait être utilisé comme un second facteur d'authentification [33], c'est-à-dire en plus des mécanismes d'authentification classiques.

Même si de nombreuses publications sur le RFF traitent de l'authentification dans les réseaux sans fil, il est aussi possible d'en utiliser pour les systèmes de diffusion sans

fil unidirectionnels. En effet, de nombreuses technologies radio sont unidirectionnelles comme le système de positionnement par satellites ou *Global Navigation Satellite Systems* (GNSS) ou dans les systèmes de positionnement comme le *Automatic Depedant Surveillance-Broadcast* (ADS-B) pour les avions. Dans [39], Sun et al. proposent l'utilisation de l'authentification par empreinte radio pour la détection d'usurpation d'un émetteur GNSS, plus particulièrement pour le système GPS (*Global Positioning System*). Dans [40], Jian et al. présentent les résultats d'une expérience massive comportant des milliers d'émetteurs ADS-B pour le RFF basé sur de l'apprentissage profond.

1.1.4.2 La détection d'attaques

L'authentification par empreinte radio a aussi été utilisée pour la détection de certaines attaques dans les réseaux sans fil. Dans [41], Nguyen et al. proposent d'utiliser le RFF pour la détection des attaques appelées *sybil*. Ces attaques consistent pour un attaquant à émettre plusieurs messages sous différentes identités à l'aide du même émetteur. Pour la détection de ces attaques, ils utilisent une approche d'apprentissage automatique non supervisé (partitionnement) pour détecter les différents émetteurs (appelés *clusters*) et déterminer si plusieurs identités sont reliées à un même émetteur. Ils mentionnent également que cette approche peut aussi être utilisée pour la détection d'attaques *masquerade* qui consiste à usurper l'identité d'un émetteur illégitime. Pour cela, il suffit de détecter si plusieurs *clusters* sont reliés à un unique identifiant. Dans [42], Rasmussen et al. proposent l'utilisation du RFF pour la détection d'attaques de type *wormhole*. Cette attaque consiste, pour un attaquant, à faire croire aux émetteurs d'un réseau sans fil qu'il existe un chemin plus rapide en utilisant deux nœuds corrompus reliés, possiblement par un lien rapide, pour créer un «tunnel». Cette attaque, menée contre le protocole de routage, a pour conséquence que l'unique route entre la source et la destination passe par le «tunnel», permettant ainsi à l'attaquant de filtrer les paquets non voulus ou de bloquer le trafic.

1.1.4.3 Autres cas d'application

Il existe d'autres cas d'application du RFF :

— La détection de clonage d'appareils sans fil : il est possible d'utiliser une méthode RFF pour la détection de clonage d'étiquette RFID [15], [43]. De plus, l'usage du RFF a aussi été évoqué dans [42] pour la détection de clonage de carte SIM.

— Désanonymisation du trafic : dans [15], [33], [42], les auteurs mentionnent que le RFF pourrait être utilisé pour la désanonymisation du trafic, car il est capable

d'associer chaque trame à son émetteur respectif grâce à son empreinte radio.
— Échange de clés : le RFF peut aussi être utilisé comme authentification pour le protocole d'échange de clés Diffie-Hellman en utilisant les caractéristiques uniques du canal [22].

Dans ce manuscrit, nos contributions s'intéressent principalement à assurer l'authenticité des sources de messages dans un réseau IoT, même s'il serait aussi possible de les utiliser pour de la détection d'intrusions ou d'attaques.

1.2 Taxonomie du RFF

Dans cette sous-section, nous proposons une taxonomie du RFF présentée en figure 1.4 et inspirée d'autres taxonomies de la littérature [1], [14], [41]. Il est important de noter que de nombreuses taxonomies ont été proposées, notamment sur des aspects orthogonaux aux aspects que nous allons présenter [17], [23], [28]. Par exemple, dans [28], Riyaz et al. proposent une taxonomie basée sur le type d'apprentissage, en faisant la distinction entre deux types d'apprentissage : l'apprentissage supervisé et l'apprentissage non supervisé.

Notre taxonomie du domaine de l'authentification par empreinte radio peut être décomposée en deux sous-domaines en fonction des descripteurs considérés : *location-based* et *radiometric* [1], [21], [23]. Les approches *location-based* exploitent des descripteurs basés sur le canal de propagation entre l'émetteur et le récepteur, dits *location-dependant features* [23]. À l'inverse, les approches dites *radiometric* exploitent des descripteurs basés sur les défauts des composants de l'émetteur, dits *location-independant features* [23]. Il est possible de retrouver des termes similaires en authentification passive sur la couche physique [21] : *channel-based PLA* et *device-based passive PLA*. Cependant, il est quand même possible de trouver des méthodes qui exploitent les deux types de descripteurs [21].

1.2.1 *Location-based*

Le sous domaine dit *location-based* correspond aux descripteurs dépendant du canal. Il est assez courant de découper ce sous-domaine en deux branches par rapport aux descripteurs considérés : *Received Signal Strength* (RSS) et *Channel State Indicator* (CSI). De plus, certains auteurs ajoutent un troisième type de descripteurs basés sur l'angle d'arrivée ou *AoA : Angle of Arrival* [28].

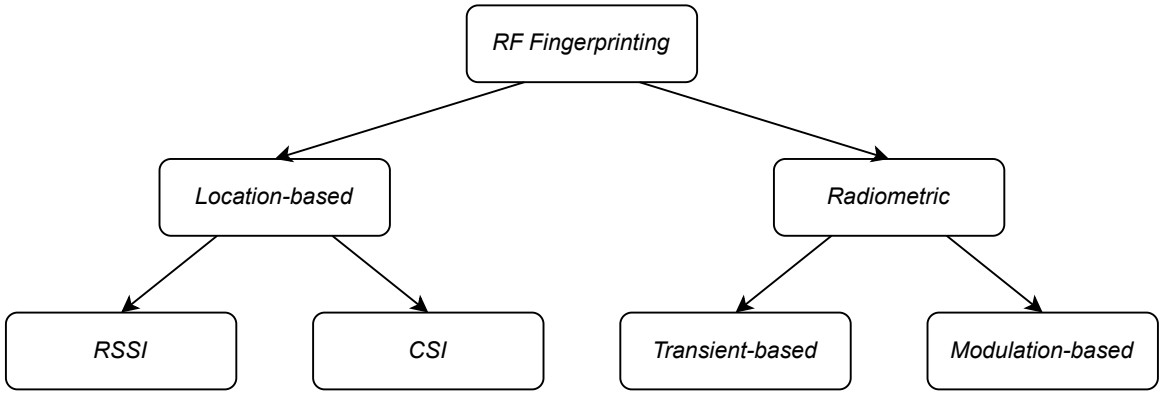


FIGURE 1.4 – Taxonomie de l'authentification par empreinte radio

1.2.1.1 RSS

Le RSS correspond à la puissance d'un signal reçu et il serait le descripteur le plus utilisé dans les descripteurs dépendant du canal selon Xu et al. [23]. Il est notamment influencé par la puissance d'émission de l'émetteur ainsi que par l'atténuation du canal. De plus, dépendants de leur position, deux émetteurs distants peuvent avoir des RSS très différents. Dans [21], Xie et al. indiquent que le RSS fait partie des informations statistiques sur le canal, au même titre que certains descripteurs basés sur la densité spectrale de puissance (DSP) dans le cas des canaux invariants dans le temps. Ainsi, ce descripteur est considéré comme simple à extraire à partir du signal ainsi que plus robuste, mais moins sécurisé, car basé sur des informations génériques (ou «*coarse-grained*») à propos du canal.

1.2.1.2 CSI

Le CSI correspond aux propriétés du canal de propagation entre l'émetteur et le récepteur et fait partie des informations instantanées du canal [21].Il est généralement décomposé en deux notions : la réponse impulsionnelle du canal et la réponse fréquentielle du canal. La réponse impulsionnelle du canal, aussi appelée *CIR : Channel Impulsive Response*, correspond au comportement temporel du canal, alors que la réponse fréquentielle du canal, aussi appelée *CFR : Channel Frequency Response*, correspond au comportement fréquentiel du canal. Dans [21], Xie et al. indiquent que la sécurité de ces approches est meilleure que celles basées sur le RSS, car le CSI correspond à une information plus spécifique (ou «*fine-grained*») à propos du canal. Cependant, les approches basées sur le CSI sont plus complexes à extraire et moins robustes. En effet, le CSI peut varier fortement

en fonction du mouvement, par exemple lorsque l'émetteur se déplace d'une fraction de longueur d'onde de la fréquence porteuse [23].

1.2.2 Radiometric

Le sous-domaine dit *radiometric* correspond aux descripteurs dépendant des imperfections des composants de l'émetteur et qui sont indépendants du canal. En effet, les variabilités lors du processus de fabrication génèrent des imperfections au niveau des composants de l'émetteur. Comme indiqué dans [21], ces imperfections sont considérées comme uniques même pour des émetteurs d'un même modèle. Dans la figure 1.5, les différentes parties d'un signal radiofréquence sont illustrées : le régime transitoire et le régime permanent. Ainsi, il est commun de décomposer ce sous-domaine en deux branches en fonction des descripteurs considérés [44] : *transient-based* et *modulation-based*. Les approches utilisant des descripteurs *transient-based* exploitent le régime transitoire du signal reçu, alors que les approches utilisant des descripteurs *modulation-based*, aussi appelées *steady-state* dans la littérature [14], exploitent quant à elles le régime permanent du signal reçu. Cependant, il est important de noter qu'il existe des approches de RFF exploitant les deux régimes à la fois [45], [46].

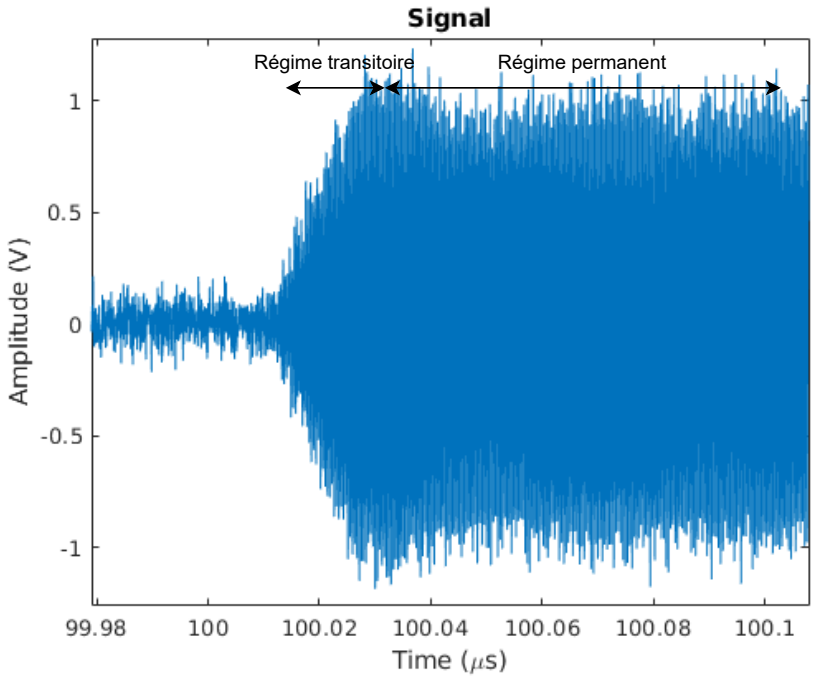


FIGURE 1.5 – Les différentes parties d'un signal RF

1.2.2.1 *Transient-based*

Comme nous venons de l'évoquer, ces approches sont basées sur des descripteurs extraits à partir du régime transitoire, aussi appelé *transient-state* en anglais. Il correspond au régime d'évolution du signal, lorsque celui-ci n'a pas encore atteint son régime permanent. Cependant, il est aussi possible d'exploiter le régime transitoire présent à la fin du signal [46].

Une des étapes primordiales de ces approches est l'estimation précise de la localisation du transitoire. En effet, il est important de bien déterminer la partie du signal correspondant au régime transitoire. Ainsi, dans [47], Mohamed et al. présentent différentes méthodes et comparent leurs performances pour l'extraction du régime transitoire à partir de signaux Wi-Fi. Une fois que la partie correspondante au régime transitoire est récupérée à partir du signal reçu, il est nécessaire d'extraire des descripteurs comme la moyenne ou l'écart-type de la trajectoire de l'énergie du signal [48].

Un des désavantages majeurs des approches basées sur le régime transitoire est qu'elles nécessitent un équipement coûteux pour enregistrer le signal [44]. Par exemple, dans [34], Ozturk et al. utilisent un oscilloscope qui échantillonne le signal à une fréquence de 20 Giga-échantillons par seconde. En effet, le régime transitoire étant extrêmement bref, de l'ordre de quelques microsecondes [1], [48], il est nécessaire d'utiliser une fréquence d'échantillonnage importante pour extraire un maximum d'informations.

1.2.2.2 *Modulation-based*

À l'inverse des approches basées sur le régime transitoire, ces approches s'appuient sur le régime permanent, appelées *steady-state* en anglais. Elles extraient généralement les descripteurs à partir de la représentation en bande de base du signal, nécessitant ainsi un matériel beaucoup moins coûteux, car dépendant de fréquences d'échantillonnage plus faibles. Cependant, tout comme pour les communications sans fil, il est parfois nécessaire de réaliser un ensemble de pré-traitements du signal reçu (synchronisation, ...) avant de réaliser l'extraction des descripteurs [44].

De plus, il est possible de différencier les approches *modulation-based* en plusieurs sous-catégories en nous inspirant partiellement de la taxonomie présentée dans [28] : *feature-based*, *model-based* et *feature-learning*.

Feature-based Cette sous-catégorie consiste à extraire des descripteurs définis par des experts du domaine pour réaliser l'authentification. L'une des approches les plus citées

a été proposée par Brik et al. dans [1]. En effet, dans cet article, les auteurs utilisent le décalage fréquentiel, le décalage en phase et en quadrature ou *In-Phase and Quadrature* (I/Q), la corrélation du préambule, l'erreur en amplitude, ainsi que l'erreur en phase comme descripteurs pour leur système RFF. De plus, de nombreux articles de la littérature exploitent ce type de descripteurs similaires comme dans [13], [41].

Model-based Cette sous-catégorie consiste à utiliser des modèles d'imperfections, c'est-à-dire des modèles de défauts des composants pour réaliser l'authentification. De nombreux modèles d'imperfections ont été proposés pour modéliser les défauts du mélangeur I/Q [16], de l'amplificateur de puissance ou *Power Amplifier* (PA) [49], des Convertisseurs Numérique-Analogiques (CNAs) [50] ou encore de l'Oscilatteur Local (OL) [51].

Feature-learning Cette sous-catégorie est à l'opposé des deux dernières catégories que nous venons de présenter. En effet, au lieu d'utiliser des descripteurs ou des modèles d'imperfections définis par des experts du domaine, ces approches apprennent directement les descripteurs à partir des signaux RF. Elles reposent notamment sur de l'apprentissage de représentation [52] et contrairement aux approches précédentes, elles ne reposent pas (ou peu) sur des connaissances des signaux analysés [28]. Plus particulièrement, la majorité de ces approches reposent sur des modèles d'apprentissage profond [28], [53], [54].

Dans ce manuscrit, les contributions majeures sont présentées au chapitre 3, il s'agit de méthodes *modulation-based* de type *feature learning*. Cependant, la section 2.2 décrit une méthode se basant sur les paramètres du canal de propagation et qui pourrait être utilisée comme une méthode *location-based* de RFF de type CSI. De plus, dans la section 2.1, nous présentons une méthode *model-based* de RFF se basant sur les défauts des CNAs, du mélangeur I/Q, du PA ainsi que de l'OL.

1.3 Modélisation des imperfections pour le RFF

Dans cette sous-section, nous allons présenter l'état de l'art à propos de la modélisation des imperfections radiofréquence (RF). Cependant, ce domaine étant vaste, nous nous limiterons à la modélisation de ces imperfections pour le RFF. Le lecteur intéressé pourra se référer à des ouvrages comme [55]-[57], des thèses [58] ou des articles comme [59], [60] pour approfondir le domaine de la modélisation des imperfections RFF.

Il est possible d'aborder la modélisation des imperfections pour le RFF de différentes façons. Dans cette sous-section, nous en présenterons trois : la présentation des sources d'imperfections d'une chaîne de transmission, la simulation et les approches *model-based*.

Tout d'abord, certains articles de la littérature du RFF présentent les différentes sources d'imperfections d'une chaîne de transmission à l'aide de modélisations. Cela permet aux auteurs d'expliquer aux lecteurs les différentes imperfections responsables de l'empreinte radio. Dans [61], Wang et al. présentent des modèles d'imperfections pour de nombreux composants d'un émetteur radio (CNAs, mélangeur I/Q, ...) mais aussi pour ceux d'un récepteur radio. À l'inverse, dans [62], Zhang et al. se focalisent sur la modélisation du mélangeur I/Q, de l'OL et du PA. Enfin, d'autres auteurs ont proposé des modélisations génériques d'une chaîne de transmission lorsqu'ils présentent le contexte de leur article [13], [28], [63].

De plus, les modélisations des imperfections ont été utilisées pour évaluer les performances de certains algorithmes de RFF. Dans [16], Zhuo et al. ont simulé les imperfections d'un modulateur I/Q (déséquilibre I/Q) pour évaluer les performances de leur approche *model-based*. Dans [39], Sun et al. ont utilisé un modèle de Hammerstein pour modéliser conjointement l'effet du PA et du canal de propagation pour une communication GNSS. Enfin, dans [35], Karunaratne et al. ont utilisé un modèle de Volterra, assimilable à un modèle *odd-order*, pour simuler les non-linéarités du PA. Ces travaux ont l'avantage d'avoir une bonne reproductibilité, car les valeurs des défauts sont généralement disponibles dans les articles. De plus, certains auteurs de la littérature ont aussi étudié en simulation l'impact des différents défauts sur les performances de classification [62], [64].

Enfin, la modélisation des imperfections en RFF est utile dans certaines approches, notamment pour les approches *model-based* qui se servent de ces modèles pour réaliser l'authentification. Ainsi, dans [49] et [50], les auteurs modélisent l'amplificateur de puissance comme une série de Volterra [56], [59]. De plus, dans [50], Polak et al. modélisent les non-linéarités intégrales des CNAs à l'aide d'un pont brownien. Dans un troisième article [51], les mêmes auteurs proposent une méthode de RFF basée sur une modélisation des imperfections de l'oscillateur à l'aide de sa fonction d'autocorrélation. Enfin, dans [16], Zhuo et al. modélisent le phénomène de déséquilibre I/Q du mélangeur I/Q.

Bien que les chapitres 3 et 4 traitent de nos contributions pour le RFF dans un contexte IoT, le chapitre 2 présente nos apports à la modélisation et à l'estimation des imperfections dans une chaîne de transmission. En effet, la sous-section 2.1.2 présente un modèle d'imperfections qui est utilisé à la fois pour la simulation d'imperfections dans la section

3.3, mais qui sert aussi à une méthode *model-based* de RFF décrite dans la sous-section 2.1.4. De manière générale, le chapitre 2 contient plusieurs modèles d'imperfections, que ce soit pour modéliser les défauts de l'émetteur, du canal de propagation ou du récepteur.

1.4 Authentification par empreinte radio pour l'IoT

Dans cette sous-section, nous allons présenter l'utilisation du RFF pour l'IoT. Cependant, la conception de méthodes RFF pour l'IoT a fait l'objet de nombreuses publications, notamment pour le protocole Wi-Fi [1], [28], [45], [65], Zigbee [10], [13], [44], [65], LoRa [33], [66] et Bluetooth [67], [68] entre autres. Dans cette partie, nous ne traiterons pas de manière exhaustive de l'utilisation du RFF pour l'IoT, mais nous nous focaliserons sur les propriétés présentes dans la littérature. De plus, nous traiterons du *RF Fingerprint Enhancement* qui consiste pour un émetteur à modifier la forme d'onde transmise pour accroître l'empreinte radio du signal transmis.

1.4.1 Les propriétés pour l'IoT

Nous nous focaliserons sur les propriétés suivantes : l'adaptabilité, la scalabilité et la complexité. D'ailleurs, nous réutiliserons ces trois propriétés dans le chapitre 3 de ce manuscrit.

1.4.1.1 Adaptabilité

Comme nous l'avons évoqué lors de la présentation du contexte de ce manuscrit, les protocoles IoT et l'IoT, en général, présentent une grande hétérogénéité. Ainsi, le concept d'adaptabilité, présenté dans [69], consiste à développer des méthodes permettant de s'adapter aux différents réseaux IoT. Plus particulièrement, certains auteurs de la littérature mentionnent que l'utilisation de l'apprentissage profond pourrait permettre de résoudre ce problème sans dépendre de descripteurs dépendant d'un protocole [30], [54], [65].

1.4.1.2 Scalabilité

Dans ce paragraphe, nous allons évoquer l'un des défis de l'IoT [70] : la scalabilité puisque, comme nous l'avons mentionné dans le contexte, deux des caractéristiques de l'IoT sont sa dynamicité et son échelle. En effet, ce qui nous intéresse ici, est la gestion d'un

grand nombre d'appareils en constante évolution. De plus, nous découperons cette notion de scalabilité en deux parties : l'ajout/retrait d'émetteurs et la capacité. Cependant, il est important de noter que les deux notions ne sont pas antagonistes mais complémentaires.

Ajout/retrait d'émetteurs Comme nous l'avons présenté dans le contexte de ce manuscrit, l'une des caractéristiques de l'IoT est sa dynamicité. Ainsi, il est nécessaire de concevoir des méthodes de RFF permettant d'ajouter ou de supprimer des émetteurs de la liste des émetteurs légitimes pour gérer cette contrainte. Dans [66], Shen et al. évoquent cette propriété en introduisant notamment une phase d'enrôlement grâce à un réseau siamois. De plus, Arroyo et al. mentionnent aussi que cette propriété serait utile pour le déploiement du RFF dans un contexte opérationnel [71]. Plus particulièrement, dans ce même article, les auteurs indiquent que les réseaux siamois pourraient être une solution intéressante pour répondre à cette problématique.

Capacité Comme nous l'avons présenté dans le contexte de ce manuscrit, l'une des caractéristiques de l'IoT est son échelle. Ainsi, il est nécessaire de concevoir des méthodes de RFF permettant d'identifier un nombre important d'émetteurs. Dans [40], Jian et al. mentionnent cette notion de scalabilité et présentent des architectures de réseaux de neurones profonds capables de prendre en compte un nombre important d'émetteurs, compris entre 50 et 5 000. D'autres auteurs de la littérature se sont intéressés à cette problématique, comme Mattei et al. dans [65], qui exposent une méthode de RFF qui présente de bonnes performances de classification avec 500 émetteurs Wi-Fi.

1.4.1.3 Complexité

La dernière notion que nous évoquerons dans cette partie est la complexité. Dans [13], [54], les auteurs mentionnent qu'il est nécessaire de concevoir des méthodes d'authentification économes en énergie dans le cadre de l'IoT. En effet, comme indiqué dans [5], l'un des défis de l'IoT concerne la prise en compte des contraintes énergétiques des appareils. De plus, dans [71], Arroyo et al. présentent cette notion sous le nom de complexité. Ainsi, ils présentent les performances de plusieurs méthodes de RFF ayant une complexité croissante et allant d'une méthode basée sur des descripteurs à un réseau de neurones convolutif profond. D'autres auteurs de la littérature présentent aussi des architectures de réseaux de neurones profonds optimisés vis-à-vis de cette notion [30], [72], [73].

1.4.2 *RF Fingerprint Enhancement*

Le terme de *RF fingerprint enhancement* a été proposé dans le programme *Radio Frequency Machine Learning Systems* (RFMLS) de l'agence américaine DARPA [74]. Ce programme se focalise sur l'utilisation de l'apprentissage automatique pour les signaux radio, notamment pour l'authentification par empreinte radio. Plus particulièrement, la tâche 1A, appelée *Feature Learning* est assez similaire au concept d'adaptabilité que nous venons de présenter. De plus, la tâche 1B, appelée *RF Waveform Synthesis*, consiste à apprendre une forme d'onde permettant de renforcer l'empreinte radio d'un émetteur. Par exemple, dans [75], les auteurs présentent une méthode basée sur un filtre conçu pour améliorer les performances de décision d'un système RFF.

1.5 Implémentations logicielles et matérielles en RFF

À notre connaissance, il existe peu d'articles de la littérature qui traitent de l'implémentation de méthodes de RFF sur des plateformes logicielles ou matérielles, que ce soit sur processeur ou *Central Processing Unit* (CPU), processeur graphique ou *Graphical Processing* (GPU) ou *Field Programmable Gate Array* (FPGA). Dans [30], Jian et al. proposent des implémentations CPU, GPU et FPGA de plusieurs réseaux de neurones de RFF visant plusieurs protocoles (Wi-Fi et ADS-B). Plus particulièrement, les auteurs utilisent des techniques de *pruning*, consistant à supprimer les poids/coefficients d'un réseau de neurones avec une amplitude trop faible pour obtenir des réseaux de neurones plus facilement implémentable. Dans [10], Lowder propose une méthode de RFF conçue pour le protocole Zigbee et implémentée sur la partie FPGA d'une plateforme radio logicielle ou *Software-Defined Radio* (SDR) USRP X310. Enfin, dans [73], Peng et al. présentent l'implémentation de deux réseaux de neurones pour le RFF d'émetteurs Zigbee sur FPGA. Selon les auteurs, leur architecture, nommée *Photonics-Inspired Recurrent Neural Network*, contient beaucoup moins de paramètres que les architectures de la littérature.

Dans le chapitre 4, nous présentons les implémentations logicielles et matérielles des méthodes que nous avons proposées dans le chapitre 3. Ainsi, nous comparerons nos implémentations à celles que nous venons de présenter.

Chapitre 2

Modélisation et estimation des imperfections dans une chaîne de transmission radio

Ce chapitre traite de différentes imperfections présentes dans une chaîne de transmission radio. Ainsi, la chaîne de transmission radio sera décomposée en trois parties, correspondant aux différentes sections :

1. Émetteur radio
2. Canal de propagation
3. Récepteur radio

Trois modèles d'imperfections seront introduits dans ce chapitre, un pour chaque partie de la chaîne de transmission. De plus, l'estimation de ces différentes imperfections sera aussi présentée. Enfin, plusieurs cas d'utilisation des modèles d'imperfections et de leurs estimations seront proposés parmi :

— La simulation des imperfections.
— La correction des imperfections.
— L'authentification par empreinte radio.

Dans la suite de ce manuscrit, nous nous focaliserons sur les signaux de communication à bande-étroite et nous utiliserons la notation suivante pour les représenter [56], [57] :

$$x(t) = a(t)cos(2\pi f_0 t + \varphi(t)) \tag{2.1}$$
$$= x_I(t)cos(2\pi f_0 t) - x_Q(t)sin(2\pi f_0 t) \tag{2.2}$$

avec :

— $a(t)$: la modulation d'amplitude.
— $\varphi(t)$: la modulation de phase et/ou de fréquence.

— $x_I(t)$: la composante du signal en phase.
— $x_Q(t)$: la composante du signal en quadrature.
Ainsi que sa représentation en bande de base (ou signal I/Q) :

$$\tilde{x}(t) = a(t)e^{j\varphi(t)} \tag{2.3}$$
$$= x_I(t) + jx_Q(t) \tag{2.4}$$

Dans ce manuscrit, la représentation en bande de base du signal sera centrale, que ce soit pour la modélisation des imperfections de la chaîne de transmission ou bien pour les méthodes de RFF proposées. En effet, comme indiqué par Tse et al. dans [76], la représentation en bande de base du signal est d'une grande utilité pour la conception et la modélisation des systèmes de communications.

Pour faciliter les expérimentations de ce chapitre, nous considérerons le cas de modulations *M-ary Phase Shift Keying* (M-PSK). Cependant, les modélisations ainsi que la plupart des méthodes d'estimations présentées sont utilisables avec d'autres modulations.

2.1 L'émetteur radio : le cas de la modulation en quadrature

L'émetteur a pour rôle de transposer fréquentiellement le signal modulant, puis de l'amplifier avant de le transmettre à l'antenne [57]. Nous nous focaliserons plus particulièrement sur l'étude des architectures d'émetteurs basées sur la représentation en bande de base du signal.

2.1.1 Hypothèses pour la modélisation des imperfections d'un émetteur

Dans le cas d'un émetteur, nous cherchons à obtenir le signal $x(t)$ à partir de sa représentation en bande de base $\tilde{x}(t)$ [57] :

$$x(t) = \Re(\tilde{x}(t)e^{j2\pi f_0 t}) \tag{2.5}$$

Cette équation nous permet d'obtenir la structure du mélangeur I/Q (en émission) ou modulateur en quadrature, présentée en figure 2.1, utilisée dans de nombreuses archi-

tectures d'émetteurs RF et permettant de combiner la voie en phase $\tilde{x}_I(t)$ et la voie en quadrature $\tilde{x}_Q(t)$.

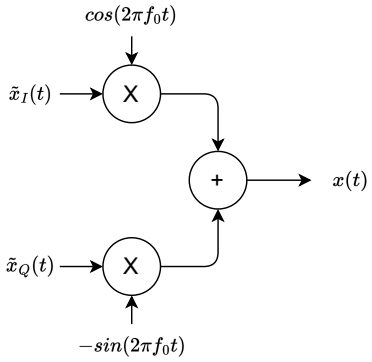


FIGURE 2.1 – Structure du mélangeur I/Q (en émission)

Dans [57], Baudoin et al. présentent plusieurs architectures d'émetteurs radio. D'après eux, les architectures les plus généralistes sont basées sur des modulateurs en quadrature exploitant la décomposition en bande de base du signal de communication. Il existe de nombreuses architectures d'émetteurs, notamment basées sur l'utilisation d'une boucle de synthèse de fréquence modulée. Cependant, ces dernières sont réservées à des modulations plus spécifiques comme les modulations de fréquence ou de phase.

Dans cette section, nous allons considérer les deux architectures d'émetteurs considérées comme les plus classiques selon [57] :

— L'architecture d'émetteurs sans fréquence intermédiaire (FI0) (figure 2.2) : consiste à transposer un signal I/Q à la fréquence f_0.

— L'architecture d'émetteurs avec fréquence intermédiaire (FI) (figure 2.3) : consiste d'abord à transposer un signal I/Q à la fréquence intermédiaire f_I, puis à le transposer une seconde fois à la fréquence f_0 à l'aide d'un second mélangeur.

La différence entre les deux architectures repose sur l'utilisation d'un second mélangeur dans le cas de l'architecture d'émetteurs avec fréquence intermédiaire. D'après [57], la première architecture est considérée comme «plus grand public», car nécessitant moins de composants et ayant donc une consommation moindre. Cependant, la seconde architecture est considérée de meilleure qualité, car limitant les problématiques de couplage entre le PA et l'oscillateur, ainsi que des problèmes de bruits parasites émis.

Ces deux architectures sont composées de plusieurs composants :

— Les convertisseurs numériques-analogique (CNA) : ces composants permettent de convertir un signal numérique en un signal analogique.

— Mélangeur I/Q : ce composant permet de combiner les voies I (en phase) et Q (en quadrature) pour obtenir le signal de communication à la fréquence f_0 (ou f_I).
— Oscillateur local (OL) : ce composant génère le signal de porteuse $cos(2\pi f_0 t)$ ($cos(2\pi f_I t)$ et $cos(2\pi(f_0 - f_I)t)$) utilisé pour la transposition en fréquence du signal de communication.
— Synthétiseur de fréquence : ce composant est utilisé pour faire varier la fréquence de l'OL.
— Amplificateur de puissance (PA) : ce composant est utilisé pour amplifier le signal à transmettre.
— Filtres : ces composants sont utilisés pour filtrer les fréquences non souhaitées. Il peut s'agir de filtres passe-bas ou passe-bande.
— Antenne : ce composant est utilisé pour émettre le signal de communication dans l'environnement de propagation, sous la forme d'une onde électromagnétique.

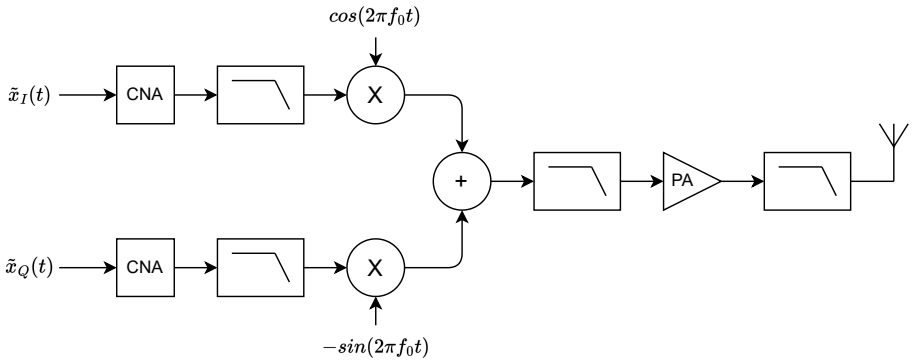


FIGURE 2.2 – Architecture d'émetteurs sans fréquence intermédiaire (FI0)

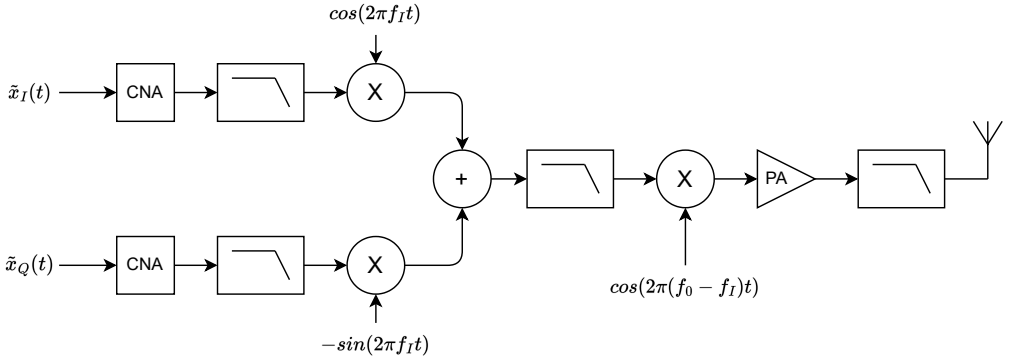


FIGURE 2.3 – Architecture d'émetteurs avec fréquence intermédiaire (FI)

2.1.2 Modélisation des imperfections d'un émetteur radio

Dans cette sous-section, nous allons introduire un modèle d'imperfections en bande de base, présenté en figure 2.11, permettant de prendre en compte les imperfections suivantes :

- Décalage I/Q : créant un décalage de la constellation par rapport à l'origine due aux CNA et au mélangeur I/Q [1], [77], [78].
- Déséquilibre en gain : créant une distorsion de la constellation due à un déséquilibre en gain entre la voie en phase et en quadrature des CNA et du mélangeur I/Q [16], [77], [78].
- Déséquilibre en phase : créant une distorsion de la constellation due à un déséquilibre en phase entre la voie en phase et en quadrature due au mélangeur I/Q [16], [61], [77].
- Non-linéarités : créant une distorsion non-linéaire de la constellation due aux composants non-linéaires de l'émetteur (PA, mélangeur(s)) [56], [57], [59].
- Décalage fréquentiel : créant une rotation dynamique de la constellation due à l'OL [1], [51].
- Bruit de phase : correspondant à une instabilité en phase de la constellation due à l'OL, ainsi qu'à divers couplages [51], [58], [79].

La figure 2.4, inspirée de celle proposée dans [28], montre l'impact des défauts considérés dans cette section pour une architecture FI0. Les différents défauts évoqués sont illustrés en montrant leurs effets respectifs sur la constellation d'une modulation d'amplitude en quadrature à 16 états (16-QAM). Plus particulièrement, les différents traits en pointillés reliant le ou les composants à l'origine d'un défaut à la figure illustrant son effet.

Nous avons choisi de décomposer le modèle proposé en trois parties :

- Modélisation conjointe des imperfections des CNA et du mélangeur I/Q .
- Modélisation des imperfections de l'oscillateur local.
- Modélisation des non-linéarités du PA.

Il serait possible de prendre en compte les problèmes de désadaptation d'impédance ou d'autres imperfections comme la distorsion harmonique des CNA [28] ou les distorsions linéaires des filtres et de l'antenne [61], mais le modèle proposé offre un compromis intéressant entre la simplicité et la fidélité de la modélisation. De plus, dans [62], Zhang et al. précisent qu'il s'agit des défauts d'un émetteur majoritairement pris en compte dans la littérature du RFF.

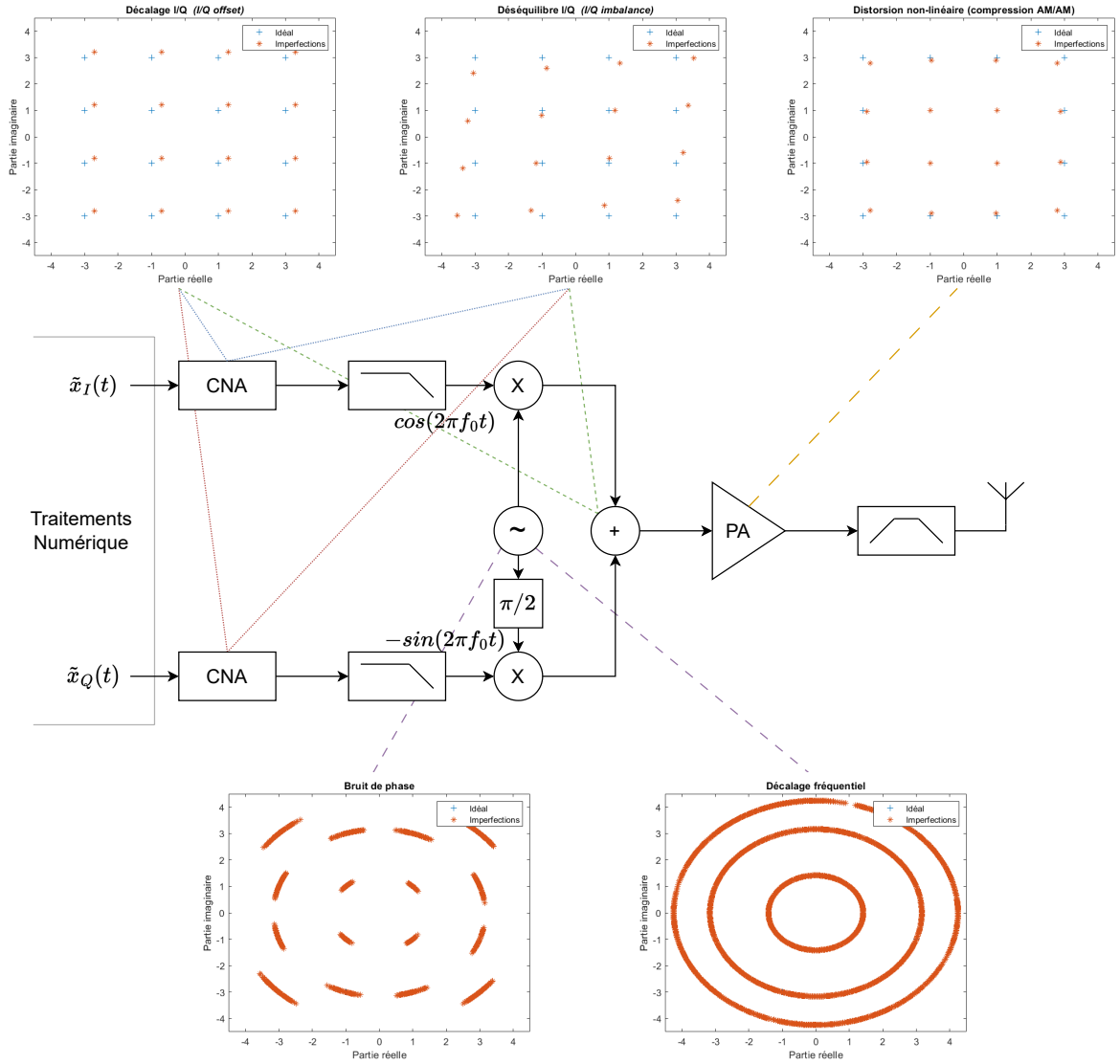


FIGURE 2.4 – Impact des défauts considérés sur une constellation 16-QAM

2.1.2.1 La modélisation conjointe des défauts des CNA et du mélangeur I/Q

Le mélangeur I/Q présente généralement deux défauts majeurs selon la littérature [16], [62], [77] : le déséquilibre en gain et le déséquilibre en phase. D'une part, le déséquilibre en gain ou *gain imbalance*, modélisé par un paramètre ϵ, correspond à une différence d'amplitude entre la porteuse en phase $cos(2\pi f_0 t)$ et la porteuse en quadrature $sin(2\pi f_0 t + \pi/2)$. D'autre part, le déséquilibre en phase ou *phase skew*, modélisé par un paramètre θ, correspond à un déphasage différent de $\pi/2$ entre la porteuse en phase et celle en quadrature. Il existe également des problèmes d'isolation, appelés décalage I/Q ou *I/Q*

offset et modélisés par les paramètres A_I (partie réelle) et A_Q (partie imaginaire). La figure 2.5 présente les défauts d'un mélangeur I/Q, ainsi qu'un modèle de ses imperfections en bande de base, inspiré des modèles proposés dans [16], [77].

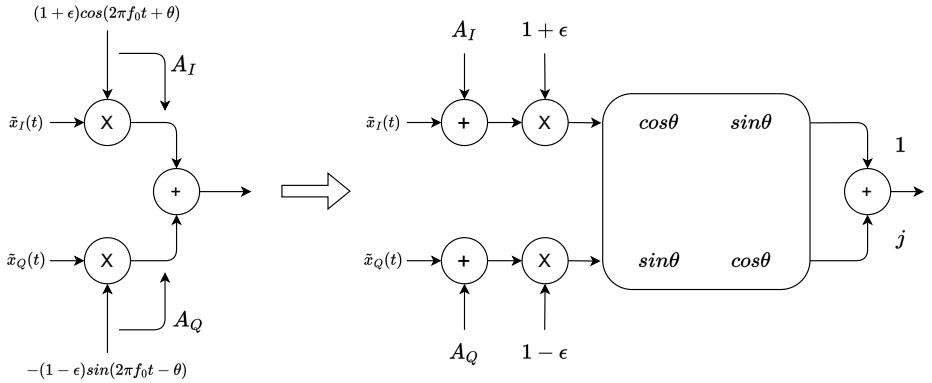


FIGURE 2.5 – Défauts du mélangeur I/Q (en émission) et modèle bande de base associé

Cette modélisation permet aussi de prendre en compte les erreurs de décalage et de gain présentes dans les CNA [78]. D'une part, un CNA peut présenter une erreur de décalage, consistant en une tension non nulle en sortie se superposant au signal généré par le CNA. D'autre part, un CNA peut présenter une erreur de gain, consistant en une pente de la courbe de transfert différente de celle de la courbe idéale. Ainsi, les erreurs de décalage seront modélisées conjointement aux problèmes d'isolation à l'aide des paramètres A_I et A_Q. Quant aux erreurs de gain, elles seront modélisées conjointement au déséquilibre en gain du mélangeur I/Q à l'aide du paramètre ϵ. La figure 2.6, présente le modèle d'imperfections conjoint des défauts des CNA et d'un mélangeur I/Q.

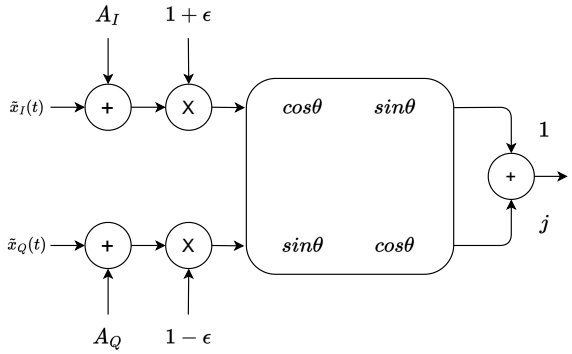


FIGURE 2.6 – Modélisation conjointe des défauts des CNA et d'un mélangeur I/Q

2.1.2.2 Modélisation des défauts de l'oscillateur local

L'OL est un composant servant à générer le signal de porteuse $cos(2\pi f_0 t)$, et si nécessaire les fréquences porteuses intermédiaires pour les architectures FI. Généralement, les défauts de ce composant sont modélisés par un terme $\phi_{OL}(t)$ de la manière suivante [62], [80] :

$$cos(2\pi f_0 t + \phi_{OL}(t)) \tag{2.6}$$

Les défauts majoritairement considérés dans la littérature sont le décalage fréquentiel Δf, ainsi que le bruit de phase $\phi(t)$ [51], [62], [80]. D'une part, le décalage fréquentiel Δf mesure le décalage de la porteuse générée par rapport à la fréquence porteuse idéale f_0. D'autre part, le bruit de phase $\phi(t)$ est relié à la stabilité fréquentielle à court terme de l'oscillateur [57]. La modélisation de ces deux défauts est décrite par l'équation ci-dessous :

$$\phi_{OL}(t) = 2\pi\Delta f t + \phi(t) \tag{2.7}$$

En ce qui concerne le décalage fréquentiel, nous le considérons comme constant [1], [13], [54] et sa valeur caractérisée par un paramètre δf, exprimée en partie par millions (ppm) de la manière suivante [62] :

$$-\frac{\delta f}{10^6} f_0 \leq \Delta f \leq \frac{\delta f}{10^6} f_0 \tag{2.8}$$

Pour ce qui est du bruit de phase, il existe plusieurs modèles dans la littérature [58], [60], [79]. Nous avons choisi de le modéliser comme un bruit gaussien filtré [58], car il s'agit d'un modèle déjà utilisé dans la littérature du RFF [62] et implémenté dans la fonction **comm.PhaseNoise** du logiciel Matlab.

La modélisation des défauts de l'OL en bande de base est décrite à la figure 2.7 et elle est généralement réalisée à l'aide d'une exponentielle complexe $e^{j\phi_{OL}(t)}$ ainsi :

$$\tilde{y}[t] = \tilde{x}(t)e^{j\phi_{OL}(t)} \tag{2.9}$$

La figure 2.8 présente la modélisation conjointe des défauts d'un mélangeur I/Q ainsi que de l'OL. Il est également possible d'utiliser le même raisonnement que précédemment, ce qui nous permet de modéliser les erreurs de décalage et de gain des CNA à l'aide des paramètres A_I, A_Q et ϵ.

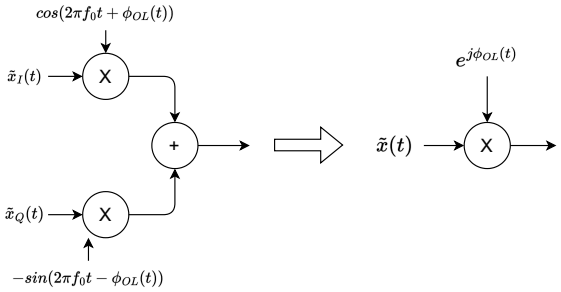


FIGURE 2.7 – Modélisation des défauts de l'oscillateur (en émission)

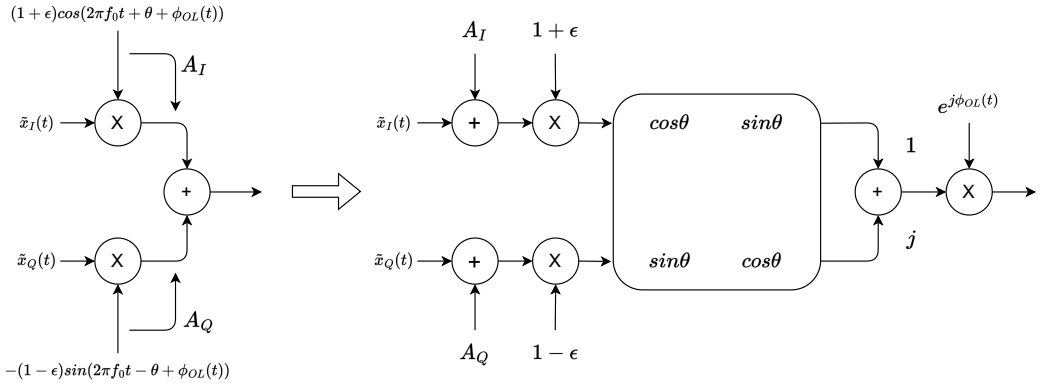


FIGURE 2.8 – Modélisation conjointe des défauts des CNA, du mélangeur I/Q et de l'oscillateur

2.1.2.3 Modélisation des non-linéarités du PA

Les sources de distorsions non-linéaires sont multiples dans une chaîne d'émission, que ce soit les CNA, les différents mélangeurs ou le PA. Cependant, l'élément principal engendrant ces distorsions est l'amplificateur de puissance (PA) [57].

Ces non-linéarités sont généralement caractérisées par deux grandeurs : la compression AM/AM et la conversion AM/PM [59]. D'une part, la compression AM/AM engendre une distorsion de l'amplitude du signal de sortie en fonction de l'amplitude du signal d'entrée $a(t)$, modélisée par un terme $f(a(t))$. D'autre part, la conversion AM/PM engendre une distorsion de la phase de sortie en fonction de l'amplitude du signal d'entrée $a(t)$, modélisée par un terme $g(a(t))$. Une modélisation en bande de base du signal de sortie $\tilde{y}(t)$ en

fonction du signal d'entrée $\tilde{x}(t)$ peut être décrite par l'équation ci-dessous [57], [62] :

$$\tilde{y}(t) = f(a(t))e^{j(\phi(t)+g(a(t)))} \tag{2.10}$$

$$= h(a(t))\tilde{x}(t) \tag{2.11}$$

avec $h(a(t)) = \frac{f(a(t))}{a(t)}e^{jg(a(t))}$: le gain complexe du PA en fonction de l'amplitude $a(t)$ du signal d'entrée.

Il existe un grand nombre de modèles dans la littérature permettant de modéliser des non-linéarités en bande de base parmi [56], [59] : *odd-order*, Wiener, Hammerstein, Wiener-Hammerstein, Volterra et Saleh. Nous avons choisi de modéliser les non-linéarités en utilisant le modèle *odd-order*, car il s'agit d'un modèle populaire [81] qui a été déjà utilisé en simulation dans la littérature du RFF [35], [39]. Ce modèle de non-linéarité en bande base, décrit en figure 2.9, est basé sur une décomposition en série de Taylor à l'ordre N de la non-linéarité et largement utilisée pour caractériser les non-linéarités des circuits RF [57]. Bien que ce modèle permette principalement de modéliser le phénomène de compression AM/AM, il est aussi possible de modéliser le phénomène de conversion AM/PM en considérant des coefficients complexes b_i [81]. Il est également utilisé dans des modèles de non-linéarités avec mémoire, qui prennent aussi en compte les distorsions linéaires, comme le modèle de Wiener, Hammerstein ou *parallel-Hammerstein* [39], [59].

Le modèle *odd-order* est décrit par l'équation suivante [57], [59], [81] :

$$\tilde{y}(t) = \sum_{k=1}^{M} b_{2k-1}|\tilde{x}(t)|^{2k-2}\tilde{x}(t) \tag{2.12}$$

avec :

- $M = \lfloor (N+1)/2 \rfloor$: l'ordre de la non-linéarité en bande de base. Dans la littérature, les auteurs se limitent à $M = 3$ [35], voire $M = 4$ [39].
- $b_k = \binom{2k-1}{k} a_{2k-1}/2^{2k}$: un des coefficients du modèle *odd-order*.

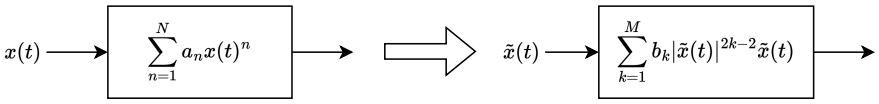


FIGURE 2.9 – Modélisation des non-linéarités du PA

2.1.2.4 Modèles des imperfections en bande de base

La figure 2.10 correspond à un modèle intermédiaire des imperfections en bande de base réunissant les différentes modélisations que nous avons proposées et permettant de prendre en compte des défauts provenant des CNA, du mélangeur I/Q, de l'OL ainsi du PA.

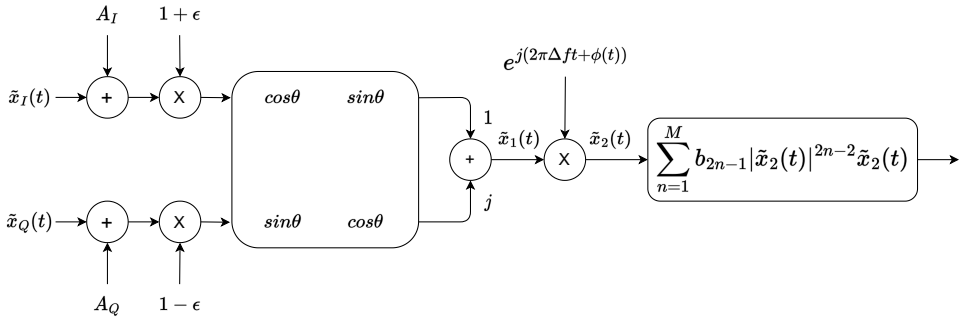


FIGURE 2.10 – Modèle intermédiaire des imperfections de l'émetteur en bande de base

De son côté, la figure 2.11 présente le modèle d'imperfections des défauts d'un émetteur en bande de base. Une des spécificités de ce modèle est que la modélisation des défauts de l'oscillateur local et le modèle *odd-order* sont inversés. La démonstration permettant de justifier cette commutation est la suivante :

$$\sum_{n=1}^{M} b_{2n-1}|\tilde{x}_1(t)e^{j\phi_{OL}(t)}|^{2n-2}\tilde{x}_1(t)e^{j\phi_{OL}(t)} = (\sum_{n=1}^{M} b_{2n-1}|\tilde{x}_1(t)|^{2n-2}\tilde{x}_1(t))e^{j\phi_{OL}(t)} \tag{2.13}$$

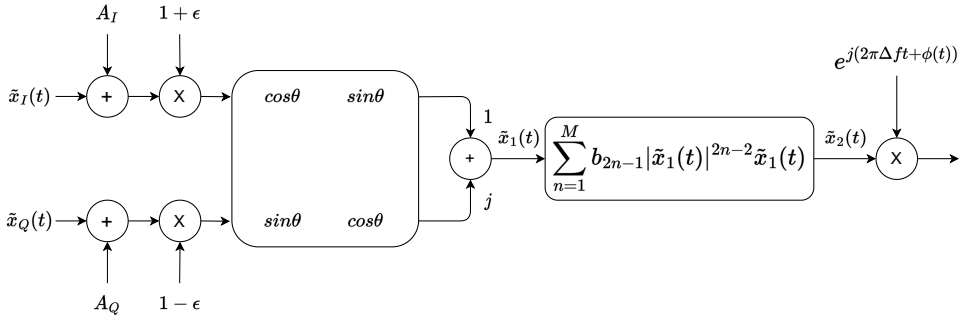


FIGURE 2.11 – Modèle des imperfections de l'émetteur en bande de base

L'intérêt de commuter la modélisation des défauts de l'oscillateur local et des non-linéarités sera notamment utile pour corriger le décalage fréquentiel, comme nous le verrons par la suite.

Plusieurs variations de ce modèle d'imperfections en bande de base seront utilisées en section 3.3 pour simuler les imperfections de plusieurs émetteurs et évaluer les performances et les propriétés de nos méthodes de RFF.

2.1.3 Estimation des imperfections d'un émetteur radio

Dans cette sous-section, nous allons tout d'abord introduire des techniques permettant d'estimer indépendamment les paramètres des différentes parties du modèle d'imperfections en bande de base. Puis, nous discuterons des stratégies pour estimer les paramètres du modèle de manière conjointe.

Dans cette section, nous considérerons des signaux numériques. Ainsi, $\tilde{x}[n]$, respectivement $\tilde{y}[n]$, correspondra au n-ème échantillons du signal d'entrée $\tilde{x}(t)$, respectivement du signal de sortie $\tilde{y}(t)$.

2.1.3.1 Estimation des imperfections du modèle conjoint des défauts des CNA et du mélangeur I/Q

L'algorithme que nous proposons pour estimer les paramètres du modèle d'imperfections (A_I, A_Q, ϵ et θ) s'appuie sur l'algorithme de la descente de gradient stochastique [31], où la fonction de coût associée est l'erreur moyenne quadratique :

$$C[n] = \mathbb{E}(|e[n]|^2) \tag{2.14}$$

avec $e[n] = \tilde{y}[n] - \tilde{x}[n]$, $\tilde{x}[n]$ étant l'entrée du modèle et $\tilde{y}[n]$ étant la sortie du modèle.

La descente de gradient stochastique consiste, à chaque étape k, à calculer les gradients sur une seule observation $e[n^{(k)}]$ choisie aléatoirement [31], avec $n^{(k)}$ l'indice de l'observation choisie à l'étape k.

La descente de gradient à l'étape $k+1$ est exprimée de la manière suivante :

$$\hat{A_I}^{(k+1)} = \hat{A_I}^{(k)} - 2\mu\mathbb{E}(\Re(e^*[n^{(k+1)}]\frac{\partial e[n^{(k+1)}]}{\partial \hat{A_I}^{(k)}})) \tag{2.15}$$

$$\hat{A_Q}^{(k+1)} = \hat{A_Q}^{(k)} - 2\mu\mathbb{E}(\Re(e^*[n^{(k+1)}]\frac{\partial e[n^{(k+1)}]}{\partial \hat{A_Q}^{(k)}})) \tag{2.16}$$

$$\hat{\epsilon}^{(k+1)} = \hat{\epsilon}^{(k)} - 2\mu\mathbb{E}(\Re(e^*[n^{(k+1)}]\frac{\partial e[n^{(k+1)}]}{\partial \hat{\epsilon}^{(k)}})) \tag{2.17}$$

$$\hat{\theta}^{(k+1)} = \hat{\theta}^{(k)} - 2\mu\mathbb{E}(\Re(e^*[n^{(k+1)}](j(1+\hat{\epsilon}^{(k)})\frac{\partial e[n^{(k+1)}]}{\partial\hat{\theta}^{(k)}})) \tag{2.18}$$

avec les gradients associés :

$$\frac{\partial e[n^{(k+1)}]}{\partial \hat{A_I}^{(k)}} = (1+\hat{\epsilon}^{(k)})e^{j\hat{\theta}^{(k)}} \tag{2.19}$$

$$\frac{\partial e[n^{(k+1)}]}{\partial \hat{A_Q}^{(k)}} = j(1-\hat{\epsilon}^{(k)})e^{-j\hat{\theta}^{(k)}} \tag{2.20}$$

$$\frac{\partial e[n^{(k+1)}]}{\partial\hat{\epsilon}^{(k)}} = (\tilde{x}_I[n^{(k+1)}] + \hat{A_I}^{(k)})e^{j\hat{\theta}^{(k)}} - j(\tilde{x}_Q[n^{(k+1)}] + \hat{A_Q}^{(k)})e^{-j\hat{\theta}^{(k)}} \tag{2.21}$$

$$\frac{\partial e[n^{(k+1)}]}{\partial\hat{\theta}^{(k)}} = j(1+\hat{\epsilon}^{(k)})(\tilde{x}_I[n^{(k+1)}]+\hat{A_I}^{(k)})e^{j\hat{\theta}^{(k)}}+(1-\hat{\epsilon}^{(k)})(\tilde{x}_Q[n^{(k+1)}]+\hat{A_Q}^{(k)})e^{-j\hat{\theta}^{(k)}} \tag{2.22}$$

Le choix de l'utilisation d'une descente de gradient stochastique est justifié par le fait que cet algorithme est capable d'atteindre un optimum global, car l'aspect stochastique de cette méthode peut permettre d'échapper aux minimums locaux [31]. L'aspect stochastique rend également l'algorithme beaucoup plus rapide puisqu'il n'est pas nécessaire de manipuler l'ensemble des observations à chaque étape [31].

2.1.3.2 Estimation des imperfections du modèle d'imperfections de l'amplificateur de puissance

La technique proposée permet d'estimer les paramètres du modèle *odd-order*, i.e. des différents paramètres b_{2n-1}. Cette technique consiste à réaliser une méthode d'estimation des moindres carrés du système d'équations ci-dessous à l'aide de N couples d'échantillons $(\tilde{x}[n], \tilde{y}[n])$ [49], [50], [59].

$$\underbrace{\begin{bmatrix}\tilde{y}[1]\\ \tilde{y}[2]\\ \vdots\\ \tilde{y}[N]\end{bmatrix}}_{\mathbf{Y}} = \underbrace{\begin{bmatrix}\tilde{x}[1] & |\tilde{x}[1]|^2\tilde{x}[1] & \dots & |\tilde{x}[1]|^{2M-2}\tilde{x}[1]\\ \tilde{x}[2] & |\tilde{x}[2]|^2\tilde{x}[2] & \dots & |\tilde{x}[2]|^{2M-2}\tilde{x}[2]\\ \vdots & & & \\ \tilde{x}[N] & |\tilde{x}[N]|^2\tilde{x}[N] & \dots & |\tilde{x}[N]|^{2M-2}\tilde{x}[N]\end{bmatrix}}_{\mathbf{X}}\underbrace{\begin{bmatrix}b_1\\ b_3\\ \vdots\\ b_{2M-1}\end{bmatrix}}_{\mathbf{h}} \tag{2.23}$$

Nous avons considéré M=3, comme dans [35], ce qui nous permet de prendre en compte les non-linéarités jusqu'à l'ordre 5.

Ce qui permet d'obtenir alors la fonction de coût suivante :

$$J(h) = \frac{1}{N}\sum_{i=1}^{N} |e[i]|^2 \tag{2.24}$$

avec $e[i] = \tilde{y}[i] - \sum_{j=1}^{M} \hat{b}_{2i-1}|\tilde{x}(i)|^{2j-2}\tilde{x}(i)$

$$J(h) = \frac{1}{N}(Y - Xh)^H(Y - Xh) \tag{2.25}$$

$$= \frac{1}{N}(h^H X^H Xh - Y^H Xh - h^H X^H Y + Y^H Y) \tag{2.26}$$

Après étude de cette fonction de coût, il s'avère qu'elle est convexe si la matrice $X^H X$ est semi-définie positive [82], ce qui est le cas puisqu'elle est hermitienne. Le minimum global peut ainsi être trouvé en utilisant le gradient fourni par [83] :

$$\frac{\partial J(h)}{\partial h^H} = \frac{1}{N}(X^H Xh - X^H Y) \tag{2.27}$$

$$\hat{h} = (X^H X)^{-1} X^H Y \tag{2.28}$$

De plus, un terme de régularisation $\sum_i w_i h_i^2$ $(h^H Rh)$ a spécialement été conçu pour prendre en compte des informations *a priori* sur l'amplitude des coefficients de h à l'aide des termes w_i. Ainsi, la nouvelle fonction de coût pour ce problème est la suivante :

$$J(h) = \frac{1}{N}(Y - Xh)^H(Y - Xh) \tag{2.29}$$

$$= \frac{1}{N}(Y - Xh)^H(Y - Xh) + \lambda h^H Rh \tag{2.30}$$

avec $R = \begin{bmatrix} w_1 & 0 & 0 \\ 0 & w_3 & 0 \\ 0 & 0 & w_5 \end{bmatrix}$

Les coefficients w_i permettent de régulariser la solution en prenant en compte les valeurs *a priori* des coefficients b_{2i-1} dépendants des points d'interception [57].

Il est important de préciser que par convention w_1 vaut 1. Ainsi, la nouvelle fonction de coût est toujours convexe. En effet, la matrice $X^H X + \lambda R$ reste hermitienne, car la

matrice λR ne modifie que les termes diagonaux de $X^H X + \lambda R$.

2.1.3.3 Estimation des paramètres des imperfections de l'oscillateur

Dans cette partie sur l'estimation des paramètres des imperfections de l'oscillateur, nous proposerons d'abord une méthode d'estimation du décalage fréquentiel Δf, puis des pistes permettant d'estimer le bruit de phase. Nous nous placerons dans le cas de la modulation M-PSK qui sera largement utilisée dans la suite de ce manuscrit.

Une des méthodes d'estimation du décalage fréquentiel la plus courante est la récupération de la porteuse par élévation à la puissance M [56], [84]. Cette technique de récupération de porteuse aveugle est conçue pour un signal $\tilde{x}(t)$ de type M-PSK. En effet, ce type de signal peut être décrit en bande de base de la manière suivante :

$$\tilde{x}(t) = \sum_{k=-\infty}^{\infty} a_k p(t - kT_s) \tag{2.31}$$

avec :

— $a_k \in \{1, e^{j\frac{2\pi}{M}}, \cdots, e^{j\frac{2\pi(M-1)}{M}}\}$: les symboles M-PSK transmis.

— $p(t) = rect_T(t)$: le filtre de mise en forme rectangulaire de largeur T.

— T_s : le temps symbole.

Ainsi, si le signal M-PSK $\tilde{x}(t)$ subit un décalage fréquentiel Δf, le signal reçu peut être décrit de la manière suivante :

$$\tilde{y}(t) = \tilde{x}(t) e^{j2\pi\Delta f t} \tag{2.32}$$

La première étape de la méthode d'estimation du décalage fréquentiel consiste à traiter le signal reçu en l'élevant à la puissance M :

$$\tilde{y}(t)^M = (\tilde{x}(t) e^{j2\pi\Delta f t})^M \tag{2.33}$$

$$= \tilde{x}(t)^M e^{j2M\pi\Delta f t} \tag{2.34}$$

$$= e^{j2M\pi\Delta f t} \tag{2.35}$$

L'astuce de cette technique d'estimation fréquentielle repose sur le fait qu'élever un signal M-PSK à la puissance M a pour effet d'annuler le signal de communication car $a_k^M = 1$.

Une fois le signal de communication $\tilde{x}(t)$ «annulé», il est possible d'estimer le décalage

fréquence $\hat{\Delta}f = \frac{1}{M}(arg\,max_f TF(\tilde{y}(t)^M))$ avec $TF(.)$ la transformée de Fourier. De plus, cette technique d'estimation est robuste à la présence d'un bruit de phase $e^{j\phi(t)}$, car le bruit de phase n'a qu'un effet d'étalement fréquentiel de la porteuse [55], [57].

Une fois que le décalage fréquentiel $\hat{\Delta}f$ est estimé à l'aide de la méthode de récupération de la puissance par élévation à la puissance M, il suffit de multiplier $\tilde{y}(t)^M$ par le conjugué du décalage fréquentiel modifié $e^{-j2M\pi\hat{\Delta}ft}$ afin d'obtenir le bruit de phase :

$$\tilde{y}(t)^M e^{-j2M\pi\hat{\Delta}ft} = e^{j4\phi(t)} \tag{2.36}$$

Concernant l'estimation des paramètres du bruit de phase, plusieurs méthodes ont été proposées dans la littérature, notamment basées sur la fonction d'autocorrélation ou du spectre [45], [58]. Ainsi, en se basant sur $\tilde{y}(t)^M e^{-j2M\pi\hat{\Delta}ft}$ et sur l'une de ces méthodes, il serait possible d'estimer les différents paramètres du bruit de phase.

2.1.3.4 Stratégie d'estimation globale des paramètres du modèle d'imperfections simplifié

En se basant sur les techniques d'estimation que l'on vient de présenter, nous allons proposer différentes stratégies pour estimer les paramètres du modèle d'imperfection. La première étape consiste à estimer le décalage fréquentiel Δf à partir du signal reçu $\tilde{z}(t) = \tilde{y}(t)e^{j2\pi\Delta ft}$. Puis, pour synchroniser le signal reçu, il suffit de le multiplier par $e^{-j2\pi\hat{\Delta}ft}$ pour obtenir $\tilde{y}(t)$.

Une fois de décalage fréquentiel compensé, une stratégie d'estimation globale permettrait d'estimer les paramètres A_I, A_Q, ϵ, θ, ainsi que les coefficients b_{2n-1} à partir du signal $\tilde{y}(t)$. Des méthodes heuristiques ou des méta-heuristiques pourraient être utilisées comme celles décrites dans [85], par exemple, en utilisant l'optimisation par essaims particulaires. En effet, dans notre cas, il est nécessaire d'utiliser une méthode d'optimisation nous permettant d'obtenir l'optimum global et ainsi estimer les paramètres réels du modèle et non un optimum local. Cependant, ces pistes n'ont pas été explorées dans cette thèse à cause de la quantité de calculs nécessaires pour exécuter ces algorithmes, notamment sur les cibles embarquées existant dans l'IoT.

2.1.4 Authentification d'un émetteur radio : approche *model-based*

Dans les sous-sections précédentes, nous avons proposé un modèle d'imperfections en bande de base d'un émetteur, ainsi que les techniques d'estimation associées. Dans cette section, nous allons nous servir de ce modèle et des techniques introduites précédemment pour authentifier un appareil à l'aide d'un algorithme de RFF *model-based*. Cependant, contrairement aux approches de la littérature ne prenant en compte qu'un ou deux défauts [16], [49], nous prendrons en compte davantage de défauts : le décalage I/Q (A_I, A_Q) et le déséquilibre I/Q (ϵ, θ), voire le décalage fréquentiel (Δf).

Dans cette section, nous prendrons le cas d'une modulation M-PSK, nous permettant notamment d'utiliser la technique de récupération de porteuse précédemment introduite dans la section 2.1.3.

2.1.4.1 Principe de fonctionnement de l'approche *model-based* proposée

Comme nous l'avons expliqué dans l'état de l'art, une approche *model-based* en authentification par empreinte radio (voir figure 2.12) consiste à estimer les imperfections d'un émetteur à l'aide d'un modèle d'imperfections et à se servir des paramètres estimés comme de descripteurs pendant la phase d'authentification. Le principe de cette approche est de se baser sur une modélisation physique du problème, notamment des imperfections des composants d'un émetteur.

Notre approche est composée de 3 étapes distinctes :

— Modélisation : la modélisation utilisée dans cette sous-section, présentée en figure 2.13, est basée sur le modèle d'imperfections des défauts des CNA et du mélangeur I/Q introduit en figure 2.6. Notre approche consiste donc à estimer les paramètres suivants : A_I, A_Q, ϵ et θ. Cependant, il serait aussi possible de prendre en compte le décalage fréquentiel Δf.

— Estimation : l'estimation des paramètres du modèle d'imperfections est réalisée à l'aide de l'algorithme d'estimation présenté en section 2.1.3.1. Cet algorithme a l'avantage de présenter une faible complexité calculatoire, car il s'appuie sur une descente de gradient stochastique. De plus, l'estimation et la correction du décalage fréquentiel peuvent aussi être réalisés avec la technique de récupération de porteuse à la puissance M précédemment introduite.

— Décision : les paramètres estimés lors de l'étape précédente sont utilisés comme des

descripteurs dans cette partie. La phase de décision est constituée de trois sous-étapes décrites dans la figure 2.14 : la détection d'anomalies, la classification et le partitionnement (*clustering*).

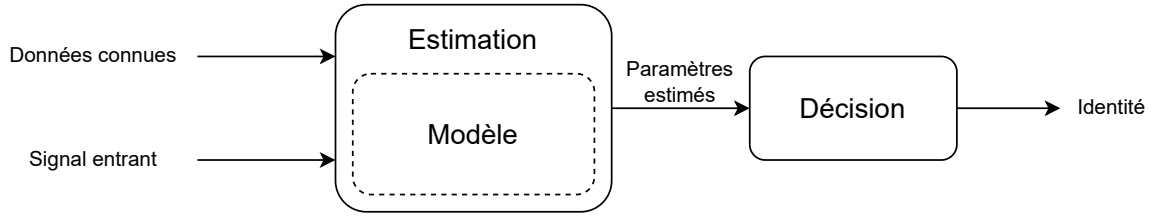


FIGURE 2.12 – Principe de fonctionnement de l'approche *model-based* proposée

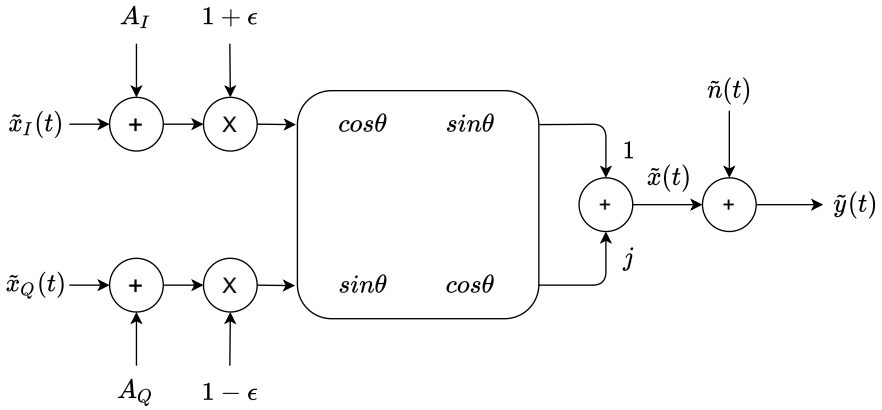


FIGURE 2.13 – Modèle d'imperfections de l'approche *model-based* proposée

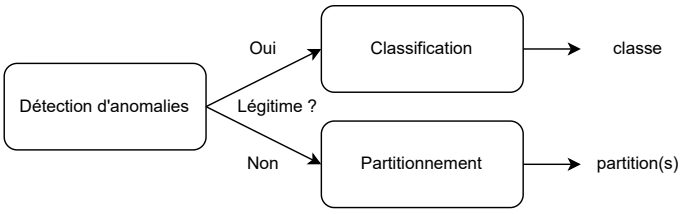


FIGURE 2.14 – Phase de décision de l'approche *model-based* proposée

2.1.4.2 Hypothèses du modèle

Les approches *model-based* basées sur un modèle d'imperfections requièrent plusieurs hypothèses (classées par ordre d'importance) pour garantir leur validité :

— Hypothèse 1 : l'architecture de l'émetteur est de type sans fréquence intermédiaire (figure 2.2) ou de type avec fréquence intermédiaire (figure 2.3).
— Hypothèse 2 : une hypothèse très importante est la disponibilité de données connues (préambule, pilote, ...) pour estimer les paramètres du modèle. Dans notre cas, nous supposerons l'existence d'un préambule comme source de données connues.
— Hypothèse 3 : les descripteurs sont supposés être générés par une loi normale multidimensionnelle.
— Hypothèse 4 : l'environnement de propagation et le récepteur sont considérés comme parfait, c'est-à-dire que le signal émis a été uniquement modifié par les imperfections de l'émetteur.
— Hypothèse 5 : les effets des non-linéarités sont négligeables sans le cas d'une M-PSK.
— Hypothèse 6 : la probabilité d'émission de chaque émetteur est identique.

2.1.4.3 Décision

Comme nous l'avons évoqué dans l'état de l'art, l'utilisation d'algorithme de RFF implique l'hypothèse «monde ouvert», i.e. il existe C émetteurs légitimes connus ainsi qu'un nombre inconnu d'émetteurs illégitimes. Ainsi, dans notre approche, les défauts des émetteurs légitime sont modélisés à l'aide d'un mélange de gaussiennes ou *Gaussian Mixture Model* (GMM) :

$$g(\vec{X}) = \sum_{c=1}^{C} \pi_c f(\vec{X}|\mu_c, \Sigma_c) \tag{2.37}$$

avec :

— $\vec{X}$: les paramètres estimés.
— $g(\vec{X})$: la densité de probabilité du mélange de gaussiennes.
— π_c : la probabilité *a priori* que la donnée appartienne à la classe k.
— $f(\vec{X}|\mu_c, \Sigma_c)$: la densité de probabilité d'une loi normale multidimensionnelle avec μ_c le vecteur d'espérance et Σ_c la matrice de covariance.

Il est important de souligner que la détection d'anomalies et la classification étant, dans notre cas, basées sur une approche supervisée, les paramètres du modèle seront appris grâce aux estimateurs suivants :

— $\tilde{\pi}_c = \frac{1}{C}$
— $\tilde{\mu_c} = \frac{1}{N_c} \sum_{i=n}^{N} \vec{X_n} \delta(y_n, c)$
— $\tilde{\Sigma_c} = \frac{1}{N_c - 1} \sum_{n=1}^{N} (\vec{X_n} - \tilde{\mu_c})^T (\vec{X_n} - \tilde{\mu_c}) \delta(y_n, c)$

avec :
— $\vec{X_i}$: le vecteur de caractéristiques correspondant au i-ème signal.
— y_i : le label associé au i-ème signal.
— $N_c = \sum_n \delta(y_n, c)$: le nombre de signaux correspondant à la classe c.
— N : le nombre total de signaux.

Détection d'anomalies L'approche utilisée pour la détection d'anomalies est similaire à celle présentée dans [33] et consiste à comparer la vraisemblance des paramètres estimés à un seuil préalablement appris. Plus particulièrement, le seuil est fixé en fonction d'un percentile de la vraisemblance des données d'entraînement, comme indiqué dans [31].

Classification La phase de classification est basée sur une approche par maximum de vraisemblance, identique à l'approche par maximum *a posteriori* dans notre cas puisque les probabilités d'appartenance *a priori* sont considérées uniformes, étant donné que la probabilité d'émission d'un émetteur est la même pour tous les émetteurs.

Ainsi, la règle utilisée pour la classification sera la suivante [86] :

$$n = arg\,max_c\, f(\vec{X}|\mu_c, \Sigma_c) \tag{2.38}$$

Clustering Concernant le partitionnement, la phase de *clustering* consiste à partitionner les paramètres estimés qui ont été considérés comme anormaux lors de la phase de détection d'anomalies. Nous avons donc utilisé une approche bayésienne nommée *Infinite Gaussian Mixture Mode* [31], [41] et basée sur un modèle GMM. Ainsi, cette approche peut permettre de déterminer *a posteriori* le nombre d'émetteurs illégitimes, ainsi que les messages associés à chacun et donc de connaître les intentions des attaquants lors d'une cyberattaque.

2.1.4.4 Simulations

Pour les différentes simulations présentées ci-après, les imperfections des différents émetteurs ont été simulées grâce au modèle d'imperfections proposé, avec les paramètres suivants :
— $A_I \sim U(-0.05, 0.05)$: partie réelle du décalage I/Q [1].
— $A_Q \sim U(-0.05, 0.05)$: partie imaginaire du décalage I/Q.

1. U étant la loi uniforme.

— $\epsilon \sim U(-0.05, 0.05)$: déséquilibre en gain.
— $\theta \sim U(-0.05, 0.05)$: déséquilibre en phase.

Une première simulation a été réalisée pour évaluer la phase de détection d'anomalies, de classification et de partitionnement. Pour cela, nous avons généré un préambule dépendant d'une forme d'onde correspondant à une modulation en quadrature de phase ou *Quadrature Phase Shift Keying* (QPSK). Nous avons considéré le cas de 10 émetteurs légitimes et 10 émetteurs illégitimes. Plus particulièrement, nous avons généré 50 signaux par émetteur où seule la réalisation du bruit est différente pour les signaux, avec un rapport signal à bruit ou *Signal-to-Noise Ratio* (SNR) de 30 dB. Ensuite, pour chaque signal, les paramètres du modèle d'imperfections ont été estimés. Pour l'entraînement du modèle probabiliste, les 500 exemples ont été séparés en 70 % - 30 % : 350 exemples d'entraînement (70 % des exemples) et 150 exemples de test (30 % des exemples). Pour ce qui est de la phase de détection d'anomalies, le seuil a été fixé à 95 %.

Les résultats de l'expérimentation ont montré que toutes les signatures correspondant aux émetteurs illégitimes ont été détectés comme illégitimes alors que seulement 8 exemples du jeu de test ont été détectés comme des anomalies (5.33 %). Concernant la classification, la justesse (*accuracy*, [86]) est de 100 %, c'est-à-dire que tous les exemples de test correspondant aux émetteurs légitimes ont été correctement prédits. Enfin, concernant la phase de partitionnement, tous les signaux des émetteurs illégitimes ont été correctement partitionnés.

Une seconde simulation a été réalisée pour évaluer les performances de la phase de classification en fonction de valeurs de SNR variables allant de 0 à 30 dB. Pour cela, nous avons généré un préambule dépendant d'une forme d'onde QPSK. Nous avons considéré le cas de 10 émetteurs avec 50 signaux par émetteur où seule la réalisation du bruit est différente pour les signaux. Ensuite, pour chaque signal, les paramètres du modèle d'imperfections ont été estimés. Pour l'entraînement du modèle probabiliste, les 500 exemples ont été séparés en 70 % - 30 % : 350 exemples d'entraînement (70 % des exemples) et 150 exemples de test (30 % des exemples).

Les performances de l'algorithme sont présentées en figure 2.15. De plus, l'espace des descripteurs pour un SNR de 30 dB est présenté en figure 2.16. Nous pouvons observer que les performances de l'algorithme sont proches de 100% pour des niveaux de SNR fort ($>$ 20 dB). En effet, comme nous pouvons le voir sur la figure 2.16 les émetteurs sont relativement bien séparables dans l'espace des descripteurs pour des SNRs forts. Cependant, les performances de l'algorithme sont fortement impactées lorsque le niveau

de SNR devient faible.

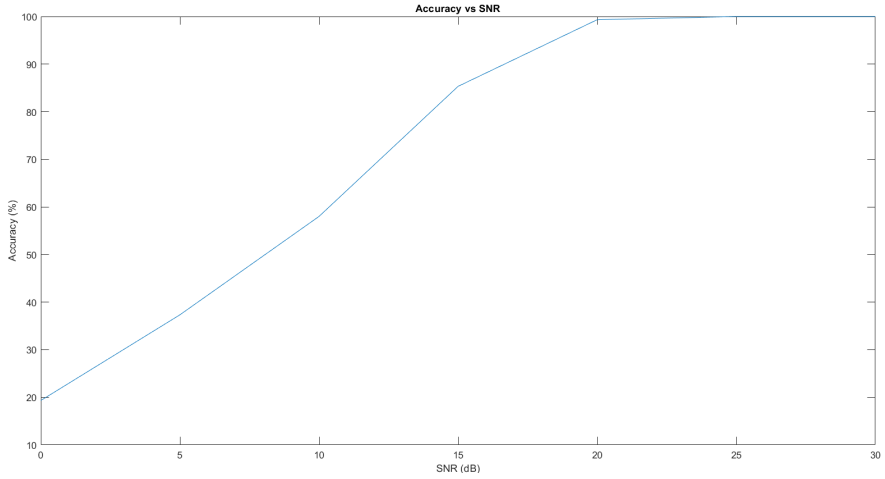


FIGURE 2.15 – Performances de l'approche *model-based* proposée

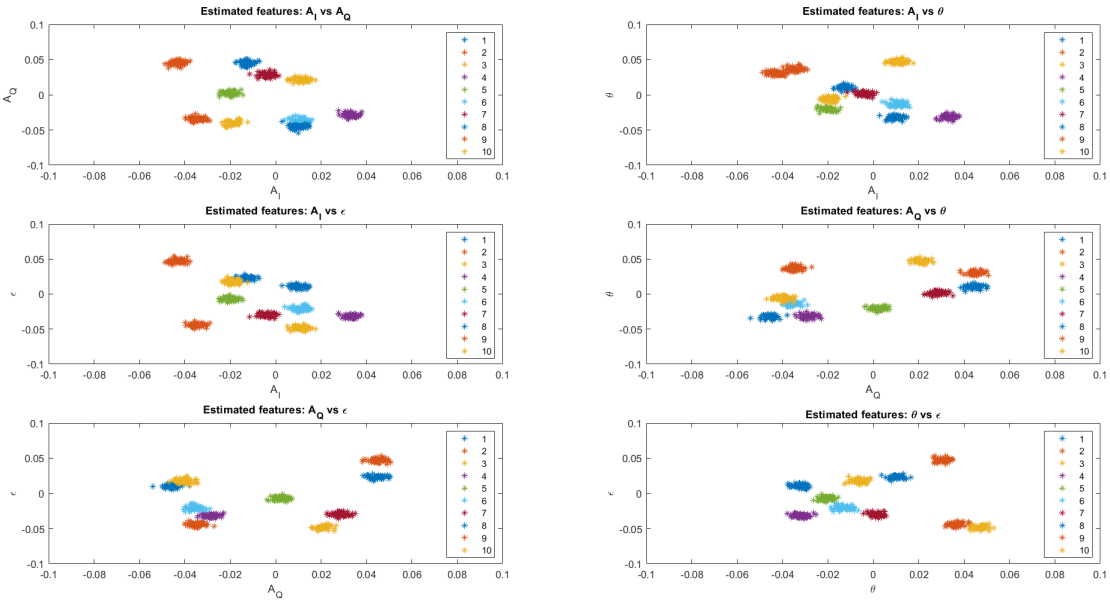


FIGURE 2.16 – Paramètres estimés à partir du modèle d'imperfections de l'approche *model-based proposée*

Des tests en conditions réelles avec des plateformes SDR Adalm-PLUTO [87] ont été réalisés, mais ils n'ont pas donné les résultats escomptés. En effet, nous n'avons pas obtenu les mêmes performances que lors de nos simulations, même avec un SNR de 30 dB. Une des raisons de ce manque de performance semble être le fait que l'Adalm-PLUTO corrige les décalage I/Q, ainsi que le déséquilibre I/Q en interne, comme indiqué dans [88].

2.2 Le canal de propagation : le cas de la modulation Doppler

Le canal de propagation correspond à l'environnement de propagation entre l'émetteur et le récepteur. Il dépend à la fois de différents paramètres comme la distance ou la vitesse relative entre l'émetteur et le récepteur. Les différents types d'affaiblissement du canal de propagation sont les suivants [57], [76] :

- Affaiblissement à large échelle :
  - Affaiblissement de parcours (*path loss*) : affaiblissement de la puissance du signal dû à l'évolution de la distance entre l'émetteur et le récepteur.
  - Effet de masque (*shadowing*) : affaiblissement de la puissance du signal causé par les obstacles (bâtiments, ...).
- Affaiblissement à faible échelle (*fading*) : affaiblissement de la puissance du signal dû aux multi-trajets ainsi qu'à l'effet Doppler.

2.2.1 Hypothèses pour la modélisation de la modulation Doppler

Dans ce manuscrit, nous nous intéresserons à la modélisation de l'effet de modulation Doppler sur des signaux de radiocommunications, i.e. l'effet d'un mouvement relatif entre l'émetteur et le récepteur sur le signal reçu. D'autres modélisations permettant de modéliser l'effet Doppler dans un contexte de multi-trajets existent dans la littérature, que ce soit avec des modèles déterministes [89]-[91] ou même probabilistes [92]-[95].

Dans notre cas, nous allons introduire l'effet de la modulation Doppler sur un signal de communication bande-étroite, ainsi que sur sa représentation en bande de base. La modulation Doppler est l'effet d'un mouvement complexe (par rapport à une translation linéaire) sur un signal de communication [96]. Plus particulièrement, nous allons nous intéresser à la modélisation du micro-Doppler causée par une vibration $m(t)$ au niveau de l'émetteur. Ces travaux ont fait l'objet d'une publication dans MDPI Remote Sensing en 2022 [97].

Le principe de cette démonstration consiste à représenter la situation émetteur/récepteur à l'aide d'un modèle géométrique (voir figure 2.17) et à démontrer les effets du mouvement de l'émetteur sur le signal reçu par le récepteur. Notre modèle géométrique est basé sur la théorie du lancer de rayon, c'est-à-dire que nous considérons que la propagation

du signal est modélisée par des rayons. Ces rayons sont les trajectoires perpendiculaires aux fronts d'ondes (surface de phase constante), correspondant à la direction de propagation de l'onde. Selon Tse et al. [76], cette théorie est une bonne approximation de la propagation d'une onde dans l'espace.

Nous allons reprendre les notations provenant de notre article [97], nous permettant notamment de réutiliser les développements présentés dans ses annexes. Ainsi, nous considérerons le cas d'un signal à bande-étroite $s(t)$ et de sa représentation en bande de base $\tilde{s}(t)$, envoyé par un émetteur TX à destination d'un récepteur RX. De plus, nous allons considérer $\vec{l}(t)$ comme étant le vecteur distance entre l'émetteur et le récepteur à l'instant t et $\vec{x}(t)$ le mouvement de l'émetteur réalisé depuis $t = 0s$. Enfin, nous avons repris la même organisation que dans notre article, pour faciliter la comparaison.

Le signal est envoyé à un temps t par l'émetteur (TX) à l'aide d'une antenne isotrope et il est reçu à l'instant $t + \tau(t)$ par le récepteur (RX), considéré comme fixe, à l'aide d'une antenne isotrope :

$$r(t + \tau(t)) = A(t)s(t) \tag{2.39}$$

avec :

- $A(t)$: l'atténuation du signal dépendant de la distance $\left\|\vec{l}(t)\right\|$ selon l'équation de Friis [57].
- $\tau(t) = \frac{\left\|\vec{l}(t)\right\|}{c}$: le retard du signal dépendant du temps t.

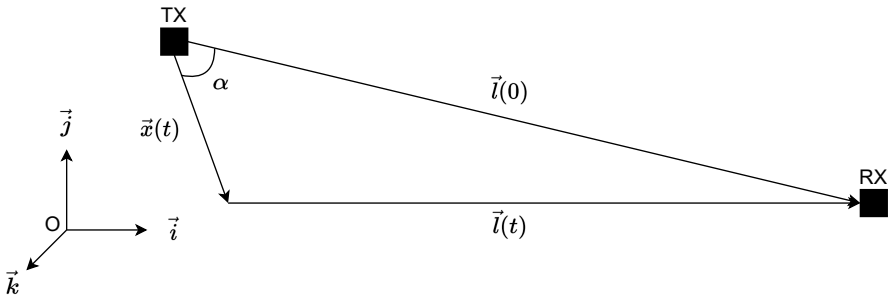


FIGURE 2.17 – Modèle géométrique de la modulation Doppler

Pour démontrer l'effet du mouvement $\vec{x}(t)$ sur le signal reçu $r(t)$, nous supposons les hypothèses suivantes :

- Hypothèse 1 : $\|\vec{x}(t)\| << \left\|\vec{l}(0)\right\|$, i.e. l'amplitude du mouvement $\|\vec{x}(t)\|$ est négligeable comparée à la distance initiale $\left\|\vec{l}(0)\right\|$ entre l'émetteur et le récepteur durant

le temps d'observation T du signal.

— Hypothèse 2 : $\forall(t,t_1) \in [0,T]^2, \|\tau(t-\tau(t_1))\| \approx \|\tau(t-\tau(0))\|$, i.e. le mouvement de l'émetteur pendant le trajet du signal n'a pas d'impact sur le retard de celui-ci.

— Hypothèse 3 : l'atténuation du signal $A(t)$ est considérée comme indépendante de $\|\vec{x}(t)\|$ et donc constante : $A(t) \approx A(0)$.

2.2.2 Modélisation de la modulation Doppler

Maintenant que nous avons formulé nos hypothèses nécessaires à l'étude de la modulation Doppler, nous allons nous baser sur l'équation 2.39 et, en effectuant le changement de variable suivant $t_1 = t + \tau(t)$, nous obtenons la relation suivante :

$$r(t_1) = A(t_1 - \tau(t))\, s(t_1 - \tau(t)) \tag{2.40}$$

$$= A(t_1 - \tau(t_1 - \tau(t)))\, s(t_1 - \tau(t_1 - \tau(t))) \tag{2.41}$$

Ensuite, en utilisant l'hypothèse 2 et en remplaçant t_1 par t pour des soucis de notation, nous obtenons :

$$r(t) = A(t - \tau(t - \tau_0))s(t - \tau(t - \tau_0)) \tag{2.42}$$

avec $\tau_0 = \tau(0)$: le retard initial

Ainsi, en utilisant le même développement que dans l'annexe A de [97] [2], nous obtenons dans un premier temps :

$$\left\|\vec{l}(t-\tau_0)\right\| \approx \left\|\vec{l}(0)\right\| - \langle \vec{x}(t-\tau_0), \vec{l_0}\rangle \tag{2.43}$$

Il est alors possible exprimer $\tau(t-\tau_0)$ en fonction de l'approximation de $\left\|\vec{l}(t-\tau_0)\right\|$ et en utilisant l'hypothèse 3, cela nous permet de simplifier la forme du signal reçu :

$$r(t) = As(t - \tau_0 + \frac{\langle \vec{x}(t-\tau_0), \vec{l_0}\rangle}{c}) \tag{2.44}$$

avec :

2. La démonstration de notre article est légèrement différente de celle-ci. En effet, nous considérions que le signal reçu était $r(t) = A(t-\tau(t))s(t-\tau(t))$ [76], ainsi que des hypothèses différentes. La modélisation proposée dans cette section est plus généraliste que celle proposée dans [97], car permettant de prendre en compte l'effet du retard τ_0 sur le terme de la modulation Doppler.

— $A = A(0)$: un terme d'atténuation constant.
— $\vec{l_0} = \frac{\vec{l}(0)}{\|\vec{l}(0)\|}$: un vecteur unitaire.

Sachant que $s(t)$ est un signal de communications bande-étroite, nous pouvons développer $r(t)$ en faisant apparaître la modulation d'amplitude $a(t)$ et de phase/fréquence $\varphi(t)$:

$$r(t) = Aa(t - \tau_0 + \frac{\langle \vec{x}(t-\tau_0), \vec{l_0} \rangle}{c}) \\ cos(2\pi f_0 t + k\langle \vec{x}(t-\tau_0), \vec{l_0} \rangle + \varphi(t - \tau_0 + \frac{\langle \vec{x}(t-\tau_0), \vec{l_0} \rangle}{c}) - \phi_0) \quad (2.45)$$

avec :

— $\phi_0 = 2\pi f_0 \tau_0$: la phase initiale.
— $k = \frac{2\pi}{\lambda}$: le nombre d'onde.

Il est ainsi possible d'obtenir sa représentation en bande de base :

$$\tilde{r}(t) = Aa(t - \tau_0 + \frac{\langle \vec{x}(t-\tau_0), \vec{l_0} \rangle}{c}) e^{j(k\langle \vec{x}(t-\tau_0), \vec{l_0} \rangle + \varphi(t - \tau_0 + \frac{\langle \vec{x}(t-\tau_0), \vec{l_0} \rangle}{c}) - \phi_0)} \quad (2.46)$$

$$= A\tilde{s}(t - \tau_0 + \frac{\langle \vec{x}(t-\tau_0), \vec{l_0} \rangle}{c}) e^{-j\phi_0} e^{jk\langle \vec{x}(t-\tau_0), \vec{l_0} \rangle} \quad (2.47)$$

Pour le reste de cette démonstration, nous allons considérer le mouvement $\vec{x}(t) = \vec{v}t + \vec{m}(t)$ comme la somme d'une translation linéaire dépendant d'un vecteur $\vec{v}$ et d'un micro-mouvement $\vec{m}(t)$ dû à une vibration périodique de faible amplitude. De plus, nous écrirons [3] $\langle \vec{x}(t), \vec{l_0} \rangle = (vt + m(t))cos(\alpha)$, ce qui simplifie la représentation en bande de base du signal reçu :

$$\tilde{r}(t) = A\tilde{s}(\xi(t - \tau_0) + \frac{m(t-\tau_0)}{c} cos(\alpha)) e^{-j\phi_0} e^{jk\langle \vec{x}(t-\tau_0), \vec{l_0} \rangle} \quad (2.48)$$

avec $\xi = (1 + \frac{v}{c} cos(\alpha))$, le facteur de dilatation/compression Doppler [90].

D'une part, nous pouvons remarquer la présence d'un terme de modulation Doppler $e^{jk\langle \vec{x}(t-\tau_0), \vec{l_0} \rangle}$ sur le signal reçu. En utilisant la décomposition du mouvement $x(t)$ comme une combinaison d'une translation vt et d'une vibration $m(t)$, le terme $e^{j2\pi f_d t}$ (avec $f_d = \frac{v}{c} f_0 cos(\alpha)$) correspond au décalage Doppler et le terme $e^{jkm(t)cos(\alpha)}$ dépendant de la vibration $m(t)$ est appelé terme de micro-Doppler. D'autre part, nous pouvons

3. Pour des soucis de simplicité, nous considérons les deux vecteurs dépendant de la même direction, mais il est possible d'étendre cette démonstration à une somme de mouvement $\vec{x}(t) = \sum_{i=1}^{N} \vec{x}(t)$.

remarquer qu'en plus de l'effet de modulation Doppler, l'enveloppe complexe du signal émis est modifiée par deux termes dépendant du mouvement : ξ et $\frac{m(t)}{c}$. Ainsi, le terme $\xi = (1+\frac{v}{c}cos(\alpha))$ dépend de la translation linéaire et le terme $\frac{m(t)}{c}cos(\alpha)$ est appelé micro-gigue Doppler. Comme indiqué dans [76], le facteur de dilatation/compression Doppler ξ peut généralement être négligé si f_d est petit par rapport à la largeur de bande B. Le terme de micro-gigue Doppler peut également être négligé. En effet, comme nous considérons $m(t)$ comme une vibration de faible amplitude, il est raisonnable de considérer que la micro-gigue résultante de cette vibration est négligeable par rapport au bruit de phase.

Ainsi, une première approximation du signal, consistant à négliger le terme de micro-gigue comme dans [97], est donnée par :

$$\tilde{r}(t) \approx \tilde{s}(\xi(t-\tau_0))e^{-j\phi_0}e^{jk\langle\vec{x}(t-\tau_0),\vec{l_0}\rangle} \tag{2.49}$$

De plus, comme nous l'avons vu précédemment, le facteur de dilatation/compression Doppler ξ peut aussi être généralement négligé [76], ce qui nous permet d'obtenir :

$$\tilde{r}(t) \approx \tilde{s}(t-\tau_0)e^{-j\phi_0}e^{jk\langle\vec{x}(t-\tau_0),\vec{l_0}\rangle} \tag{2.50}$$

Le modèle obtenu est assez proche du modèle de Chen et al. pour le domaine du radar [96], où le phénomène de micro-Doppler est largement étudié. Cependant, son modèle ne prend pas en compte le phénomène de dilatation/compression Doppler, ni même celui de micro-gigue Doppler.

Il serait possible d'étendre cette modélisation à une configuration de multi-trajets en utilisant les travaux de Lyonnet [4]. En effet, en utilisant des sources virtuelles, cela permet de modéliser les différentes réflexions (L trajets). Ce modèle, prenant en compte l'effet de dilatation/compression Doppler, peut être décrit de la manière suivante :

$$\tilde{r}(t) = \sum_{n=1}^{L} a_n\tilde{s}(\xi_n(t-\tau_n))e^{jk\langle\vec{x}(t-\tau_n),\vec{l}_{n0}\rangle} \tag{2.51}$$

avec :

— a_n : l'amplitude complexe du n-ème trajet.

— $\xi_n = \frac{v}{c}cos(\alpha_n)$: le terme de dilatation/compression Doppler associé au n-ème

4. Lyonnet modélise aussi l'amplitude complexe $\alpha_i(t)$ associée à chaque trajet. Cependant, nous supposons qu'il n'y a pas de phénomène de *clutter* dans notre modélisation.

trajet.

— τ_n : le retard associé au n-ème trajet.

— $\vec{l}_{n0}$: le vecteur unitaire associé au n-ème trajet.

Pour mettre en évidence ce phénomène, nous avons réalisé une expérimentation montrant lumière l'effet d'une vibration $\vec{m(t)}$ au niveau de l'émetteur sur le signal reçu. Plus particulièrement, cette expérimentation a permis de mettre en évidence le phénomène de micro-Doppler lié à $\vec{m(t)}$, en négligeant l'effet d'une translation $\vec{v}$, qui a été largement étudié dans la littérature. La configuration de notre expérimentation, présentée en figure 2.18, est la suivante :

— Hypothèses :
  — Antenne isotrope transmettant une sinusoïde :
    $s(t) = cos(2\pi f_0 t + \Phi_{em})$
  — Micro-mouvement sinusoïdal :
    $m(t) = A_v sin(2\pi f_m t + \Phi_0)$
  — La représentation en bande de base du signal théorique :

$$\tilde{r}(t) = Ae^{j(\beta sin(2\pi f_m t + \Phi_0) + \Phi)} \ (\beta = \frac{2\pi A_v}{\lambda}) \tag{2.52}$$

$$= Ae^{j\Phi} \sum_{n=-\infty}^{+\infty} (e^{j\Phi_0})^n J_n(\beta) e^{j2\pi n f_m t} \tag{2.53}$$

— Système émetteur :
  — Générateur de sinusoïde : ANRITSU MG3692B
  — Antenne émettrice : Ettus VERT 2450
  — Générateur de vibration : WOVELOT 037606, tension nominal 1.5V
  — Centrale intertielle ou *Inertial Measurement Unit* (IMU) : SparkFun 9DoF Razor IMU M0
— Système récepteur :
  — Enregistreur de signal : Signal Hound BB60C
  — Antenne réceptrice : Ettus VERT 2450
— Paramètres de test :
  — Fréquence : 2.45 GHz (influant sur λ)
  — Tension du générateur de vibration : 1.5 V (influant sur A_v et f_0)
  — Distance : 2 m (influant sur A)

La figure 2.19 présente un ensemble de résultats de l'expérience décrite précédemment.

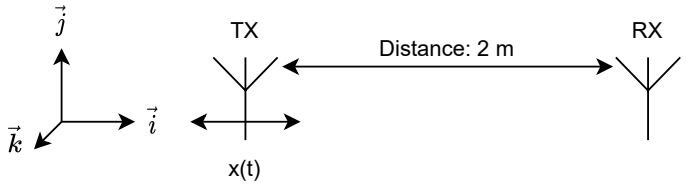
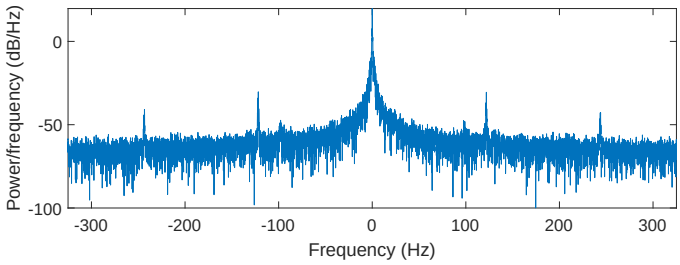


FIGURE 2.18 – Schéma du banc de test pour l'expérimentation sur le micro-Doppler

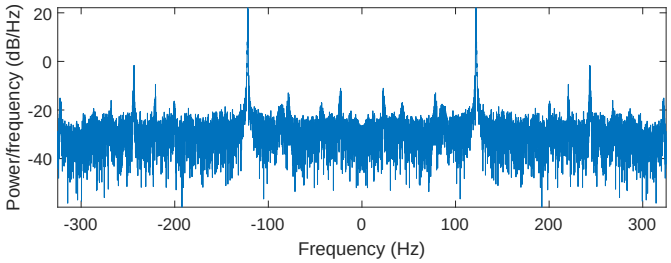
D'une part, la figure 2.19a correspond au spectre de la représentation bande de base du signal reçu et synchronisé, c'est-à-dire que le décalage fréquentiel a été corrigé. D'autre part, la figure 2.19b correspond au spectre du signal de l'IMU mesurant la vibration au niveau de l'antenne d'émission. Le modèle théorique prédit que les harmoniques du spectre de la représentation en bande de base du signal reçu sont espacées régulièrement de la valeur f_m. Nous pouvons observer que cet espacement est présent dans le spectre et nous pouvons également observer que l'espacement des harmoniques est le même que la fréquence de vibration ($f_m = 122$ Hz) mesurée par l'IMU (placé sur l'antenne d'émission). Cependant, nous pouvons noter que leurs amplitudes ne sont pas les mêmes que pour la prédiction du modèle, probablement parce que l'antenne utilisée n'est pas isotrope (modèle théorique) et que la vibration n'est pas une pure sinusoïde, puisqu'elle est composée de plusieurs harmoniques (voir figure 2.19b). En effet, la vibration est probablement plus proche d'une vibration carrée due à la technologie employée par le générateur de vibrations (vibreur).

2.2.3 Estimations du micro-Doppler

Dans cette partie, nous nous intéressons à l'estimation du terme de modulation Doppler $e^{jk\langle\vec{x}(t),\vec{l}_0\rangle}$, et plus particulièrement du terme micro-Doppler $e^{jk\langle\vec{m}(t),\vec{l}_0\rangle}$, qui s'apparente à une signature du mouvement $\vec{x}(t)$ sur le signal reçu. Comme nous l'avons évoqué précédemment, le micro-Doppler est un phénomène largement étudié dans le domaine du radar [96]. Cependant, à l'inverse du domaine du radar, dans le domaine des radiocommunications, le signal émis n'est pas connu, excepté pour la forme d'onde utilisée, ainsi que son protocole. Il est donc nécessaire d'annuler, à partir du signal reçu $\tilde{r}(t)$, le signal $\tilde{s}(t)$ (les données transmises) pour mieux exploiter le micro-Doppler. Par la suite, nous extrairons des informations sur le micro-Doppler, à l'aide d'une phase de détection puis d'estimation.



(a) Densité de puissance du signal reçu



(b) Densité spectrale du signal de l'IMU

FIGURE 2.19 – Résultats de l'expérimentation sur la mise en lumière du micro-Doppler en radiocommunications

2.2.3.1 Annulation du signal de communication

Pour la suite, dans le but de simplifier la notation, nous négligerons le retard initial τ_0, ainsi que la phase initiale ϕ_0 et nous écrirons $x(t) =< \vec{x}(t), \vec{l}_0 >$. Nous considérerons également la présence d'un bruit blanc gaussien additif et complexe $\tilde{n}(t)$. Ainsi, le modèle résultant sera le suivant :

$$\tilde{r}(t) = A\tilde{s}(t)e^{jkx(t)} + \tilde{n}(t) \tag{2.54}$$

Nous pouvons remarquer qu'il semble très difficile d'estimer le terme de modulation Doppler $e^{jkx(t)}$ à partir du terme $\tilde{s}(t)e^{jkx(t)}$, que ce soit dans le domaine temporel ou dans le domaine fréquentiel. Ainsi, il est nécessaire d'annuler le signal de communication $\tilde{s}(t)$ et d'extraire le terme de modulation Doppler. Pour cela, nous considérerons que le signal $\tilde{s}(t)$ est un signal M-PSK. La technique d'extraction ou d'annulation du signal que nous introduisons dans cette sous-section est basée sur la méthode de récupération de fréquence porteuse à la puissance M évoquée en section 2.1.3 et peut-être exprimée de la manière suivante :

$$\tilde{r}(t)^M = (A\tilde{s}(t)e^{jkx(t)} + \tilde{n}(t))^M \tag{2.55}$$

Il est possible d'utiliser la formule du binôme, pour reformuler l'équation précédente :

$$\tilde{r}(t)^M = \sum_{m=0}^{M} \binom{M}{m} A^m \tilde{s}(t)^m e^{jkmx(t)} \tilde{n}(t)^{M-m} \tag{2.56}$$

Comme indiqué dans l'appendice B de [97], si le SNR est assez élevé, nous obtenons l'approximation suivante :

$$\tilde{r}(t)^M \approx A^M e^{jkMx(t)} + MA^{M-1} \tilde{s}(t)^{(M-1)} e^{jk(M-1)x(t)} \tilde{n}(t) \tag{2.57}$$

$$\approx A^M e^{jkMx(t)} + \tilde{n}_1(t) \tag{2.58}$$

Nous pouvons observer que le signal résultant, consiste en la somme d'un terme $e^{jkMx(t)}$ dépendant de la modulation Doppler, ainsi que d'un bruit blanc additif et complexe $\tilde{n}_1(t)$ (voir appendice B de [97]).

Pour mettre en évidence la technique d'extraction, nous avons réalisé une seconde expérimentation inspirée de l'expérience précédente (voir figure 2.18). La configuration de notre expérimentation est la suivante :

— Hypothèses :
  — Antenne isotrope transmettant un signal 2-PSK :
    $\tilde{s}(t) = \sum_{k=-\infty}^{+\infty} a_k \pi(\frac{t-kT_0}{T_0})$
  — Micro-mouvement sinusoidal :
    $m(t) = A_v sin(2\pi f_m t + \Phi_0)$
  — La représentation en bande de base du signal théorique :

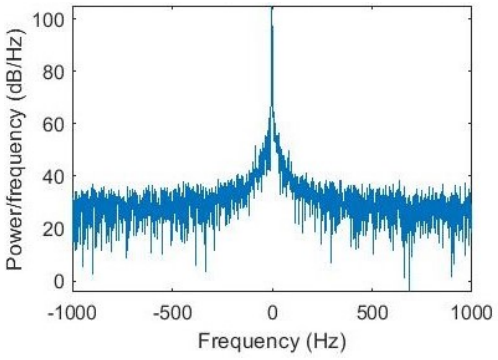
$$\tilde{r}(t) = Ae^{j(\beta sin(2\pi f_m t + \Phi_0) + \Phi)} \ (\beta = \frac{2\pi A_v}{\lambda}) \tag{2.59}$$

$$= Ae^{j\Phi} \sum_{n=-\infty}^{+\infty} (e^{j\Phi_0})^n J_n(\beta) e^{j2\pi n f_m t} \tag{2.60}$$

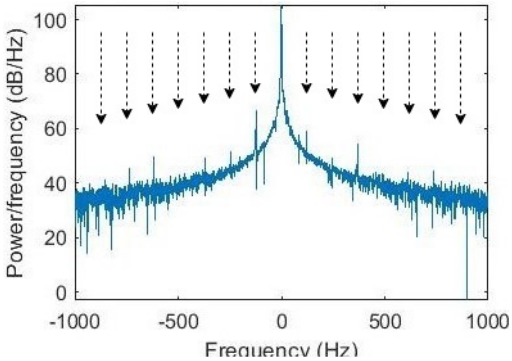
— Système émetteur :
  — Générateur de sinusoïde : ANRITSU MS2830A
  — Antenne émettrice : Ettus VERT 2450
  — Générateur de vibration : WOVELOT 037606, tension nominal 1.5V
  — IMU : SparkFun 9DoF Razor IMU M0
— Système récepteur :
  — Enregistreur de signal : Signal Hound BB60C

  - Antenne réceptrice : Ettus VERT 2450
- Paramètres de test :
  - Fréquence : 2.45 GHz (influant sur λ)
  - Tension du générateur de vibration : 1.5 V (influant sur A_v et f_m)
  - Distance : 2 m (influant sur A)

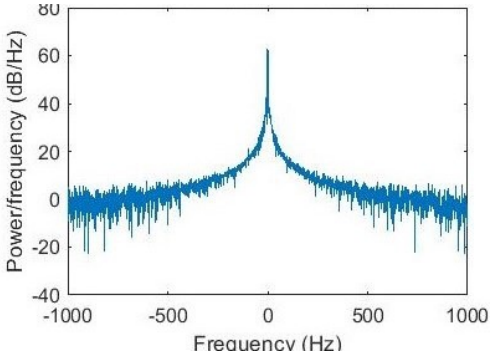
Nous pouvons observer dans la figure 2.20 qu'il est possible d'extraire le micro-Doppler à partir d'un signal BPSK en utilisant le traitement proposé. Pour cette expérience, nous avons utilisé un signal de référence qui est une sinusoïde, ce qui permet de relier certains résultats de cette expérience à ceux de l'expérience précédente. D'une part, la figure 2.20b correspond au signal de référence (sinusoïde) sans vibration et la figure 2.20a correspond au signal de référence (sinusoïde) avec vibrations. D'autre part, la figure 2.20c correspond au signal BPSK traité sans vibration et la figure 2.20d correspond au signal BPSK traité en présence de vibrations. Ainsi, à partir de la figure 2.20d, nous pouvons confirmer qu'il est bien possible d'extraire le micro-Doppler à partir d'un signal BPSK.



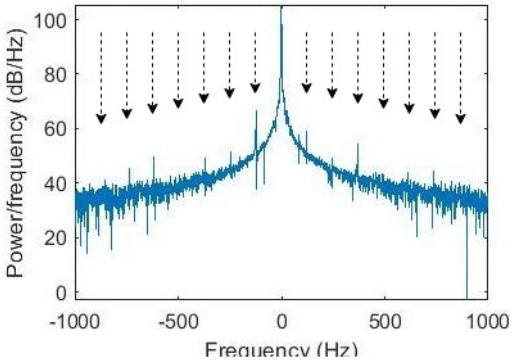
(a) Sinusoïde sans micro-Doppler



(b) Sinusoïde avec micro-Doppler



(c) Signal BPSK sans micro-Doppler ($\tilde{r}(t)^2$)



(d) Signal BPSK avec micro-Doppler ($\tilde{r}(t)^2$)

FIGURE 2.20 – Résultats de l'expérimentation sur l'extraction du micro-Doppler en radiocommunications

2.2.3.2 Modèle de signal

Maintenant que nous avons introduit une méthode permettant d'extraire le terme de modulation Doppler à partir du signal reçu pour le cas des signaux M-PSK, nous allons nous intéresser à l'étude du terme de modulation Doppler $e^{jkx(t)}$. Pour cela, il est nécessaire de définir un modèle de signal qui nous permettra de proposer, par la suite, des traitements adaptés.

La représentation en bande de base du signal extrait est le suivant :

$$\tilde{y}(t) = ae^{jkx(t)}e^{j2\pi\Delta ft} + \tilde{n}(t) \tag{2.61}$$

$$= ae^{jkm(t)}e^{j2\pi f_1 t} + \tilde{n}(t) \tag{2.62}$$

avec :

- a : un terme d'amplitude complexe.
- $x(t)$: le mouvement de l'émetteur suivant les hypothèses de décomposition $x(t) = vt + m(t)$.
- f_1 : une fréquence modélisant à la fois les effets potentiels d'un effet Doppler (f_d) dû à une translation dépendant de v, ainsi que d'un potentiel décalage fréquentiel entre l'émetteur et le récepteur (Δf).
- $\tilde{n}(t)$: un bruit blanc complexe centré gaussien ($\tilde{n}(t) \sim CN(0, \sigma^2)$).

Ainsi, il est possible de décomposer le terme de micro-Doppler par une série de Fourier du fait de la caractéristique périodique (f_m) du signal $m(t)$ [98] :

$$e^{jkm(t)} = \sum_{n=-\infty}^{+\infty} b_n e^{j2\pi n f_m t} \tag{2.63}$$

avec b_n : le coefficient de Fourier ($b_n = \int_{\frac{-1}{2f_m}}^{\frac{1}{2f_m}} e^{jkm(t)}e^{-j2\pi n f_m t}dt$).

Nous obtenons donc le modèle de signal suivant :

$$\tilde{y}(t) = a(\sum_{n=-\infty}^{+\infty} b_n e^{j2\pi n f_m t})e^{j2\pi f_1 t} + \tilde{n}(t) \tag{2.64}$$

$$= \sum_{n=-\infty}^{+\infty} a_n e^{j2\pi(n f_m + f_1)t} + \tilde{n}(t) \tag{2.65}$$

avec $a_n = a \times b_n$: un coefficient complexe.

Ainsi que sa transformée de Fourier :

$$\tilde{Y}(f) = \sum_{n=-\infty}^{+\infty} b_n \, \delta(f - (nf_m + f_1)) + \tilde{N}(f) \tag{2.66}$$

avec $\tilde{N}(f)$: la représentation fréquentielle du bruit blanc complexe centré et gaussien.

Ce modèle de signal va permettre, par la suite, de proposer une méthode de détection du micro-Doppler dans un signal observé, ainsi qu'une méthode permettant d'estimer les différents paramètres f_1, f_m et les $\{a_n\}_{n\in\mathbb{Z}}$.

2.2.3.3 Détection du micro-Doppler

Avant de procéder à l'estimation des paramètres du modèle de signal, il est nécessaire d'effectuer un test d'hypothèse pour détecter la présence de micro-Doppler dans un signal observé. Ainsi, la première étape de la conception de ce test consiste à formuler deux hypothèses :

— H_0 : $ae^{j2\pi f_1 t} + \tilde{n}(t)$: absence de micro-Doppler

— H_1 : $\sum_{n=-\infty}^{+\infty} a_n e^{j2\pi(nf_m+f_1)t} + \tilde{n}(t)$: présence de micro-Doppler

Pour réaliser cette phase de détection, nous avons choisi d'utiliser le test d'hypothèse utilisé pour la détection de signaux cyclostationnaires dans [99], [100]. Cependant, les hypothèses précédentes ne correspondent pas à celles requissent pour ce test, car ne prenant pas en compte le terme de décalage fréquentiel $e^{j2\pi f_1 t}$. Il est donc nécessaire d'effectuer plusieurs transformations (voir figure 2.21) pour faire correspondre les hypothèses précédentes aux hypothèses requises par la méthode utilisée dans [99], [100].

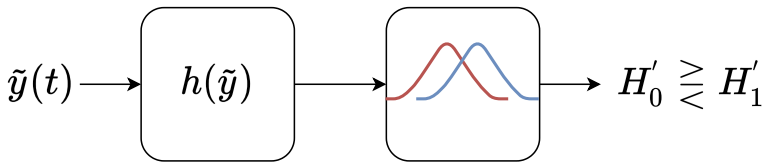


FIGURE 2.21 – Schéma de la méthode de détection du micro-Doppler

Dans notre cas, la première transformation (synchronisation) est la suivante :

$$f(x) = x(t)e^{-j2\pi\hat{f}_1 t} \tag{2.67}$$

Cela permet de supprimer le décalage fréquentiel dû à la fréquence f_1. Cette transformation est basée sur l'estimation (au préalable) de la fréquence f_1 (qui peut être réutilisé

lors de la phase d'estimation pour l'approche proposée).

Les hypothèses résultantes sont les suivantes :

$$H_0 \overset{h(\tilde{y})}{\rightarrow} H_0^{'} : a + \tilde{n}^{'}(t) \tag{2.68}$$

$$H_1 \overset{h(\tilde{y})}{\rightarrow} H_1^{'} : \sum_{n=-\infty}^{+\infty} a_n e^{j2\pi n f_m t} + \tilde{n}^{'}(t) \tag{2.69}$$

avec $\tilde{n}'(t) = \tilde{n}(t)e^{-j2\pi\hat{f}_1 t}$: un bruit blanc gaussien complexe (AWGN).

Si l'on considère que la vibration $m(t)$ est périodique, alors le terme de micro-Doppler $e^{jkm(t)}$ possède des propriétés de cyclostationnarité [98]. Ainsi, nous allons utiliser le test d'hypothèse utilisé dans [99], [100] pour détecter la présence de micro-modulation dans le signal extrait. Ce test d'hypothèse repose sur la statistique suivante :

$$C_I = \frac{I(\alpha)}{\sqrt{\frac{1}{N}\sum_{\alpha=\alpha_{min}}^{\alpha_{max}} I(\alpha)^2}} \tag{2.70}$$

Le terme $I(\alpha)$ est le *Cycle frequency Domain Profile* (CDP), décrit dans [99]. Il s'agit d'un outil, obtenu à l'aide de la cohérence spectrale du signal [99], permettant d'étudier les propriétés de cyclostationnarité d'un signal.

avec le seuil C_{TH} associé correspondant à l'hypothèse nulle pour $\mathbb{P}(\hat{H}_1|H_0) = 0.1$:

$$C_{TH} = \frac{max(I(\alpha))}{\sqrt{\frac{1}{N}\sum_{\alpha=\alpha_{min}}^{\alpha_{max}} I(\alpha)^2}} \tag{2.71}$$

Le test d'hypothèse est réalisé de la manière suivante :

$$\begin{aligned} C_I \leq C_{TH} &: déclarer\, H_0 \\ C_I > C_{TH} &: déclarer\, H_1 \end{aligned} \tag{2.72}$$

Une centaine de simulations ont été réalisées pour comparer les performances dépendant de différents temps d'observation T et de différentes valeurs de SNR avec une fréquence d'échantillonnage de 1 000 Hz (voir Figure 2.22). Le signal utilisé pour modéliser le micro-Doppler est un signal sinusoïdal modulé en fréquence multiplié par une composante de décalage en fréquence et en présence d'un bruit blanc gaussien additif et complexe $\tilde{n}(t)$. La raison qui nous a poussé à utiliser ce type de signal est principalement parce qu'il correspond à un sous-cas du modèle de signaux précédemment introduit dans l'équation 2.65. De plus, tout au long de cette étude, nous avons considéré une vibra-

tion sinusoïdale pour les micro-mouvements. Enfin, nous avons utilisé la simulation, car cela rend l'expérience plus contrôlable et reproductible. Ainsi, le signal utilisé pour la simulation est le suivant :

$$\tilde{r}(t) = e^{j\beta sin(2\pi f_m t} e^{j\pi f_1 t} + \tilde{n}(t) \tag{2.73}$$

avec :

- $\beta = 0.1$: l'indice de modulation.
- $f_m \in [30, 150]$: la fréquence de vibration.
- $f_1 \in [-50, 50]$: le décalage fréquentiel.
- $SNR \in [-10, 20]$: le SNR.

Les résultats de notre simulation sont présentés en figure 2.22. D'une part, nous pouvons voir que les performances de détection pour $T = 10s$ sont proches des 100 % pour un SNR supérieur à 5 dB. D'autre part, nous pouvons voir que les performances de détection pour $T = 1s$ sont toujours inférieures des 95 %, pour des valeurs de SNR supérieures à 20 dB. En effet, comme indiqué dans [99], [100], nous pouvons voir que les performances de détection dépendent fortement du temps d'observation T. L'implémentation de la simulation est décrite plus en détail dans l'article [97].

2.2.3.4 Estimation du micro-Doppler

La structure du signal décrit dans l'équation 2.65 peut être reformulée de la manière suivante :

$$\tilde{Y} = D\alpha + N \tag{2.74}$$

avec :

- $\tilde{Y} = [\tilde{y}(0)...\tilde{y}(L-1)]^T$: le vecteur contenant le signal (L×1).
- $D = [V_{-N}...V_0...V_N]$: le dictionnaire (L×(2N+1)) contenant les différentes exponentielles complexes V_n.
- $V_n = [1\ e^{2\pi(f_1+nf_m)T_s}...\ e^{2\pi(f_1+nf_m)(L-1)T_s}]^T$: un vecteur représentant la n-ème exponentielle complexe.
- T_e le temps d'échantillonnage.
- $\alpha = [a_{-N}...a_0...a_N]^T$: le vecteur contenant l'amplitude complexe de chaque exponentielle complexe.
- $N = [\tilde{n}(0)...\tilde{n}(L-1)]^T$: le vecteur contenant le bruit blanc gaussien complexe.

La méthode pour l'estimation des paramètres du signal que nous introduisons dans

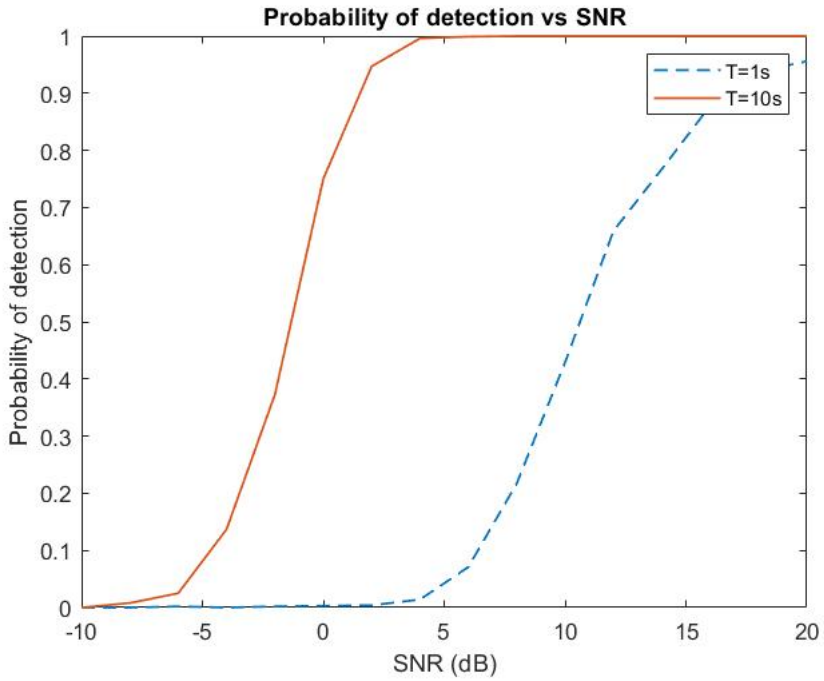


FIGURE 2.22 – Courbe de probabilité de bonne détection du micro-Doppler, avec un taux de fausse alarme de 10%

cette sous-section est appelée *Parameters Estimation Algorithm using Structure Dictionary* (PEA-SD). Cette méthode exploite les différentes connaissances *a priori* sur le signal : (1) la fréquence fondamentale f_1 a une amplitude prépondérante par rapport aux harmoniques ($\forall n \in \mathbb{Z}^*, a_0 >> a_n$), (2) le terme de micro-Doppler est périodique et (3) la structure du signal est un spectre de raie ($f_1 + nf_m$ avec $n \in \mathbb{Z}$). Notre approche est basée sur la construction d'un dictionnaire D, dépendant de l'estimation des paramètres f_m et f_1. Ce type d'approche a notamment été utilisé pour la détection de drones dans [101] ou dans des méthodes à haute résolution, comme MUSIC ou ESPRIT, qui utilisent un dictionnaire structuré à base d'exponentielles complexes [102].

Algorithm 1: Algorithme proposé

Data: $\tilde{Y} = [\tilde{y}(0)...\tilde{y}(L-1)]^T$ le vecteur contenant le signal
Result: La fréquence de vibration $\hat{f}_0$, la fréquence centrale $\hat{f}_1$ et les amplitude $\hat{\alpha}$
Estimer f_1;
Estimer f_m;
Construire le dictionnaire D;
Estimer α;

Pour chaque paramètre, il est possible de réaliser plusieurs estimations :

- Estimer f_1 : maximum de vraisemblance [103], méthode haute résolution (MUSIC, ESPRIT), ...
- Estimer f_m : fonction d'autocorrélation, cyclostationnarité, ...
- Estimer α :
  - Maximum de vraisemblance [102] : $\hat{\alpha} = (D^H D)^{-1} D^H X$.
  - Maximum a posteriori [5] [97] : $\hat{\alpha} = (D^H D + \gamma C)^{-1} D^H X$.

Il est aussi possible d'estimer la puissance du bruit (similaire à l'erreur de reconstruction) :

$$\hat{\sigma}^2 = \frac{1}{L} \left\| D\hat{\alpha} - X \right\|^2 \tag{2.75}$$

Comme pour la détection, une approche par simulation a été utilisée, avec les mêmes paramètres, pour évaluer les performances de notre méthode d'estimation. Nous avons choisi de comparer les trois algorithmes suivants :

- *Matching pursuit* : cet algorithme glouton exploite la parcimonie du signal en utilisant un dictionnaire d'exponentielles complexes espacées de 0.1 Hz [104].
- *Root-MUSIC* : cet algorithme de haute résolution est basé sur un modèle de signal composé d'une somme d'exponentielles complexes en présence d'un bruit blanc additif [103].
- PEA-SD : l'algorithme que nous venons de proposer exploite les différentes connaissances *a priori* du signal d'intérêt.

Pour notre algorithme, le décalage fréquentiel a été estimé de la manière suivante [103] :

$$\hat{f}_1 = arg\, max_f \left| \tilde{Y}(t) \right| \tag{2.76}$$

De plus, la fréquence de vibration f_m a été estimée ainsi [99] :

$$\hat{f}_0 = arg\, max_{\alpha \in [\alpha_{min}, \alpha_{max}]} I(\alpha) \tag{2.77}$$

Enfin, la métrique utilisée pour évaluer l'erreur de reconstruction est l'erreur quadratique moyenne de reconstruction ou *Reconstruction Mean Square Error* (RMSE) :

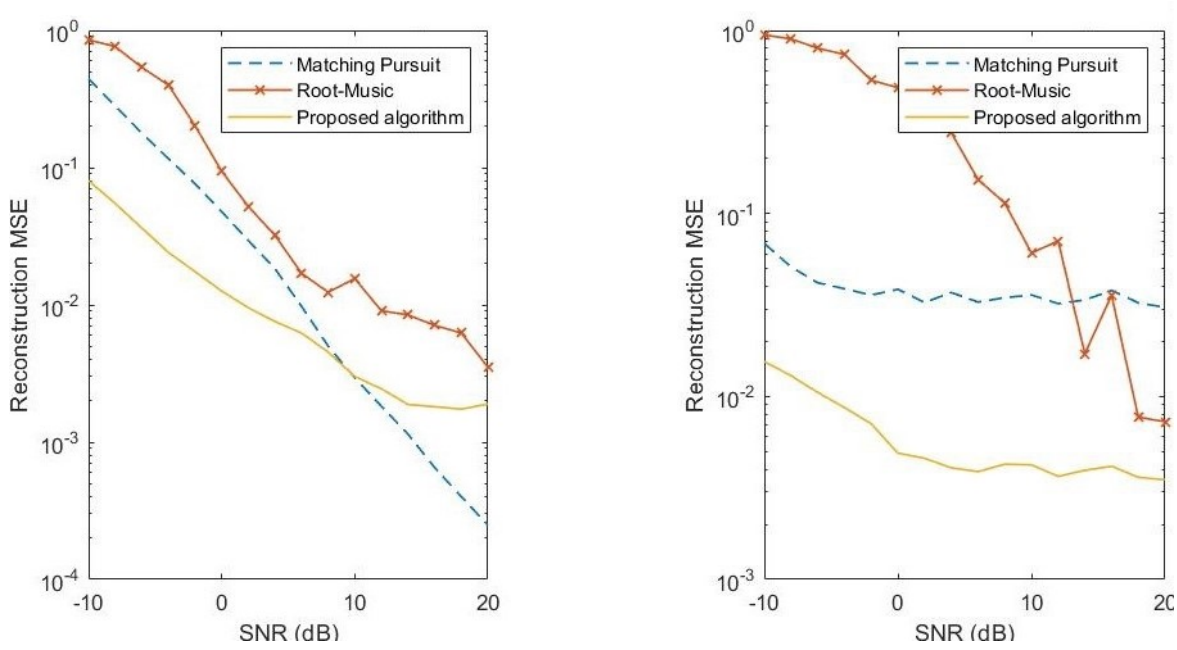
$$RMSE = \frac{\left\| \tilde{Y} - D\hat{\alpha} \right\|^2}{L} \tag{2.78}$$

5. Cette méthode d'estimation est obtenue en prenant en compte l'hypothèse que $\forall n \in \mathbf{Z}^*, a_0 >> a_n$, ce qui donne la fonction de coût : $\left\| D\alpha - X \right\|^2 + \gamma \sum_{n=-N}^{N} c_n a_n^2 = (D\alpha - X)^H (D\alpha - X) + \gamma \alpha^H C \alpha$

avec :

— L : la taille du signal.
— $\tilde{Y} = [\tilde{y}(0)...\tilde{y}(L-1)]^T$ le vecteur contenant le signal (L×1).
— D : le dictionnaire dépendant de $\hat{f}_0$ et $\hat{f}_1$.
— $\hat{\alpha}$: les amplitudes complexes estimées.

La figure 2.23 présente la comparaison des performances d'estimation en utilisant l'erreur quadratique moyenne de reconstruction comme métrique de performance. Nous pouvons voir que les performances de PEA-SD sont supérieures aux performances des deux autres algorithmes. Nous pouvons remarquer que, comme pour la détection, les performances d'estimation dépendent aussi fortement du temps d'observation T. L'implémentation de la simulation est décrite plus en détail dans l'article [97].



(a) $T = 1s$ (b) $T = 10s$

FIGURE 2.23 – Erreur de reconstruction moyenne pour la méthode d'estimation du micro-Doppler en fonction du SNR

2.2.4 Cas d'utilisation

Il existe plusieurs cas d'utilisation de la détection et l'estimation du micro-Doppler pour les radiocommunications :

— Détection d'objet vibrant : il serait possible de détecter la présence de vibrations au niveau de l'émetteur. Cela permettrait notamment de détecter des émetteurs vibrants comme des drones en vol [6].
— Authentification par empreinte radio : l'estimation des paramètres f_m, f_1 et $\{b_i\}$ pourrait être utilisée pour authentifier un émetteur radio. Ces paramètres pourraient être notamment utilisés en plus des descripteurs classiquement utilisés dans la littérature du RFF [1].
— Mesure de vibration : l'estimation de paramètres de vibrations f_m et $\{a_i\}$ permettrait de développer des systèmes d'analyse vibratoire au niveau du récepteur. En effet, le signal de l'émetteur servirait de support de mesure et permettrait ainsi de réduire les coûts en supprimant la nécessité d'avoir recours à un vibromètre.

2.3 Le récepteur radio : le cas de la démodulation en quadrature

À l'inverse de l'émetteur, le récepteur a pour but d'extraire le signal modulant à partir du signal modulé. Plus particulièrement, nous nous focaliserons sur l'étude d'une architecture particulière de récepteurs basée sur l'exploitation de la représentation en bande de base du signal : l'architecture homodyne.

Cette section a notamment pour but de réutiliser les travaux présentés en section 2.1 pour la modélisation et l'estimation des défauts du récepteur.

2.3.1 Hypothèses pour la modélisation des imperfections d'un récepteur

À l'instar de ce que nous avons fait en section 2.1, nous allons chercher à exprimer la représentation en bande de base du signal $\tilde{x}(t)$ [7] en fonction du signal à bande étroite $x(t)$:

$$\tilde{x}(t) = \mathcal{LP}(2x(t)e^{j2\pi f_0 t}) \tag{2.79}$$

avec $\mathcal{LP}(.)$: un filtre passe-bas idéal ayant une bande passante B égale à celle de la représentation en bande de base du signal.

6. Il s'agit d'ailleurs du cas d'utilisation qui a motivé ces travaux dans le cadre du projet AN DRO.
7. Nous avons choisi cette convention tout au long de ce manuscrit. Cependant, il en existe d'autres dans la littérature comme celles utilisées dans [76], [80].

Cette équation nous permet d'obtenir la structure du mélangeur I/Q (en réception) ou démodulateur en quadrature, présentée en figure 2.24, utilisée dans de nombreuses architectures de récepteur RF et permettant de récupérer la partie en phase $\tilde{x}_I(t)$ et en quadrature $\tilde{x}_Q(t)$ à partir de $x(t)$.

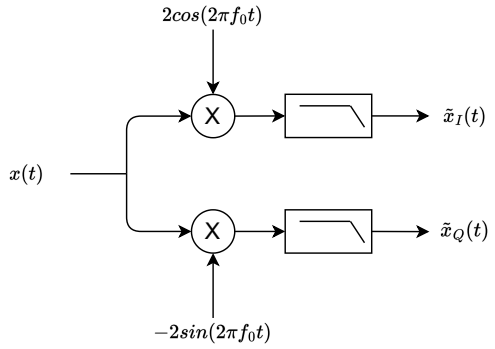


FIGURE 2.24 – Structure du mélangeur I/Q (en réception)

Dans la littérature [56], [57], [80], de nombreuses architectures de récepteurs ont été décrites parmi notamment : les récepteurs hétérodynes et les récepteurs homodynes. Dans [57], les auteurs mentionnent que l'architecture homodyne ou récepteur à fréquence intermédiaire nulle est attractive en comparaison des architectures hétérodynes ou récepteurs à fréquence intermédiaire. En effet, cette architecture est moins coûteuse que les architectures hétérodynes, car elle permet d'éliminer les filtres à réjection d'image et ne nécessite qu'un seul oscillateur pour la transposition de fréquence. De plus, dans [56], [80], les auteurs indique qu'il s'agit d'une architecture généraliste, prenant aussi en compte les architectures en fréquence presque intermédiaire, qui reposent sur une seconde étape de transposition numérique du signal.

Ainsi, dans cette section, nous allons traiter de la modélisation des imperfections de l'architecture homodyne présentée en figure 2.25. Comme nous pouvons le voir sur cette figure, le signal modulé $x(t)$ est reçu à l'aide d'une antenne, filtré avec un filtre passe-bande, amplifié par un amplificateur à faible bruit ou *Low Noise Amplifier* (LNA), puis décomposé en sa représentation en bande de base $\tilde{x}(t)$ à l'aide d'un mélangeur I/Q pour être échantillonné par deux convertisseurs analogique-numérique (CAN).

Les différents composants d'un récepteur homodyne sont les suivants :

— Antenne : ce composant est utilisé pour convertir une onde électromagnétique en un signal électrique de faible amplitude.

— Filtres : ces composants sont utilisés pour filtrer les fréquences non souhaitées. Il peut s'agir de filtres passe-bas ou passe-bande.
— LNA : ce composant permet d'amplifier les signaux à faible puissance sans trop dégrader le SNR.
— Mélangeur I/Q : ce composant permet d'obtenir la représentation en bande de base d'un signal de communication.
— Un oscillateur local (OL) : ce composant permet de générer le signal de porteuse $cos(2\pi f_0 t)$ nécessaire au mélangeur I/Q .
— Synthétiseur de fréquence : ce composant est utilisé pour faire varier la fréquence de l'OL.
— Convertisseurs Analogique-Numérique (CAN) : ces composants échantillonnent et quantifient un signal analogique permettant d'obtenir un signal numérique.

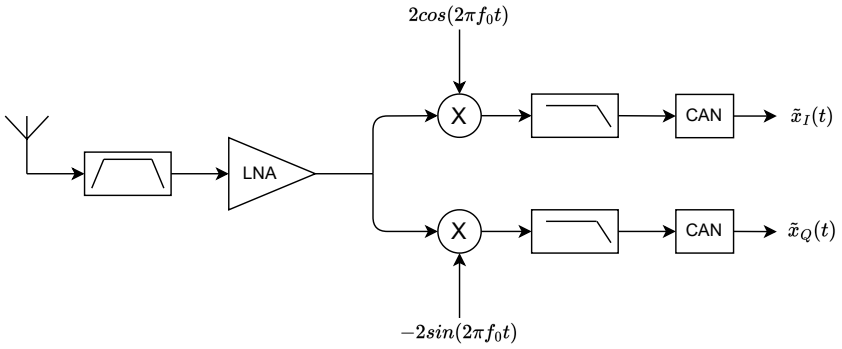


FIGURE 2.25 – Architecture de récepteurs homodyne

2.3.2 Modélisation des imperfections d'un récepteur radio

Dans cette sous-section, nous allons introduire deux modèles d'imperfections en bande de base, présentés en figure 2.29 et 2.30, permettant de prendre en compte les imperfections suivantes et qui seront décrites dans la suite de cette section :

— Décalage I/Q (CNA, mélangeur I/Q) : [57], [80].
— Déséquilibre en gain (CNA, mélangeur I/Q) : [77], [78], [80].
— Déséquilibre en phase (Mélangeur I/Q) : [61], [77], [80].
— Décalage fréquentiel (OL) : [1], [51].
— Bruit de phase (OL, couplage) : [51], [58], [79].
— Bruit interne (PA, composants passifs) : [78], [84].

— Bruit externe (Antennes) : [78], [84].

Le modèle d'imperfections proposé dans cette sous-section est composé de 3 parties :

— Modélisation conjointe des sources de bruit

— Modélisation des imperfections de l'oscillateur local

— Modélisation conjointe des imperfections des CAN et du mélangeur I/Q

Il serait possible de prendre en compte les problèmes de désadaptation d'impédance ou d'autres imperfections comme les défauts des filtres du mélangeur I/Q [80], mais notre modèle offre un compromis intéressant entre fidélité et simplicité du modèle.

2.3.2.1 Modélisation des sources de bruit

Il existe deux sources de bruit au niveau du récepteur : le bruit interne et le bruit externe [84]. D'une part, le bruit interne est une source de bruit qui a pour origine les composants du récepteur et il est généralement décomposé en deux sources de bruit distinctes : le bruit thermique provenant des composants passifs et le bruit de grenaille provenant des composants actifs. D'autre part, le bruit externe est une source de bruit qui a pour origine le milieu de transmission et il est généralement décomposé en deux sources distinctes : le bruit atmosphérique et le bruit cosmique. Même s'il est aussi possible de prendre en compte des potentielles interférences provenant d'autres émetteurs, nous ferons l'hypothèse qu'il n'y a pas d'interférences provenant d'autres émetteurs.

D'après Joindot et al. [84], les différentes sources de bruit des composants sont prépondérantes dans les systèmes de transmission et sont modélisées par une source de bruit unique située en amont du récepteur. Ce bruit $\tilde{n}(t)$ est un bruit blanc gaussien complexe : $\tilde{n}(t) \sim \mathcal{N}(0, \sigma_n^2)$ et $E(\tilde{n}(t)\tilde{n}(t-\tau)^*) = \sigma_n^2 \delta(\tau)$. Ainsi, dans notre approche de modélisation des sources de bruit, nous reprendrons cette proposition de modélisation.

2.3.2.2 Modélisation des défauts de l'oscillateur local

Nous avons déjà introduit les différents défauts de l'oscillateur local en sous-section 2.1.2 :

$$cos(2\pi f_0 t + \Phi_{OL}(t)) \tag{2.80}$$

Les défauts que nous avons choisis de modéliser via le terme $\Phi_{OL}(t) = 2\pi\Delta f t + \phi(t)$ sont : le décalage fréquentiel Δf ainsi que le bruit de phase $\phi(t)$.

La modélisation des défauts de l'OL en bande de base est décrite à la figure 2.26 et elle est généralement réalisée à l'aide d'une exponentielle complexe $e^{-j\phi_{OL}(t)}$ de la manière

suivante :

$$\tilde{y}(t) = \tilde{x}(t)e^{-j\phi_{OL}(t)} \tag{2.81}$$

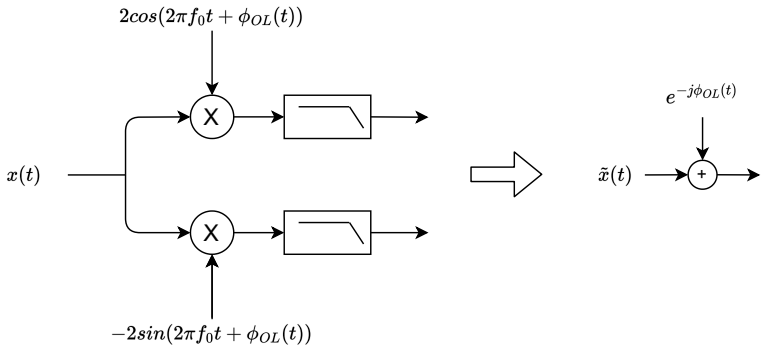


FIGURE 2.26 – Modélisation des défauts de l'oscillateur (en réception)

2.3.2.3 Modélisation conjointe des défauts des CAN et du mélangeur I/Q

Le mélangeur I/Q (en réception) et les CAN sont les derniers composants radiofréquences/analogiques d'un récepteur homodyne. Ils présentent les mêmes types de défauts que les CNA et les mélangeurs I/Q (en émission) présentés en section 2.1.2. Le modèle conjoint proposé dans cette sous-section permet la prise en compte de plusieurs défauts préalablement évoqués lors de la modélisation des défauts de l'émetteur. Tout d'abord, les erreurs de décalage et de gain présentes dans les CAN [78]. Il permet aussi de modéliser les problèmes d'isolation du mélangeur I/Q (décalage I/Q) ainsi que son déséquilibre I/Q (déséquilibre en phase et en gain) [80]. Les erreurs de décalage du CAN, ainsi que les défauts d'isolation [8] du mélangeur I/Q sont modélisés par les paramètres A_I et A_Q. Les erreurs de gain du CAN, ainsi que les défauts de déséquilibre en gain du mélangeur I/Q sont modélisés par le paramètre ϵ. Enfin, le défaut de déséquilibre en phase du mélangeur I/Q est modélisé par le paramètre θ. Ce modèle présenté en figure 2.27 est notamment inspiré des modèles présentés dans [16], [77].

La figure 2.28 présente la modélisation conjointe des défauts d'un mélangeur I/Q ainsi que de l'OL. Il est également possible d'utiliser le même raisonnement que précédemment, nous permettant aussi de modéliser les erreurs de décalage et de gain des CNA à l'aide des paramètres A_I, A_Q et ϵ.

8. Les termes I_I et I_Q sont les termes d'isolation reliés aux paramètres A_I et A_Q.

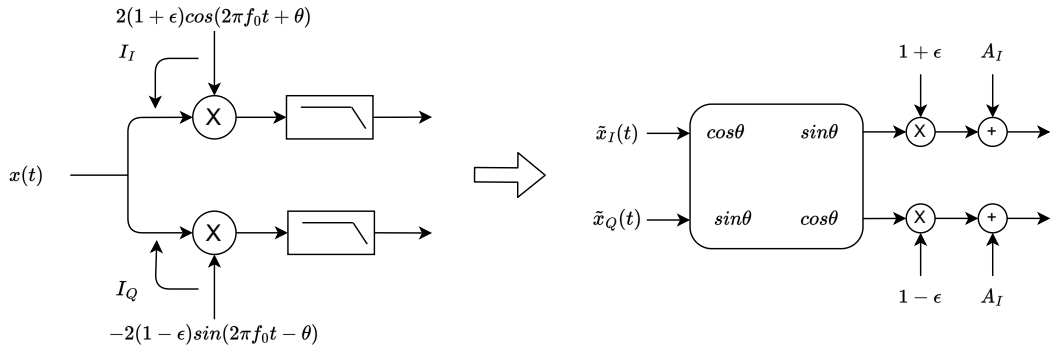


FIGURE 2.27 – Modélisation conjointe des défauts des CAN et d'un mélangeur I/Q

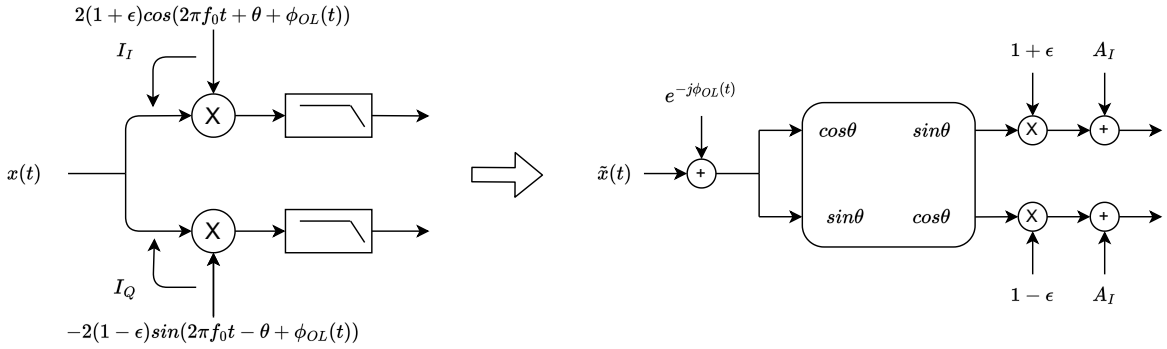


FIGURE 2.28 – Modélisation conjointe des défauts de l'oscillateur, du mélangeur I/Q et des CAN

2.3.2.4 Modèles des imperfections en bande de base

La figure 2.29 correspond au modèle intermédiaire prenant en compte les sources de bruit interne et externe comme une source de bruit unique $\tilde{n}(t)$, les défauts de l'oscillateur comme une exponentielle complexe et les défauts des CAN et du mélangeur I/Q à l'aide d'un modèle conjoint présenté en figure 2.27.

La figure 2.30 correspond au modèle d'imperfections du récepteur, qui intervertit la source de bruit avec l'exponentielle complexe modélisant les défauts de l'oscillateur. Cette commutation peut se justifier de la manière suivante :

— $\mathbb{E}(\tilde{n}(t)e^{-j\phi_{OL}(t)}) = \mathbb{E}(\tilde{n}(t))$: le moment d'ordre 1 est conservé.

— $\mathbb{E}(\tilde{n}(t)e^{-j\phi_{OL}(t)})\tilde{n}(t-\tau)e^{-j\phi_{OL}(t-\tau)}) = \sigma_n^2\delta(\tau)$: le moment d'ordre 2 est conservé.

— $|\tilde{n}(t)e^{-j\phi_{OL}(t)}| = |\tilde{n}(t)|$: la loi de l'enveloppe est conservée.

— $arg(\tilde{n}(t)e^{-j\phi_{OL}(t)}) \sim arg(\angle\tilde{n}(t))\, mod\, 2\pi$ [97] : la loi de la phase est conservée.

À l'aide de ces propriétés, il est possible de commuter les deux éléments car $\tilde{n}(t)e^{-j\phi_{OL}(t)}$ est toujours un bruit blanc gaussien complexe. Nous verrons dans la section 2.3.3 l'intérêt de cette astuce pour l'estimation des défauts du mélangeur I/Q et des CAN.

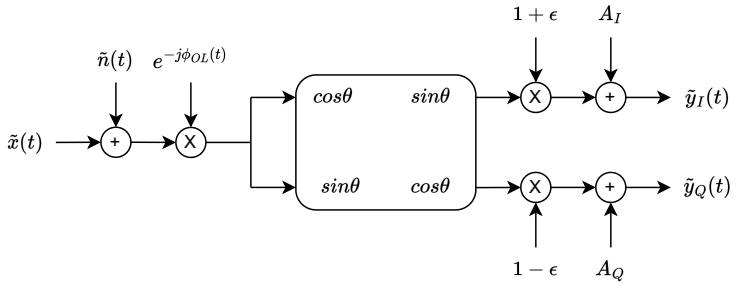


FIGURE 2.29 – Modèle intermédiaire des imperfections du récepteur en bande de base

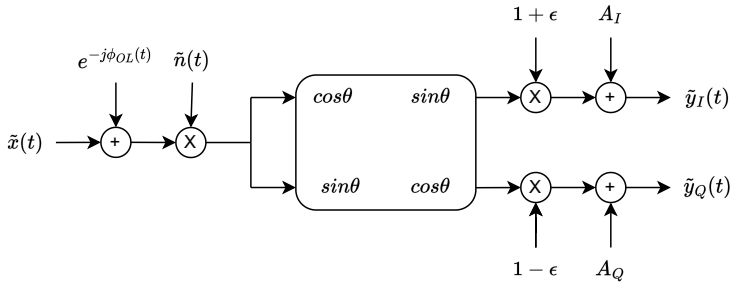


FIGURE 2.30 – Modèle des imperfections du récepteur en bande de base

2.3.3 Correction des imperfections d'un récepteur radio

Dans la sous-section 2.1.3, nous avions d'abord introduit les méthodes d'estimation des différentes imperfections du modèle d'imperfections avant de présenter les cas d'utilisation en sous-section 2.1.4 : une approche *model-based* pour le RFF. Dans cette section, nous allons d'abord introduire les techniques de correction des défauts pour ensuite présenter les techniques d'estimation, car les techniques d'estimation des défauts que nous présentons dépendent des techniques de correction que nous proposons.

Nous ne traiterons que le décalage fréquentiel de l'oscillateur local ainsi que les défauts du mélangeur I/Q et des CAN. Les deux autres défauts étant dépendants de variables aléatoires et donc difficilement compensables, car dépendantes du hasard.

La figure 2.31 présente la technique de correction des imperfections du récepteur. Le principe consiste à effectuer les opérations inverses pour compenser les défauts du récepteur. Nous commençons par compenser les défauts de décalage I/Q en utilisant les estimations $\hat{A}_I$ et $\hat{A}_Q$. Ensuite, nous compensons les défauts de déséquilibre en gain en supposant que $(1-\epsilon)(1+\epsilon) \approx 1$, i.e. ϵ^2 est négligeable. Une fois que nous avons compensé les défauts de déséquilibre en gain, nous compensons le défaut de déséquilibre en phase en multipliant par la matrice de correction dépendant de l'angle opposé $-\theta$. Une des hypothèses de cette correction suppose que $cos(2\theta) \approx 1$. En effet, le développement limité à l'ordre 1 de $cos(2\theta)$ est égal à 1. Enfin, nous multiplions par une exponentielle complexe $e^{j2\pi\Delta ft}$ pour compenser le décalage fréquentiel de l'oscillateur. Ce modèle de correction est notamment inspiré des travaux de Scott sur la correction des défauts d'un émetteur [77].

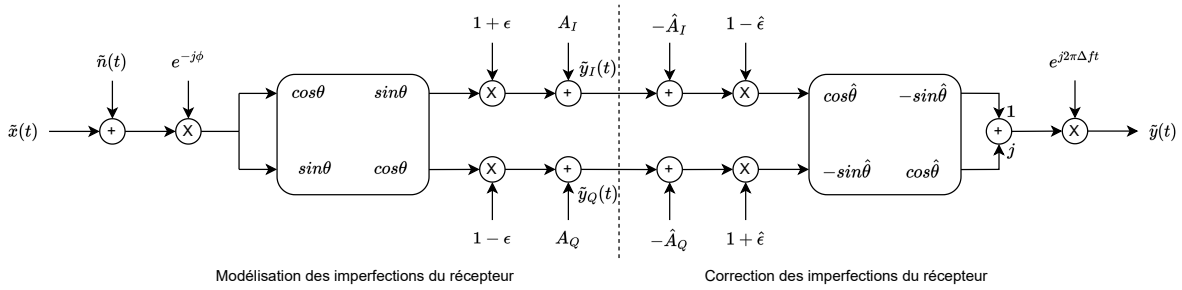


FIGURE 2.31 – Correction des défauts du récepteur

2.3.4 Estimation des imperfections d'un récepteur radio

L'algorithme d'estimation des imperfections du récepteur est basé sur la méthode de correction présentée en figure 2.31. Le principe général de notre méthode d'estimation est présenté en figure 2.32. Concernant l'estimation des paramètres du modèle d'imperfection, nous nous sommes inspirés de l'algorithme proposé en sous-section 2.1.3.1. De plus, l'estimation du décalage fréquentiel de l'oscillateur est basée sur la technique de récupération par élévation à la puissance M [84] évoquée en sous-section 2.1.3.3 dans le cas des signaux M-PSK.

Pour rappel, l'algorithme d'estimation des imperfections de l'émetteur propageait le signal d'entrée $\tilde{x}(t)$ dans le modèle d'imperfections et mesurait l'erreur avec le signal de sortie $\tilde{y}(t)$ pour mettre à jour l'estimation des imperfections. Dans notre cas, le signal reçu $\tilde{y}(t)$ est propagé dans le modèle de correction et l'erreur avec le signal d'entrée $\tilde{x}(t)$

(qui est supposé connu) est mesurée, pour mettre à jour l'estimation des imperfections. À noter, qu'il y a une phase de pré-traitements qui consiste à multiplier le signal connu par le terme de correction de phase. De plus, nous considérons une source de bruit virtuelle $\tilde{n}'(t) = -\tilde{n}(t)$, nous permettant de nous retrouver dans une configuration similaire à celle de la sous-section 2.1.3.

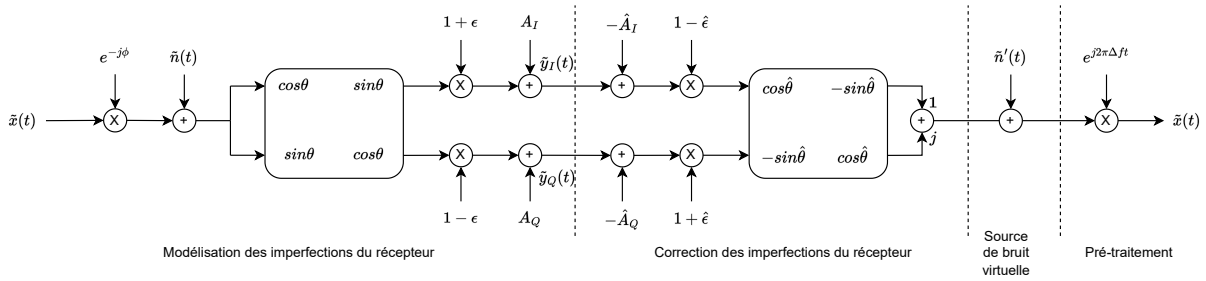


FIGURE 2.32 – Principe de l'estimation des défauts du récepteur

À l'instar de la sous-section 2.1.3, nous allons désormais considérer des signaux numériques. Ainsi, l'algorithme d'estimation utilisé pour estimer les paramètres du modèle d'imperfections (A_I, A_Q, ϵ et θ) est une descente de gradient stochastique et la fonction de coût associée est l'erreur moyenne quadratique :

$$C[n] = \mathbb{E}(|e[n]|^2) \tag{2.82}$$

avec $e[n] = \tilde{y}[n] - \tilde{x}[n]$, $\tilde{x}[n]$ le signal d'entrée et $\tilde{y}[n]$ la sortie du modèle d'imperfections.

La descente de gradient à l'étape $k+1$ est exprimée de la manière suivante :

$$\hat{A_I}^{(k+1)} = \hat{A_I}^{(k)} - 2\mu\mathbb{E}(\Re(e^*[n^{(k+1)}]\frac{\partial e[n^{(k+1)}]}{\partial \hat{A_I}^{(k)}})) \tag{2.83}$$

$$\hat{A_Q}^{(k+1)} = \hat{A_Q}^{(k)} - 2\mu\mathbb{E}(\Re(e^*[n^{(k+1)}]\frac{\partial e[n^{(k+1)}]}{\partial \hat{A_Q}^{(k)}})) \tag{2.84}$$

$$\hat{\epsilon}^{(k+1)} = \hat{\epsilon}^{(k)} - 2\mu\mathbb{E}(\Re(e^*[n^{(k+1)}]\frac{\partial e[n^{(k+1)}]}{\partial \hat{\epsilon}^{(k)}})) \tag{2.85}$$

$$\hat{\theta}^{(k+1)} = \hat{\theta}^{(k)} - 2\mu\mathbb{E}(\Re(e^*[n^{(k+1)}](j(1+\hat{\epsilon}^{(k)})\frac{\partial e[n^{(k+1)}]}{\partial \hat{\theta}^{(k)}})) \tag{2.86}$$

avec les gradients associés :

$$\frac{\partial e[n^{(k+1)}]}{\partial \hat{A_I}^{(k)}} = -(1 - \hat{\epsilon}^{(k)})e^{-j\hat{\theta}^{(k)}} \tag{2.87}$$

$$\frac{\partial e[n^{(k+1)}]}{\partial \hat{A_Q}^{(k)}} = -j(1 + \hat{\epsilon}^{(k)})e^{j\hat{\theta}^{(k)}} \tag{2.88}$$

$$\frac{\partial e[n^{(k+1)}]}{\partial \hat{\epsilon}^{(k)}} = -(\tilde{y}_I[n^{(k+1)}] - \hat{A_I}^{(k)})e^{-j\hat{\theta}^{(k)}} + j(\tilde{y}_Q[n^{(k+1)}] - \hat{A_Q}^{(k)})e^{j\hat{\theta}^{(k)}} \tag{2.89}$$

$$\frac{\partial e[n^{(k+1)}]}{\partial \hat{\theta}^{(k)}} = -j(1 - \hat{\epsilon}^{(k)})(\tilde{y}_I[n^{(k+1)}] - \hat{A_I}^{(k)})e^{-j\hat{\theta}^{(k)}} - (1 + \hat{\epsilon}^{(k)})(\tilde{y}_Q[n^{(k+1)}] - \hat{A_Q}^{(k)})e^{j\hat{\theta}^{(k)}} \tag{2.90}$$

Dans la figure 2.34, nous présentons les résultats de l'algorithme d'estimation des imperfections du récepteur pour le cas d'un signal QPSK avec un SNR de 30 dB. Nous pouvons voir que l'algorithme d'estimation converge bien vers les imperfections du modèle d'imperfections, même si l'estimation de certains paramètres semble légèrement décalée par rapport à la valeur souhaitée. Ce phénomène est probablement dû aux approximations $(1 + \epsilon)(1 - \epsilon) \approx 1$ et $cos(2\theta) \approx 1$. Cependant, comme nous pouvons l'observer dans la figure 2.33 qui présente l'erreur (quadratique) de reconstruction en fonction des étapes, nous pouvons voir que l'erreur tend (en moyenne) vers 10^{-3} correspondant à la puissance du bruit. Ainsi, nous pouvons observer sur la figure 2.35 que le signal corrigé converge bien vers le signal attendu.

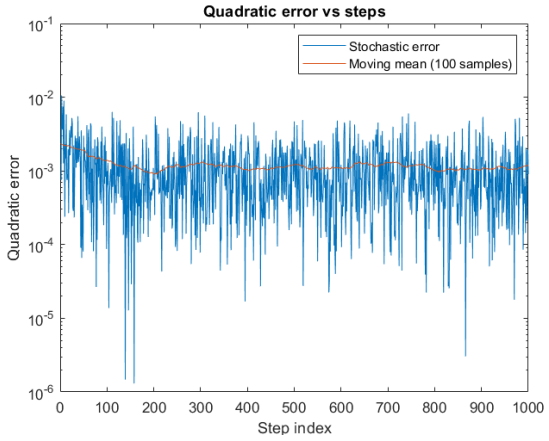


FIGURE 2.33 – Erreurs de l'estimation des défauts du récepteur

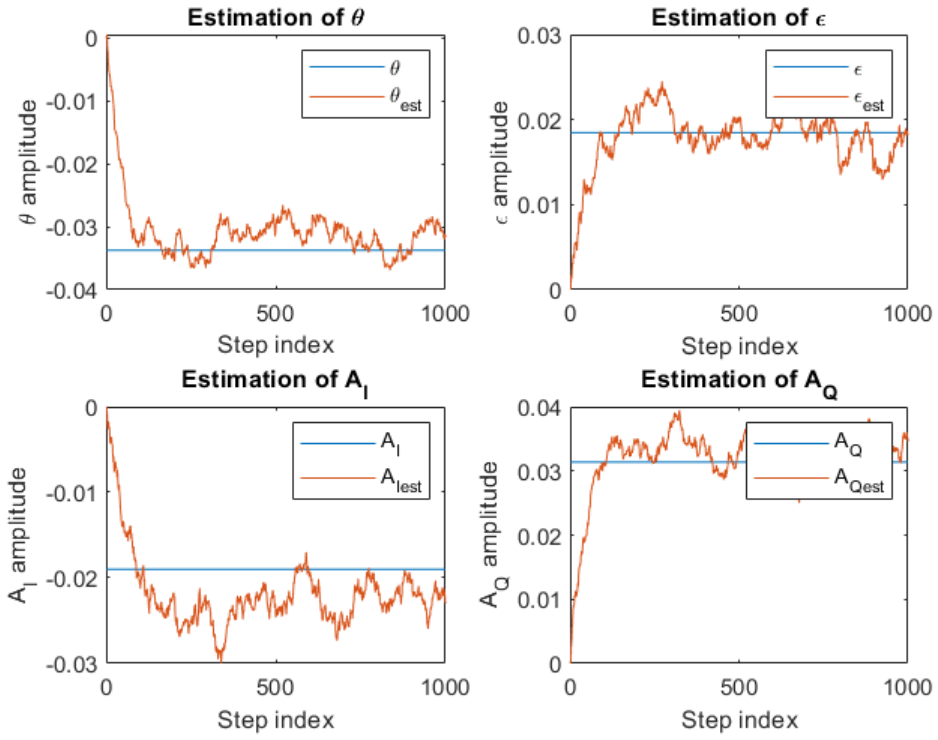


FIGURE 2.34 – Estimation des défauts du récepteur

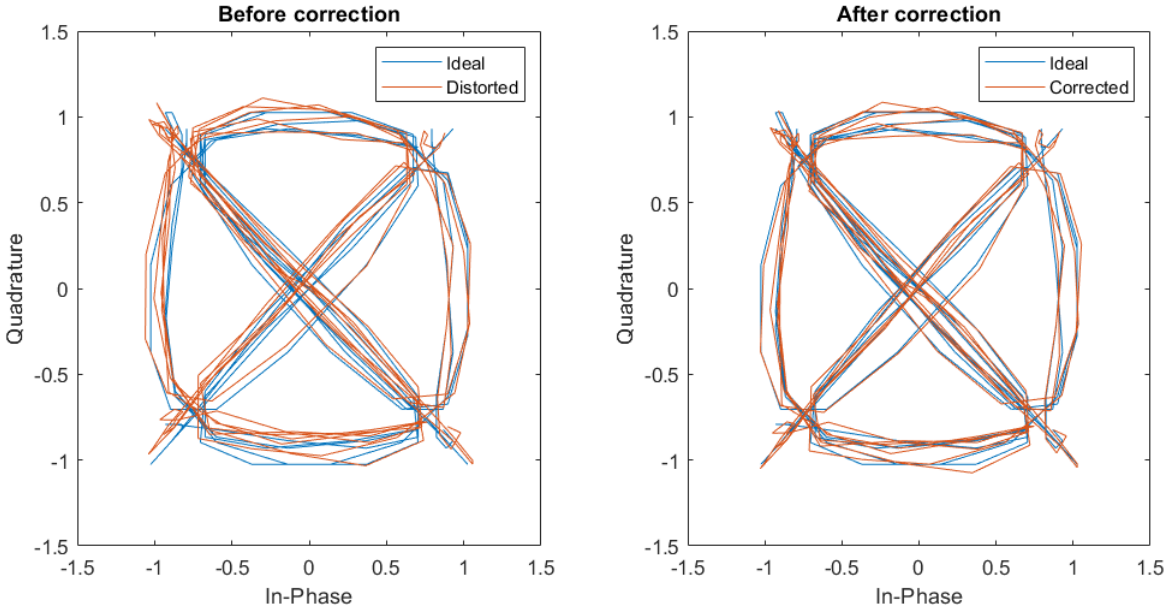


FIGURE 2.35 – Correction des défauts du récepteur

Chapitre 3

Méthodes d'authentification par empreinte radio pour l'IoT

Ce chapitre a pour but de présenter nos principales contributions au domaine de l'authentification par empreinte radio (RFF) pour l'IoT. Premièrement, nous présenterons trois propriétés d'utilisabilité que les méthodes de RFF doivent respecter pour être employées dans le contexte de la sécurité de l'IoT. Ensuite, nous présenterons en détail deux méthodes de RFF : les réseaux siamois pour le RFF et les *RF eigenfingerprints*. D'une part, pour la première méthode, nous avons évalué différentes manières d'entraîner des réseaux siamois, ainsi que différentes architectures pour le RFF. D'autre part, la seconde méthode est appelée *RF eigenfingerprints* et inspirée des *eigenfaces* en reconnaissance faciale [8], [9]. Enfin, nous comparerons les méthodes de RFF proposées vis-à-vis des propriétés d'utilisabilité proposées et nous finirons ce chapitre en décrivant différents cas d'applications du RFF dans un contexte IoT.

3.1 Propriétés d'utilisabilité du RFF pour l'IoT

Dans cette section, nous présentons les propriétés d'utilisabilité du RFF pour l'IoT que nous avons sélectionnées et qui ont guidées le développement des méthodes que nous avons proposées dans ce chapitre. Plus particulièrement, comme nous l'avons évoqué dans le contexte de ce manuscrit, l'IoT présente plusieurs caractéristiques [5] : son échelle, son hétérogénéité, son interconnectivité et sa dynamicité. De plus, certains défis de l'IoT ont une influence sur les méthodes de RFF comme la scalabilité [70], ainsi que la consommation énergétique [5]. Pour ces raisons, nous allons nous baser sur ces caractéristiques et ces défis pour définir les propriétés d'utilisabilité suivantes :

— L'adaptabilité : propriété permettant de s'adapter à l'hétérogénéité des standards IoT, notamment aux différentes formes d'onde.
— La scalabilité : propriété permettant de s'adapter à l'échelle de l'IoT, ainsi qu'à sa

dynamicité. Plus particulièrement, nous allons décomposer la scalabilité en deux sous-propriétés : l'apprentissage frugal et la réentraînabilité partielle.

— La complexité : propriété liée à la complexité algorithmique d'une méthode d'authentification par empreinte radio. Plus particulièrement, nous allons décomposer la complexité en deux sous-propriétés : la complexité mémoire et la complexité de calcul.

Ces trois propriétés d'utilisabilité du RFF pour l'IoT vont être décrites plus précisément dans la suite du manuscrit, ainsi que les sous-propriétés qui en découlent.

3.1.1 Adaptabilité

Dans ce manuscrit, l'adaptabilité est une propriété d'utilisabilité en RFF pour l'IoT permettant de s'adapter à l'hétérogénéité des protocoles sans-fils de l'IoT. En effet, dans sa présentation à la DARPA IA Colloquium [1] en 2019, E. Mattei précise que les techniques d'authentification par empreinte radio sont majoritairement basées sur l'utilisation de descripteurs définis à la main (*hand-crafted features*), [1], [105]). Or, son constat est qu'à l'heure de l'IoT, le moindre changement (dans un protocole, n.d.l.r) nécessiterait de revoir le choix des descripteurs utilisés. Ainsi, pour répondre à cette problématique, la majorité des auteurs de la littérature du RFF utilisent maintenant des approches d'apprentissage de représentations (*feature-learning* en anglais [52], [106], [107]), voire de l'apprentissage profond, pour apprendre les descripteurs directement à partir des signaux [28]. De plus, dans [65], E. Mattei explique que l'utilisation de l'apprentissage de représentations permet d'être agnostique à la forme d'onde, ainsi qu'aux spécificités du protocole. Cette problématique se retrouve aussi dans d'autres domaines, notamment dans des domaines connexes comme la reconnaissance de modulations. Par exemple, Ghasemi et al. [108] mentionnent que les descripteurs définis à la main sont dépendants d'hypothèses *a priori*, mais que l'utilisation d'algorithmes d'apprentissage profond pouvait permettre de remédier à ce problème.

Pour répondre à cette problématique d'adaptabilité en authentification par empreinte radio pour l'IoT, nous utiliserons des méthodes provenant d'un sous-domaine de l'apprentissage automatique : l'apprentissage de représentations [52], [106]. Ce sous-domaine consiste à apprendre une représentation directement à partir des données «brutes» (signaux, images, ...), au lieu de dépendre de descripteurs définis à la main. Plus particulière-

1. RF Machine Learning Systems - Learning to secure the Internet of Things (youtube.com)

ment, les méthodes d'apprentissage profond basées sur des réseaux de neurones sont aussi des méthodes d'apprentissage de représentations avec plusieurs niveaux de représentation [52]. La figure 3.1, inspirée du diagramme de Veen introduit par Goodfellow et al. dans [107], résume le lien entre ces différents sous-domaines. Par la suite, nous comparerons les résultats des méthodes de RFF proposées à ceux présentés dans [28], car il s'agit d'un des articles de référence sur l'utilisation de l'apprentissage profond pour une tâche de RFF et donc de l'apprentissage de représentations.

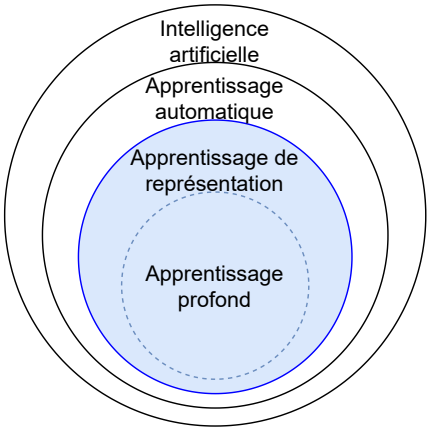


FIGURE 3.1 – Diagramme de Veen de l'apprentissage de représentations

Comme cela a été mentionné en section 1.4, la notion d'adaptabilité (*adaptability*) pour le RFF a notamment été mentionnée par Hazra et al. dans [69]. En revanche, la définition des auteurs est légèrement différente de la nôtre puisqu'ils définissent l'adaptabilité comme la propriété de s'adapter à différents réseaux IoT (ensemble d'appareils) et pas forcément à différents protocoles.

3.1.2 Scalabilité

Dans ce manuscrit, la scalabilité (horizontale, opposée à scalabilité verticale) consiste, pour un algorithme d'authentification par empreinte radio, à s'adapter facilement à l'ajout ou au retrait d'un émetteur légitime. Comme nous l'avons évoqué précédemment, beaucoup d'approches d'apprentissage de représentations utilisées en RFF sont basées sur des réseaux de neurones profonds. Or, ces réseaux apprennent des paramètres, que l'on appellera aussi poids ou coefficients, dépendant d'un nombre de classes fixe. Il est par conséquent difficile de rajouter ou même de supprimer une classe sans devoir réentraîner l'intégralité du réseau [71]. Nous pouvons ainsi considérer que le besoin de scalabilité en

RFF est dû à la dynamicité, ainsi qu'à l'échelle de l'IoT. En effet, comme Patel et al. l'ont mentionné dans [5], un réseau IoT peut être composé de nombreux appareils pouvant le quitter ou le rejoindre. Il est ainsi nécessaire de concevoir des algorithmes pouvant rajouter/supprimer des signatures d'émetteurs à la volée pour remédier à la dynamicité du réseau tout en étant capable de gérer un grand nombre d'émetteurs.

Nous avons choisi de décomposer la scalabilité en deux sous-propriétés :

— L'apprentissage frugal ou *Few-Shot Learning* (FSL) : consiste à apprendre à résoudre un problème d'apprentissage automatique avec un nombre limité d'exemples [109]. En apprentissage automatique, il est courant de considérer qu'il est nécessaire d'avoir un nombre d'exemples L supérieur à 50 pour pouvoir envisager d'entraîner un modèle [110]. De plus, certains auteurs considèrent qu'il est nécessaire d'avoir un nombre d'exemples respectant l'équation [2] : $L > 10NC$ [112]. Ainsi, les algorithmes de FSL sont capables de passer en deçà de cette limite, notamment les algorithmes de *one-shot learning* ne nécessitant qu'un seul exemple par classe. Pour cela, ces algorithmes peuvent se baser sur des connaissances *a priori*. Dans notre cas, le FSL nous permet d'avoir peu d'exemples pour l'ajout d'un nouvel émetteur légitime.
— La Réentrainabilié Partielle (RP) : permet de réentraîner un algorithme d'apprentissage automatique pour l'ajout/retrait d'une nouvelle classe sans réutiliser les données des classes déjà apprises par l'algorithme. De plus, cette sous-propriété permet aussi de supprimer une classe sans nécessité de réentraîner le réseau à partir des données des autres classes. Dans notre cas, cette propriété permet de ne pas stocker les données d'entraînement pour ajouter ou supprimer un émetteur légitime.

Comme cela a été mentionné en section 1.4, la notion de scalabilité (*scalability*) pour le RFF a notamment été mentionnée par Shen et al. [66] mais aussi par Arroyo et al. [71].

Bien que nous ayons mentionné dans la section 1.4 la notion de capacité, nous ne traiterons pas de cet aspect de la scalabilité dans cette thèse, car nous n'avons pas eu accès à des bases de signaux réels assez conséquentes pour pouvoir nous y intéresser.

2. Avec N la dimension des données et C le nombre de classes. Dans le cas de la classification d'images, il est possible d'avoir un nombre d'exemples moins contraignant en respectant $L > 1\,000C$, car une image est généralement composée d'un nombre de pixels N largement supérieur au millier [111].

3.1.3 Complexité

La complexité consiste à évaluer l'ordre de grandeurs des ressources nécessaires à l'exécution d'un algorithme. Cette notion est largement utilisée en algorithmie pour caractériser les performances d'exécution (temporelles et spatiales) d'un algorithme. Comme nous l'avons évoqué dans la section 3.1.1, les approches d'apprentissage de représentations utilisées en RFF sont basées sur des réseaux de neurones, notamment des réseaux de neurones convolutifs ou *Convolutive Neural Networks* (CNN). Or, ces architectures ont des milliers de paramètres, voire des millions, ce qui complique le stockage et l'exécution de ces modèles sur des cibles restreintes [71] que l'on peut retrouver en IoT. De plus, la complexité est liée à la consommation énergétique d'un appareil, car le stockage et l'exécution de ces modèles peut avoir un fort impact sur l'autonomie des appareils IoT [72].

Nous avons choisi de décomposer la complexité en deux sous-propriétés :

— La complexité mémoire : définissant l'espace mémoire nécessaire pour le stockage d'un modèle. Dans ce chapitre, nous considérerons le nombre de paramètres [3] pour l'évaluation de la complexité mémoire d'un modèle.

— La complexité de calcul : définissant le nombre d'opérations nécessaires à l'exécution d'un modèle. Dans ce chapitre, nous considérerons le nombre de multiplications [4] pour l'évaluation de la complexité de calcul d'un modèle.

Pour évaluer la complexité des méthodes de RFF proposées, nous allons nous baser sur des travaux en *TinyML* [115], [116], i.e. le domaine traitant de l'implémentation des algorithmes d'apprentissage automatique sur des cibles embarquées. Plus particulièrement, nous utiliserons des métriques inspirées de celles proposées [5] dans [113], un des articles de référence du *MLPerf Tiny Benchmark* [116]. Dans cet article, Zhang et al. traitent de la complexité des réseaux de neurones pour le *TinyML* pour une tâche de reconnaissance de mots-clés (*keyword spotting*). Pour cela, ils évaluent un modèle en fonction de sa taille mémoire, ainsi que de son nombre d'opérations par seconde.

Dans ce manuscrit, nous considérerons plutôt que la complexité de calculs est évaluée en nombre de multiplications et que la complexité mémoire est évaluée en nombre

3. Généralement, la complexité mémoire est évaluée en nombre d'octets, mais considérer le nombre de coefficients revient à supposer une quantification sur 8 bits pour chaque poids [113], [114]. En effet, cette quantification est la plus petite quantification possible en implémentation logicielle.

4. Généralement, le nombre d'opérations correspond aux nombres de multiplications-additions. Dans notre cas, nous ne prendrons en compte que le nombre de multiplications, qui est similaire au précédent mais également plus simple à évaluer.

5. Il est important de noter que ces valeurs ont été proposées pour réaliser 10 inférences par seconde sur des cibles embarquées avec 1 Mo (ou moins) de mémoire flash.

de paramètres. En effet, ces deux métriques sont toutes deux indépendantes de l'implémentation du modèle, à l'inverse des métriques utilisées dans [113]. De plus, nous allons considérer que les métriques proposées et leurs valeurs associées permettent également d'évaluer des modèles d'apprentissage de représentations et pas uniquement des réseaux de neurones. Enfin, les métriques et valeurs associées que nous utiliserons sont présentées dans le tableau 3.1. D'ailleurs, les valeurs sont, elles aussi, inspirées de [62].

D'autres valeurs et métriques ont été proposées dans la littérature du *TinyML*, notamment dans [114], où les auteurs proposent que la taille d'un modèle et de son pic d'utilisation mémoire doivent être inférieurs à 250 000 octets et que le nombre d'opérations doit être inférieur à 60 000 000 d'opérations. Cet article donne une limite haute inférieure à celle donnée dans [113] et utilise une métrique supplémentaire (le pic d'utilisation mémoire), mais avec une granularité moins importante que dans [113]. Cependant, pour pouvoir exploiter cette granularité et donc mieux comparer les méthodes de RFF proposées, nous avons préféré nous inspirer des travaux présentés dans [113] plutôt que ceux présentés dans [114].

Type de modèle	Mémoire (param.)	Calculs (mult.)
Petit	80×10^3	6×10^6
Moyen	200×10^3	20×10^6
Grand	500×10^3	80×10^6

TABLE 3.1 – Valeurs utilisées pour l'évaluation de la complexité des méthodes de RFF proposées

Comme cela a été mentionné en section 1.4, la notion de complexité (*complexity*) mémoire/calcul pour le RFF a été mentionnée par Arroyo et al. dans [71].

3.2 Méthode 1 : réseaux siamois pour le RFF

Cette contribution consiste à évaluer différentes manières d'entraîner des réseaux siamois, ainsi que différentes architectures pour le RFF. Plus particulièrement, nous nous limiterons à des réseaux de neurones traitant directement des signaux I/Q. Cette méthode se focalise majoritairement sur les propriétés d'adaptabilité et de scalabilité, même si nous avons aussi exploré la complexité des réseaux siamois pour le RFF.

Dans cette section, nous présentons d'abord, en sous-section 3.2.1, la méthodologie utilisée. Ensuite, les résultats de nos expérimentations avec des réseaux siamois pour le

RFF sont présentés en sous-section 3.2.2. Enfin, nous comparons les résultats que nous avons obtenus avec ceux de la littérature dans la sous-section 3.2.3.

Pour rappel, les réseaux siamois ont été proposés pour la première fois par Bromley et al. dans [6] pour une tâche de reconnaissance de signatures manuscrites. Cependant, ce type de réseaux de neurones est largement utilisé, notamment en reconnaissance faciale [7], [117], [118]. Le principe de fonctionnement d'un réseau siamois, présenté en figure 3.2, consiste à apprendre une mesure de similarité entre deux entrées X_1 et X_2 à l'aide de deux sous-réseaux de neurones, noté $G_W(.)$, partageant des poids identiques W. Plus particulièrement, cette mesure de similarité est calculée à partir des représentations latentes $G_W(X_1)$ et $G_W(X_2)$, obtenues respectivement à partir des entrées X_1 et X_2. L'espace de projection des entrées X, noté $G_W(X)$, est aussi appelé espace d'enchâssement dans la littérature [119]. Il est important de noter qu'il existe plusieurs manières d'entraîner un réseau siamois que nous appellerons paradigmes d'apprentissage dans la suite de ce manuscrit.

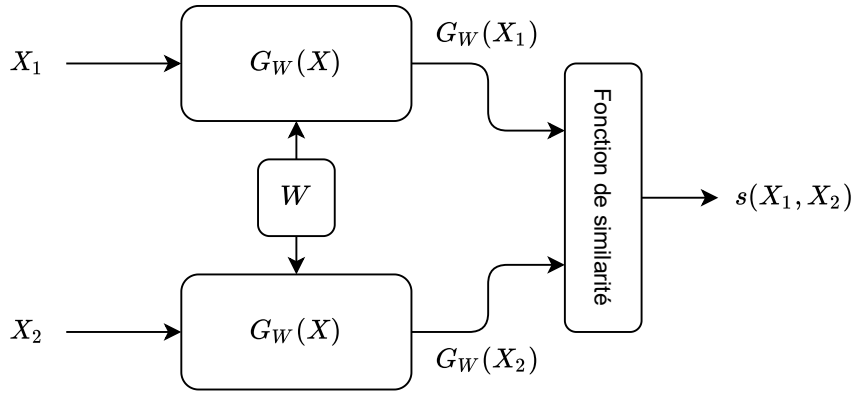


FIGURE 3.2 – Principe de fonctionnement des réseaux siamois

3.2.1 Méthodologie

Cette sous-section est constituée de trois parties distinctes : les architectures utilisées, les paradigmes d'apprentissage étudiés et la prise de décision.

L'entraînement des modèles, ainsi que leur évaluation, ont été réalisés avec Tensorflow 1.x à l'aide de l'interface de haut-niveau Keras [120]. Nous n'avons pas utilisé Tensorflow 2.x, car à l'époque de nos expérimentations (juillet-septembre 2020), cette version de Tensorflow n'était pas encore assez stable.

3.2.1.1 Architectures des sous-réseaux siamois utilisées

Les architectures des sous-réseaux $G_W(.)$ que nous avons considérées sont inspirées des architectures utilisées en reconnaissance de modulations [121], ainsi qu'en authentification par empreinte radio [28], [29]. Plus particulièrement, nous nous sommes inspirés de l'architecture introduite par Sankhe et al. dans [29] et utilisant des couches convolutives (2D) conçues initialement pour le traitement des images. Ce choix est justifié par le fait que cette architecture a obtenu de très bons résultats dans la littérature du RFF. De plus, le jeu de données que nous avons utilisé et que nous présenterons plus loin dans le document, provient justement du même article.

Les réseaux de neurones que nous avons considérés traitent des signaux I/Q de 128 échantillons comme dans [29]. Plus particulièrement, pour pouvoir prendre en compte ce type de signaux à l'aide de couches convolutives (2D), l'entrée du réseau sera une image de dimension $2 \times 128 \times 1$. La première dimension correspond à la séparation de la partie réelle (I) et la partie imaginaire (Q). La seconde dimension correspond aux nombres d'échantillons et la dernière dimension au nombre de canaux. En effet, les CNNs 2D sont classiquement utilisés pour le traitement d'images qui peuvent avoir plusieurs canaux, par exemple les canaux RGB (rouge, vert et bleu). Dans notre cas, nous ne considérerons qu'un seul canal. La figure 3.3 représente la structure de la couche d'entrée des réseaux siamois que nous avons considérés dans ce manuscrit.

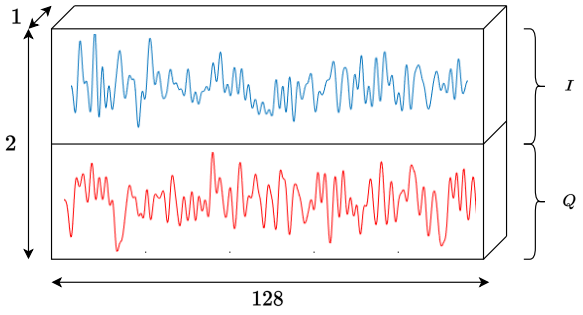


FIGURE 3.3 – Couche d'entrée des réseaux siamois pour le RFF

Nous avons évalué deux architectures différentes pour l'utilisation de réseaux siamois en RFF : l'architecture de référence et l'architecture légère. Les poids des sous-réseaux dépendent bien évidemment des données d'entraînement et aussi d'hyperparamètres qui ne sont pas apprenables, mais qui ont une grande influence sur les performances finales. Ainsi, l'apprentissage des paramètres des deux architectures utilise l'algorithme d'optimisation Adam [31], comme dans [29], dépendant d'un hyperparmètre μ : le pas d'apprentissage. De

plus, plusieurs techniques de régularisation sont utilisées pour éviter le sur-apprentissage, i.e un phénomène réduisant les propriétés de généralisation à de nouvelles données d'un algorithme d'apprentissage automatique [31]. D'une part, l'amplitude des paramètres du réseau sont contraints à l'aide d'une technique de régularisation L_2 dépendant d'un hyperparamètre l, qui contrôle la quantité de régularisation. D'autre part, nous avons utilisé une technique de régularisation appelée *dropout* ($p_{dropout} = 0.5$) et consistant, pendant l'apprentissage, à désactiver les neurones d'une couche avec une probabilité $p_{dropout}$. Cela a pour effet d'éviter que certains neurones ne se spécialisent. Enfin, la dernière couche de neurones dense dépend d'un hyperparamètre D qui correspond au nombre de neurones de cette dernière.

Pour résumer, les hyperparamètres que nous cherchons à déterminer sont les suivants :

— D : le nombre de neurones de la dernière couche dense correspondant à la dimension de l'espace d'enchâssement.

— μ : le pas d'apprentissage de l'optimiseur Adam.

— l : le terme permettant de contrôler la régularisation L_2.

Il existe de nombreux hyperparamètres comme les autres hyperparamètres de l'optimiseur Adam, mais nous avons pris des valeurs classiques ou par défaut. En effet, la recherche d'hyperparamètres est une tâche fastidieuse, même en utilisant un serveur de calcul. Ainsi, nous avons choisi d'étudier l'impact de ces trois hyperparamètres (D, μ et l) qui nous paraissaient les plus intéressants.

Pour les différentes expérimentations de cette méthode, les valeurs des différents hyperparamètres ont été trouvés à l'aide d'une recherche par quadrillage [31] et d'un jeu de validation [31], [86]. Pour chaque hyperparamètre, plusieurs valeurs ont été proposées et la recherche par quadrillage réalise le produit cartésien des différentes possibilités de chaque hyperparamètre. Ainsi, pour chacune de ces combinaisons d'hyperparamètres, un modèle sera appris sur un jeu d'entraînement et évalué sur un jeu de validation. Enfin, le réseau de neurones avec les meilleures performances sur le jeu de validation sera entraîné sur la concaténation du jeu d'entraînement et du jeu de validation [86] et évalué sur le jeu de données, dit de test, pour valider les hyperparamètres choisis. Il s'agit d'une approche de sélection d'hyperparamètres largement utilisée en apprentissage automatique [86].

Les valeurs associées à chacun des hyperparamètres considérés dans ces travaux sont les suivantes :

— $D = \{32, 80, 128\}$: le nombre de neurones de la dernière couche.

— $l = \{0.00001, 0.0001\}$: la valeur de la régularisation L_2.

— $\mu = \{0.0001, 0.001\}$: le pas d'apprentissage de l'optimisation Adam.

Ainsi, nous avons 12 tuples d'hyperparamètres possibles (3x2x2).

Architecture de référence L'architecture, décrite en figure 3.4, est constituée de deux couches convolutives de 50 filtres chacune, d'une première couche dense de 256 neurones et d'une dernière couche de D neurones. Ce paramètre D étant aussi un hyperparamètre du modèle.

Nous pouvons remarquer que la première couche convolutive (Conv) traite indépendamment le signal en phase (I) et en quadrature (Q) à l'aide du même ensemble de filtres 1D : 50 filtres de taille (1×7). Ensuite, la deuxième couche convolutive agrège la partie en phase et en quadrature filtrée à l'aide d'un ensemble de filtres 2D : 50 filtres de taille (2×7). La première couche dense (Dense) permet de créer des descripteurs de haut-niveau d'abstraction [29] et la dernière de D neurones correspond à l'espace d'enchâssement. De plus, toutes les couches utilisent des fonctions d'activation (non-linéaire) *Rectified Linear Unit* (ReLu) : $ReLu(x) = max(0, x)$.

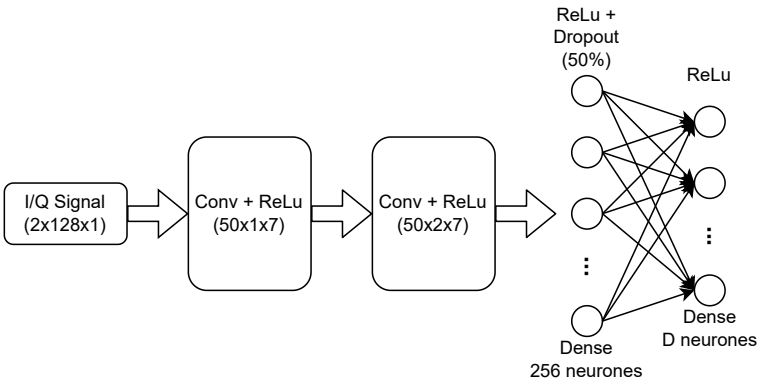


FIGURE 3.4 – Architecture de référence des sous-réseaux siamois

Architecture légère L'architecture, décrite en figure 3.5, est constituée d'une première couche convolutive de 50 filtres suivie d'une couche de *pooling* Max-Pool (regroupement [107]) avec un pas (1, 2), d'une seconde couche convolutive de 50 filtres suivie d'une couche de *pooling* Max-Pool avec un pas (1, 2) et d'une couche dense de 256 neurones. Cette architecture, inspirée de [28] et [29], permet à la fois de réduire le nombre de paramètres de l'architecture de référence, ainsi que le nombre d'opérations nécessaires. En effet, une couche de *pooling* a pour effet de réduire la dimension de sortie (ou d'entrée) d'une couche convolutive, aussi appelée cartes de caractéristiques (*feature maps*, [31]),

et donc de réduire le nombre total de paramètres de l'architecture. Nous pouvons ainsi voir ces couches de *pooling* comme des couches d'agrégation ou de sous-échantillonnage permettant de créer des représentations localement invariantes par translation [107]. Nous verrons également, dans la suite du manuscrit, que l'utilisation de couches de *pooling* a pour effet de diminuer le nombre total d'opérations du réseau de neurones lors d'une inférence. De plus, la suppression de la dernière couche de D neurones réduit le nombre de paramètres, mais aussi le nombre de multiplications nécessaires.

Tout comme l'architecture précédente, nous pouvons remarquer que la première couche convolutive traite indépendamment le signal en phase (I) et en quadrature (Q) à l'aide du même ensemble de filtres 1D : 50 filtres de taille (1×7). Ensuite, la deuxième couche convolutive agrège la partie en phase et en quadrature filtrées à l'aide d'un ensemble de filtres 2D : 50 filtres de taille (2×7). Enfin, la couche dense correspond à l'espace d'enchâssement. Cependant, la grande différence avec l'architecture de référence consiste en l'utilisation de couches de *pooling* de type Max-Pool (ou *max-pooling* [31]), ainsi que la suppression de la deuxième couche dense de D neurones. De plus, le *dropout* est utilisé pour les couches convolutives et non plus pour la couche dense. En effet, des tests préliminaires, que nous avions réalisés, avaient montré l'impact négatif de l'utilisation du *dropout* sur la couche correspondant à l'espace d'enchâssement. Ainsi, nous avons choisi de conserver cette technique de régularisation en l'appliquant aux couches convolutives comme dans [28].

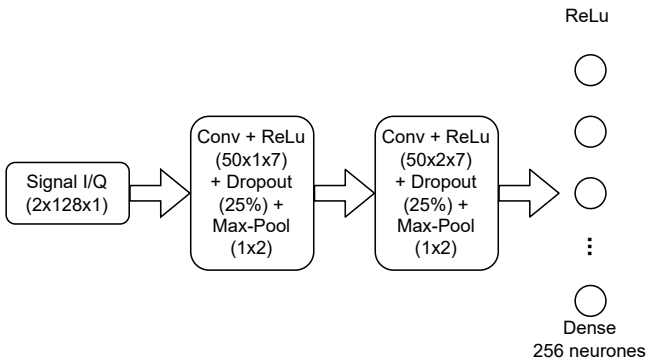


FIGURE 3.5 – Architecture légère des sous-réseaux siamois

3.2.1.2 Paradigmes d'apprentissage des réseaux siamois

Trois paradigmes d'apprentissage ont été étudiés :

— Fonction de coût contrastive

— Régression logistique
— Apprentissage par transfert contrastif (ou transfert contrastif)

La fonction de coût contrastive Le principe des fonctions de coût contrastives a été introduit par S. Chopra, R. Hadsell et Y. Le Cun dans [7]. Dans cet article, les auteurs présentent un cas d'application des réseaux siamois pour la reconnaissance faciale. Son principe consiste à contraindre les représentations latentes $G_W(X_1)$ et $G_W(X_2)$ à respecter certaines propriétés. Plus particulièrement, les auteurs proposent la fonction de coût suivante :

$$L(W) = yL_G(E_W(X_1, X_2)) + (1 - y)L_I(E_W(X_1, X_2)) \tag{3.1}$$

avec

— $E_W(X_1, X_2) = \|G_W(X_1) - G_W(X_2)\|_2$: la fonction d'énergie, aussi appelée *energy function* en anglais, permettant d'évaluer la similarité des représentations latentes des entrées X_1 et X_2.
— L_G : la fonction de coût pour les paires positives, aussi appelée fonction de coût *genuine* en anglais, permet d'entraîner le réseau siamois lorsque X_1 et X_2 proviennent de la même source, dite paire positive.
— L_I : la fonction de coût pour les paires négatives, aussi appelée fonction de coût *impostor* en anglais, permettant d'entraîner le réseau siamois lorsque X_1 et X_2 ne proviennent pas de la même source, dite paire négative.
— y : une étiquette, aussi appelée *label* en anglais, indiquant si la paire (X_1, X_2) est une paire positive ($y = 1$) ou négative ($y = 0$).

Le principe de ce type de fonction de coût est de minimiser l'énergie des paires positives et de maximiser l'énergie des paires négatives. Plus simplement, les représentations latentes des éléments constituant une paire positive se retrouveront proches dans l'espace d'enchâssement, alors que les éléments constituant une paire négative se retrouveront éloignés dans l'espace d'enchâssement. Cela permettra, lors de l'inférence, de déterminer si une paire d'entrées est une paire positive ou négative, en utilisant la fonction énergie $E_W(X_1, X_2)$.

Dans notre cas, nous nous sommes basés sur la fonction de coût introduite dans [122]. Dans cet article, R. Hadsell, S. Chopra et Y. Le Cun proposent une fonction de coût contrastive pour la réduction de dimensionnalité d'images, qui sera réutilisée dans d'autres domaines comme pour le casting de voix [123].

La contrainte associée à la fonction de coût contrastive peut être exprimée de la manière suivante : $E_W(X_1, X_2) + m < E_W(X_1, X_2')$ avec (X_1, X_2) une paire positive et (X_1, X_2') une paire négative et m une constante, appelée marge. Ainsi, pour contraindre le réseau siamois à respecter cette propriété, la fonction de coût proposée par les auteurs est la suivante :

$$L(W) = y\|G_W(X_1) - G_W(X_2)\|_2^2 + (1-y)max(0, m - \|G_W(X_1) - G_W(X_2)\|_2^2) \quad (3.2)$$

Pour l'apprentissage de notre réseau siamois contrastif, nous avons fixé la marge m égale à 1. Ainsi, les paires positives ont une distance proche de 0 dans l'espace d'enchâssement, alors que les paires négatives ont une distance supérieure ou égale à 1 dans l'espace d'enchâssement. De plus, la mesure de similarité $s(.,.)$ de ce paradigme est la fonction énergie $E_W(X_1, X_2)$ précédemment introduite. La structure du réseau siamois contrastif est ainsi présentée en figure 3.6.

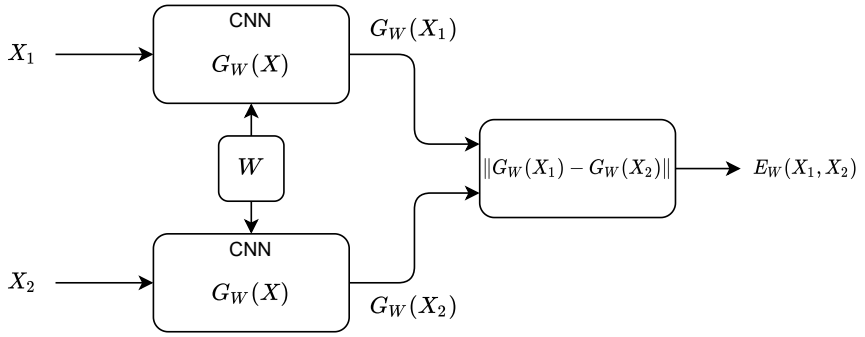


FIGURE 3.6 – Architecture d'un réseau siamois contrastif

Régression logistique Le paradigme de régression logistique provient des travaux de Koch et al. [124] sur le *one-shot learning* basé sur des réseaux siamois pour la reconnaissance de caractères manuscrits. Cependant, avant de présenter les particularités de ce dernier paradigme, il est d'abord nécessaire de faire un rappel sur la notion de régression logistique. Ce terme est utilisé en apprentissage automatique pour nommer une méthode de régression utilisée en classification binaire [31]. Elle consiste à prédire la probabilité $p(X) = \mathbb{P}(y = 1|X)$, avec y l'étiquette associée à X et $\mathbb{P}(y = 0|X) = 1 - \mathbb{P}(y = 1|X)$, comme une combinaison linéaire des descripteurs composant l'observation X :

$$\mathbb{P}(y = 1|X) = \sigma(\sum_i \alpha_i X^{(i)} + \alpha_0) \quad (3.3)$$

avec

- $\sigma(x) = \frac{1}{1+e^{-x}}$: la fonction sigmoïde présentée en figure 3.7.
- $X^{(i)}$: le i-ème descripteur associé au vecteur X.
- α_i : le coefficient associé au descripteur $X^{(i)}$.
- α_0 : le terme de biais.

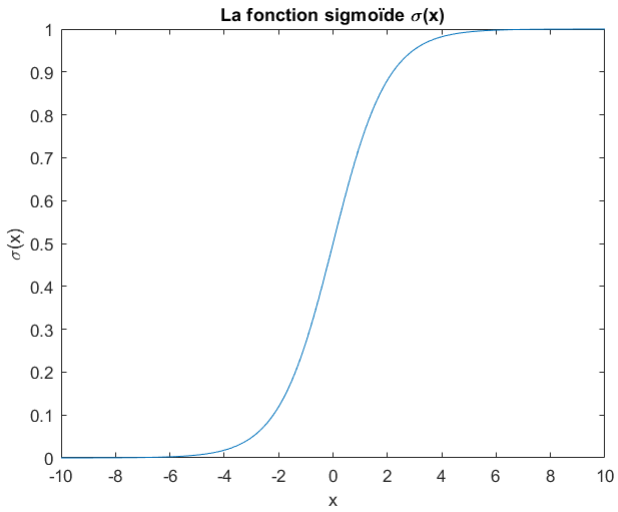


FIGURE 3.7 – La fonction sigmoïde

La procédure d'entraînement de cet algorithme est basée sur l'utilisation de la fonction d'entropie binaire croisée :

$$L(\{\alpha_i\}_i) = -ylog(p(X)) - (1-y)log(1-p(X)) \tag{3.4}$$

Dans le paradigme d'apprentissage de réseaux siamois utilisant la régression logistique, en plus d'apprendre un espace d'enchâssement $G_W(.)$, la fonction de similarité basée sur les représentations latentes $G_W(X_1)$ et $G_W(X_2)$ est aussi apprise. Ainsi, la sortie du réseau siamois prédit la probabilité que la paire (X_1, X_2) soit positive[6] :

$$p(X_1, X_2) = \sigma(\sum_i \alpha_i |G_W(X_1)^{(i)} - G_W(X_1)^{(i)}| + \alpha_0) \tag{3.5}$$

avec $G_W(X_1)^{(i)}$ la i-ème dimension de la projection de l'entrée X dans l'espace d'enchâssement.

6. Ici, $p(X_1, X_2)$ correspondant à $s(X_1, X_2)$ et le terme α_0 est généralement négligé.

Ainsi, la sortie du réseau siamois correspond à la probabilité que les deux entrées X_1 et X_2 proviennent d'une même source, i.e. la probabilité qu'il s'agisse d'une paire positive. De plus, la fonction de similarité apprise est appelée norme L_1 pondérée, car il s'agit d'une norme L_1 terme à terme des représentations latentes $G_W(X_1)$ et $G_W(X_2)$ suivie d'une régression logistique, d'où le terme «pondérée» correspondant aux α_i.

Tout comme pour la régression logistique, les auteurs utilisent l'entropie binaire croisée pour l'entraînement de leur modèle :

$$L(W) = -ylog(p(X_1, X_2)) - (1 - y)log(1 - p(X_1, X_2)) \tag{3.6}$$

La structure du réseau siamois basée sur la régression logistique est présentée en figure 3.8.

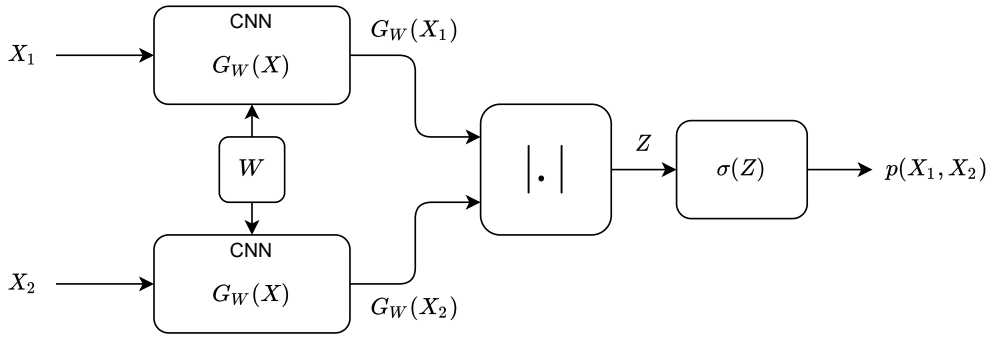


FIGURE 3.8 – Réseau siamois basé sur une régression logistique

L'apprentissage par transfert contrastif Ce paradigme a été proposé pour une tâche de reconnaissance faciale par Taigman et al. dans [117]. Leur approche, appelée DeepFace, est décrite dans l'article de la manière suivante : «Une fois appris, le réseau de reconnaissance faciale (sans la dernière couche) est répliqué deux fois (un pour chaque image d'entrée) et les descripteurs sont utilisés pour prédire directement si les deux images d'entrée appartiennent à la même personne.» (traduit de l'anglais à partir de [117]). Plus particulièrement, les auteurs avaient utilisé la norme L_1 pondérée, que nous avons présentée dans la sous-section précédente en citant les travaux de Koch et al. [124] sur le paradigme d'apprentissage logistique[7]. À l'inverse, nous avons choisi d'utiliser le paradigme d'ap-

7. À noter que lors de l'entraînement du réseau siamois, les auteurs ont uniquement permis le réentraînement des deux dernières couches.

prentissage contrastif, d'où le nom de cette sous-section : l'apprentissage par transfert contrastif. En effet, lors de nos tests préliminaires, l'utilisation d'un paradigme contrastif à la place du paradigme logistique avait donné de meilleurs résultats.

Pour l'entraînement d'un réseau de neurones en utilisant le paradigme d'apprentissage par transfert contrastif, il faut réaliser trois étapes que nous avons définies :

— Entraîner un réseau de neurones pour une tâche de classification, dans notre cas, pour du RFF.

— Extraire un sous-réseau constitué des couches du réseau de neurones, hormis les deux dernières couches et rajouter une dernière couche dense de D neurones représentant l'espace d'enchâssement.

— Apprendre les poids de la dernière couche à l'aide du paradigme contrastif, les poids des autres couches étant figés.

La figure 3.9 présente le principe de fonctionnement de l'apprentissage par transfert contrastif.

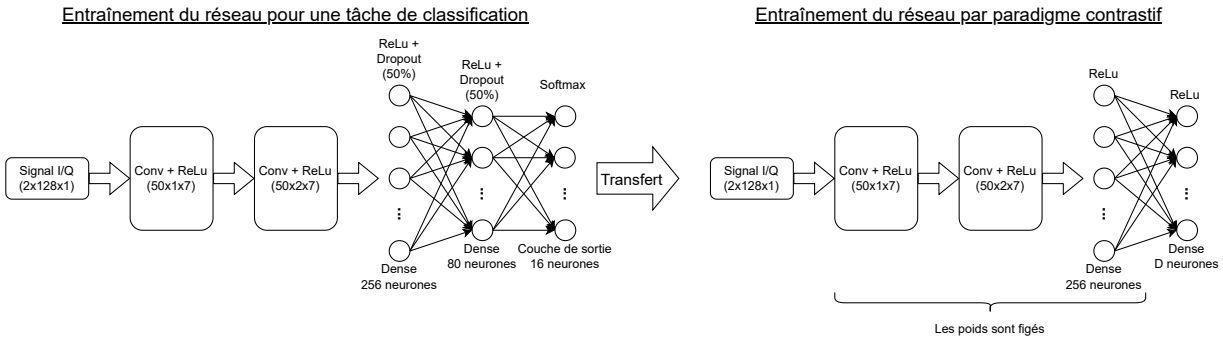


FIGURE 3.9 – Principe de fonctionnement de l'apprentissage par transfert contrastif

Ces travaux peuvent être vus comme de l'apprentissage par transfert, aussi appelé *transfer learning*, consistant à entraîner un algorithme sur une première tâche et à transférer les connaissances apprises pour la résolution d'une seconde tâche considérée comme proche de la première. En apprentissage profond, l'apprentissage par transfert est essentiellement du transfert dit inductif [125]. En général, cela consiste à réutiliser les premières couches d'un réseau de neurones entraîné pour une première tâche (dit réseau de base) comme extracteur de descripteurs pour un second réseau de neurones qui réalise une seconde tâche. En effet, les descripteurs appris par les premières couches sont considérés comme étant assez généralistes pour être réutilisés pour la seconde tâche.

Dans notre cas, nous nous sommes inspirés des travaux de Taigman et al., nous n'avons

cependant pas utilisé comme fonction de similarité une norme L_1 pondérée, mais la norme L_2 à cause de l'utilisation du paradigme contrastif et non du paradigme logistique. Ainsi, au lieu d'entraîner le réseau à l'aide de l'entropie croisée, nous avons utilisé la fonction de coût contrastive précédemment introduite dans 3.2.1.2 avec une valeur de marge $m = 1$, comme précédemment. De plus, nous avons réentraîner uniquement les paramètres de la dernière couche (de D neurones) représentant l'espace d'enchâssement. En effet, cela revient à considérer que l'extracteur de descripteurs appris par les trois premières couches est assez généraliste et qu'il n'est donc pas nécessaire de le réentraîner, c'est le but de l'apprentissage par transfert. L'intérêt de cette approche réside dans le fait qu'on apprenne à résoudre une tâche largement explorée dans la littérature du RFF, comme la classification, et qu'on réutilise ces connaissances pour résoudre une tâche connexe mais peu explorée, comme l'apprentissage d'une fonction de similarité.

3.2.1.3 Décision

Maintenant que nous avons présenté les architectures utilisées, ainsi que les différents paradigmes d'apprentissage des réseaux siamois, il est nécessaire de présenter la prise de décision à l'aide de réseaux siamois lors de la phase d'inférence. Pour cela, nous allons prendre un point de vue biométrique et différencier la prise de décision en distinguant deux cas : la vérification et l'identification.

Vérification Nous allons maintenant présenter les différentes manières de réaliser une vérification selon les différents paradigmes d'apprentissage que nous avons considérés dans ce manuscrit.

Il est important de noter que nous avons aussi utilisé ces règles de décision lors de l'apprentissage pour déterminer l'étiquette inférée par le réseau siamois.

Fonction de coût contrastive Pour une vérification entre l'entrée X_1 et l'entrée X_2 pour le paradigme basé sur la fonction de coût contrastive, nous avons défini la règle suivante [8] :

— $E_W(X_1, X_2) < 0.5 \rightarrow \hat{y} = 1$: paire positive.
— $E_W(X_1, X_2) \geq 0.5 \rightarrow \hat{y} = 0$: paire négative.

8. Pour rappel, la marge $m = 1$, donc la valeur de 0.5 correspond à la moitié de la marge.

Régression logistique Pour une vérification entre l'entrée X_1 et l'entrée X_2 pour la régression logistique, nous avons défini la règle suivante :

— $p(X_1, X_2) > 0.5 \rightarrow \hat{y} = 1$: paire positive.

— $p(X_2, X_2) \leq 0.5 \rightarrow \hat{y} = 0$: paire négative.

Apprentissage par transfert contrastif Pour une vérification entre l'entrée X_1 et l'entrée X_2 pour le paradigme de transfert contrastif, nous avons défini la règle suivante [9] :

— $E_W(X_1, X_2) < 0.5 \rightarrow \hat{y} = 1$: paire positive.

— $E_W(X_1, X_2) \geq 0.5 \rightarrow \hat{y} = 0$: paire négative.

Identification Le principe de fonctionnement de l'identification pour les réseaux siamois est présenté en figure 3.10 [126]. Plus particulièrement, la base d'empreintes est composée d'un signal I/Q X_i par émetteur légitime, avec $i \in [\![1\ ;\ C]\!]$ le numéro de l'émetteur. Ainsi, lors de la phase d'identification, pour chaque entrée de la base d'empreintes, une vérification est réalisée entre le signal d'entrée X et le signal X_i de la base d'empreintes. À la fin de cette série de vérifications, trois cas sont possibles :

— Aucun signal de la base d'empreintes $\{X_i\}_{i\in[\![1\ ;\ C]\!]}$ ne correspond au signal X. Dans ce cas, le signal d'entrée X n'est pas considéré comme provenant d'un émetteur légitime.

— Un unique signal de la base d'empreintes $\{X_i\}_{i\in[\![1\ ;\ C]\!]}$ correspond au signal X. Dans ce cas, le signal d'entrée X est considéré comme provenant d'un émetteur légitime et correspondant à l'émetteur auquel le signal de la base d'empreintes appartient.

— Plusieurs signaux de la base d'empreintes $\{X_i\}_{i\in[\![1\ ;\ C]\!]}$ correspondent au signal X. Dans ce cas, le signal d'entrée X est considéré comme provenant d'un émetteur légitime, mais il faut choisir le signal de la base d'empreintes correspondant à la mesure de similarité la plus faible.

Dans son cours [126], A. Ng précise qu'il est possible, de stocker les projections des signaux dans l'espace d'enchâssement $\{G_W(X_i)\}_{i\in[\![1\ ;\ C]\!]}$, à la place de stocker les signaux $\{X_i\}_{i\in[\![1\ ;\ C]\!]}$. Ainsi, pour chaque identification d'un signal X, il n'est plus nécessaire de faire $C+1$ projections dans l'espace d'enchâssement, mais seulement une unique projection de l'entrée X. En effet, il n'est pas nécessaire de réeffectuer la projection des signaux $\{X_i\}_{i\in[\![1\ ;\ C]\!]}$, car les projections des signaux de la base d'empreintes ne changent pas.

9. Idem, la marge $m = 1$, donc la valeur de 0.5 correspond à la moitié de la marge.

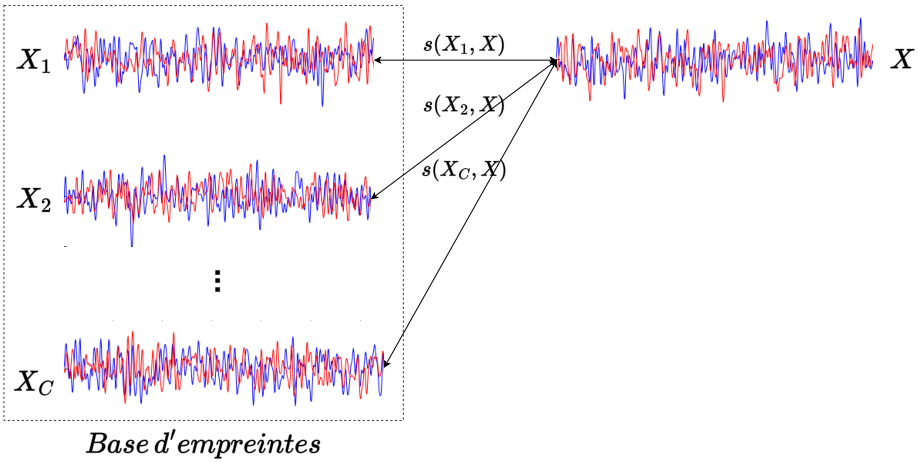


FIGURE 3.10 – Identification pour les réseaux siamois

3.2.2 Expérimentations

Dans cette sous-section, nous allons tout d'abord présenter le jeu de données initial que nous avons utilisé, ainsi que les étapes d'adaptation que nous avons réalisées pour obtenir le jeu de données nécessaire à l'entraînement et l'évaluation des différents réseaux siamois. De plus, nous avons réalisé plusieurs expérimentations pour évaluer les performances des différentes architectures, ainsi que des différents paradigmes d'apprentissage. La métrique utilisée pour l'évaluation des performances des différents paradigmes est l'*accuracy* (ou justesse), correspondant à la proportion de bonnes prédictions, à l'aide des règles de vérification présentées dans la section 3.2.1.3. Ce choix se justifie car il s'agit d'une métrique populaire en apprentissage automatique, mais qui est aussi utilisée dans la littérature du RFF, notamment pour les réseaux siamois [127].

3.2.2.1 Jeu de données et adaptation

Cette sous-section présente le jeu de données initial utilisé pour créer le jeu de données siamois à l'aide d'un jeu de données intermédiaire. En effet, le jeu de données initial étant adapté à une tâche de classification, il a été nécessaire de le transformer à l'aide de différents traitements pour qu'il soit adapté à l'entraînement de réseaux siamois. Une première étape a permis d'extraire un jeu de données intermédiaire à partir du jeu de données initial et une seconde étape a permis d'obtenir le jeu de données siamois à partir du jeu de données intermédiaire.

Jeu de données initial Le jeu de données initial[10] appelé «*Datasets for RF Fingerprinting of Bit-similar USRP X310 Radios*», a été produit et utilisé par K. Sankhe et al. dans [29] afin d'entraîner des réseaux de neurones profonds pour une tâche de classification en RFF. À l'époque de nos expérimentations et à notre connaissance, ce jeu de données était le seul jeu de données réelles disponible.

Il est composé de signaux provenant de 16 émetteurs SDR USRP X310 et enregistrés à l'aide d'un unique récepteur SDR USRP B210. La forme d'onde utilisée est conforme à la norme IEEE 802.11 et a été générée grâce à la WLAN System Toolbox de Matlab. Les paquets transmis sont constitués d'un champ d'adresse identique pour tous les émetteurs et d'un *payload* aléatoire. Le récepteur a collecté les échantillons I/Q à 5 Msps avec une fréquence centrale f_0 de 2.45 GHz. Les auteurs ont ainsi collecté 20 millions d'échantillons (4 secondes) pour chaque émetteur. De plus, les auteurs proposent deux configurations expérimentales :

— *Over-the-air transmissions* où les échantillons ont été transmis par les USRP X310 à travers un canal radiofréquence et collectés par le récepteur B210.
— *Over-the-cable transmissions* où les échantillons ont été transmis par les USRP X310 à travers un câble et collectés par le récepteur B210 puis égalisés.

Dans le cas de la configuration *over-the-air*, les auteurs ont réalisé plusieurs expérimentations en fonction de la distance allant de 2 *ft* (pied) à 62 *ft*, chaque expérimentation étant séparée de 6 *ft*. Ainsi, en prenant en compte toutes les distances, il y a 11 expérimentations différentes. Deux enregistrements, appelées *run*, ont été effectués pour chaque distance et chaque émetteur.

Les données ont été enregistrés dans un format, appelé format I/Q flottant 128 bits, qui consiste à représenter les échantillons I avec un format flottant 64 bits (double précision) et les échantillons Q avec un format flottant 64 bits (double précision).

Dans notre cas, nous nous intéressons à la configuration expérimentale *over-the-air*, car c'est la seule qui corresponde à une transmission sans-fil.

Jeu de données intermédiaire Avant de pouvoir créer le jeu de données siamois, il est nécessaire de faire un pré-traitement à partir du jeu de données initial. En effet, le jeu de données est constitué de $2 \times 11 \times 16 \times 20 \times 10^6 = 7.04 \times 10^9$ échantillons. Nous avons donc choisi d'utiliser les signaux pour la distance de 2 *ft*, car il s'agit des signaux les moins

10. Le jeu de données est disponible sur : https ://genesys-lab.org/oracle (dernière consultation en octobre 2022)

bruités. Cela permet d'évaluer la possibilité d'utiliser des réseaux siamois pour le RFF sur des signaux réels. De plus, nous avons utilisé le *run 2* pour cette distance, car nous avons considéré qu'il était de meilleure qualité après une inspection des signaux. Enfin, pour chaque émetteur, nous avons récupéré les 10.24 premières millisecondes correspondant à 512 000 échantillons, que nous avons décomposées en 4 000 signaux non recouvrants de 128 échantillons [11]. Ainsi, le jeu de données intermédiaire est composé de 64 000 signaux (4 000 par émetteur) de 128 échantillons I/Q et chaque signal X_i est associé à une étiquette y_i. Ce jeu de données est considéré comme ayant assez d'observations par classe pour un problème de classification. En effet, comme nous l'avions mentionné précédemment, une heuristique classique en apprentissage profond considère qu'il est nécessaire d'avoir au moins un millier d'exemples par classe [111].

Jeu de données siamois Le jeu de données intermédiaire n'est pas approprié pour l'entraînement des réseaux siamois, mais plutôt pour l'entraînement de réseaux de neurones pour une tâche de classification en RFF. Ainsi, pour pouvoir entraîner le réseau siamois, il est nécessaire d'avoir un jeu de données constitué d'une paire de signaux I/Q (X_1, X_2) et d'une étiquette y indiquant s'il s'agit d'une paire positive ou d'une paire négative. Cette étiquette y est égale à 1 si la paire de signaux I/Q provient du même émetteur (paire positive) ou égale à 0 si la paire de signaux I/Q ne provient pas du même émetteur (paire négative). La stratégie utilisée pour la création du jeu de données siamois est inspirée des stratégies présentées dans [6], [7], [122]. Cette stratégie impose notamment que le nombre de paires positives soit égal au nombre de paires négatives pour pouvoir avoir un jeu de données équilibré.

Le processus de construction du jeu de données siamois peut être décrit de la manière suivante : pour chaque signal du jeu de données intermédiaire (observation), nous choisissons $P = 5$ signaux provenant du même émetteur pour créer P paires positives (avec un tirage sans remise [12]) et nous choisissons P signaux provenant d'un émetteur différent pour créer P paires négatives (avec un tirage sans remise). Nous appliquons ainsi ce processus à tous les signaux du jeu de données intermédiaire. Ainsi, en supposant qu'il y ait $L = 64\,000$ signaux dans la base de données intermédiaire, le nombre de paires du jeu de données siamois est constitué de $2 \times L \times P = 640\,000$ paires : 320 000 paires positives et

11. Le choix de signaux de 128 échantillons étant justifié par la couche d'entrée de nos architectures de réseaux siamois.

12. Le signal en question est enlevé du choix pour la création de la paire positive, pour éviter que la paire soit constituée des deux mêmes entrées.

320 000 paires négatives.

En réalité, le jeu de données intermédiaire est d'abord découpé en trois jeux de données : un jeu de données d'entraînement, un jeu de données de validation, ainsi qu'un jeu de données de test. Puis, pour chacun de ces jeux, le processus de construction de jeu de données siamois est appliqué. Cela permet d'avoir une meilleure indépendance entre les différents jeux de données siamois résultants.

Ainsi, les compositions des différents jeux de données siamois sont les suivantes :

— Jeu d'entraînement : 360 000 paires, dont 180 000 paires positives et 180 000 paires négatives.

— Jeu de validation : 120 000 paires, dont 60 000 paires positives et 60 000 paires négatives.

— Jeu de test : 160 000 paires, dont 80 000 paires positives et 80 000 paires négatives.

3.2.2.2 Expérimentation 1 : architecture de référence (CAID 2020 : 32 époques)

Cette expérimentation (exp. 1) consiste à évaluer l'*accuracy* des différents paradigmes d'apprentissage précédemment introduits sur l'architecture de référence présentée en figure 3.4. Ces résultats ont été présentés dans notre publication sur les réseaux siamois pour l'authentification par empreinte radio sur des signaux I/Q [128].

Pour cette expérimentation, nous avons utilisé une optimisation Adam comme dans [29]. De plus, nous avons choisi un nombre de 32 époques (ou *epochs*), correspondant au nombre de fois que les poids sont appris sur l'intégralité du jeu d'entraînement. Le nombre d'époques peut être vu comme le nombre de cycles d'apprentissage sur l'intégralité du jeu d'entraînement. Enfin, nous avons choisi une taille de mini-lots (ou *mini-batch*) de taille 128, c'est-à-dire le nombre d'observations correspond au nombre d'exemples nécessaires pour mettre à jour les poids [31].

Le tableau 3.2 présente les valeurs des différents hyperparamètres qui donnent les meilleurs résultats pour chaque paradigme d'apprentissage. De plus, les performances de chaque paradigme sont présentées dans le tableau 3.3. Nous pouvons constater que pour cette expérimentation, la régression logistique obtient les meilleurs résultats avec une *accuracy* de 99.52%. Ensuite, le paradigme basé sur la fonction de coût contrastive obtient aussi de bons résultats avec une *accuracy* de 97.44 %. Enfin, le paradigme d'apprentissage par transfert contrastif obtient les moins bons résultats avec une *accuracy* de 93.3 %. Les moins bons résultats de l'apprentissage par transfert contrastif peuvent s'expliquer par le fait que les couches d'extraction de descripteurs apprises lors de la tâche de classifica-

tion sont potentiellement moins bien adaptées pour une mesure de similarité que celles apprises par les deux autres paradigmes, apprenant directement l'ensemble des poids du réseau siamois. Cependant, durant nos expérimentations, c'est le paradigme par transfert contrastif qui nous a permis d'obtenir les premiers résultats préliminaires satisfaisants, notamment à cause de l'utilisation de l'apprentissage par transfert. Il est difficile d'essayer d'expliquer davantage les résultats obtenus, car dépendants d'un nombre de paramètres trop important.

Paradigme d'apprentissage	μ	l	D
Régression logistique	0.0001	0.00001	128
Fonction de coût contrastive	0.001	0.0001	128
Transfert contrastif	0.001	0.0001	128

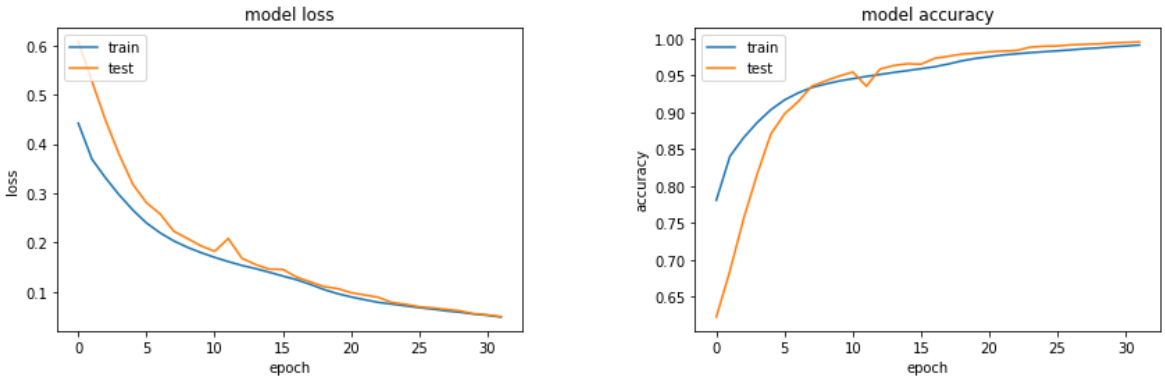
TABLE 3.2 – Valeurs des hyperparamètres pour chaque paradigme d'apprentissage (exp. 1)

Paradigme d'apprentissage	Train accuracy	Test accuracy
Régression logistique	0.9909	0.9952
Fonction de coût contrastive	0.9669	0.9744
Transfert contrastif	0.9195	0.933

TABLE 3.3 – Performances pour chaque paradigme d'apprentissage (exp. 1)

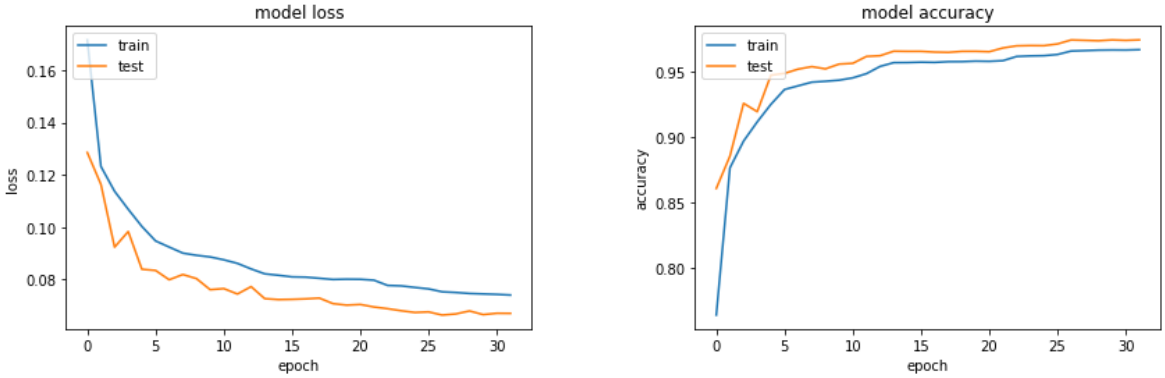
Les historiques d'apprentissage sont présentés pour la régression logistique en figure 3.11, pour la fonction de coût contrastive en figure 3.12 et pour le transfert contrastif en figure 3.13. Nous pouvons remarquer que pour les trois paradigmes, les performances sur le jeu d'apprentissage sont globalement moins bonnes que sur le jeu de test. Ce phénomène, que nous avions expliqué dans [128], est dû aux techniques de régularisations utilisées : la régularisation L_2, ainsi que le *dropout*.

Le nombre de poids, ainsi que de multiplications de la meilleure architecture sont donnés dans le tableau 3.4. Nous pouvons constater que le nombre de paramètres est supérieur aux nombres maximums de paramètres acceptables pour un gros réseau de neurones, comme indiqué dans le tableau 3.1. Cependant, le nombre d'opérations nécessaires est relativement faible et similaire à celui d'un petit réseau de neurones. Cela peut s'expliquer par la taille limitée de l'entrée du sous-réseau par rapport à ceux de [113], [114].



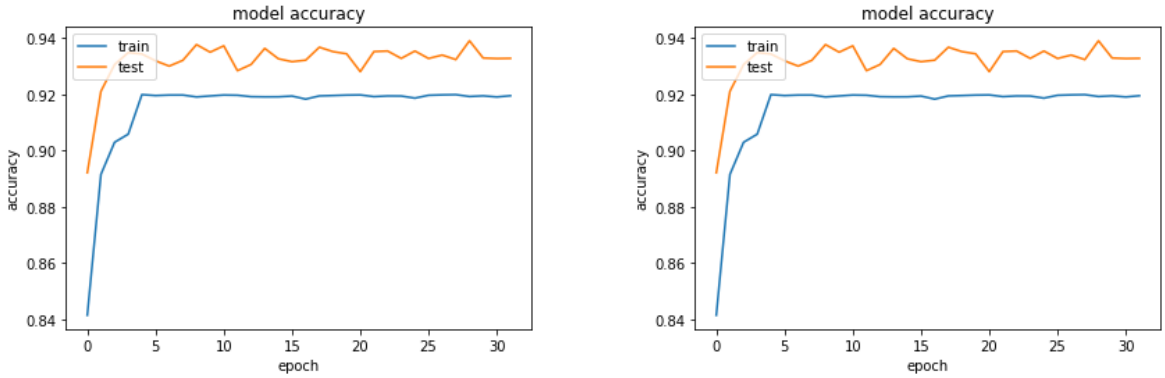
(a) Fonction de coût (b) Justesse

FIGURE 3.11 – Historique d'apprentissage : régression logistique (exp. 1)



(a) Fonction de coût (b) Justesse

FIGURE 3.12 – Historique d'apprentissage : fonction de coût contrastive (exp. 1)



(a) Fonction de coût (b) Justesse

FIGURE 3.13 – Historique d'apprentissage : transfert contrastif (exp. 1)

Nom	Type	Entrée	Paramètres	Opérations
input	Input	(2,128,1)	-	-
conv2D_1	Conv2D	(2,128,1)	400	85 400
conv2D_2	Conv2D	(2,122, 50)	35 050	4 060 000
flatten	Flatten	(1,116,50)	-	-
dense_1	Dense	5 800	1 485 056	1 484 800
dropout_1	Dropout	256	-	-
dense_2	Dense	256	32 896	32 768
dropout_2	Dropout	128	-	-
Total			1 553 402	5 662 968

TABLE 3.4 – Évaluation de la complexité de la meilleure architecture de sous-réseau pour l'expérimentation 1

3.2.2.3 Expérimentation 2 : architecture de référence (92 époques)

La seconde expérimentation réalisée (exp. 2) consiste à mesurer l'effet de l'augmentation du nombre d'époques sur les performances, par rapport aux résultats de l'expérimentation précédente, présentée en section 3.2.2.2. En effet, comme nous pouvons le constater sur les figures 3.11 et 3.12, les courbes de fonction de coût ne semblent pas avoir atteint un plateau au bout de 32 époques, contrairement à la fonction de coût présentée à la figure 3.13. Ainsi, l'augmentation du nombre d'époques permettrait d'améliorer les performances des réseaux de neurones basés sur un paradigme contrastive et/ou sur une régression logistique. Ces résultats, datant de fin août/début septembre 2020, n'avaient pas pu être incorporés à notre publication sur les réseaux siamois [128] par manque de temps.

Pour cette expérimentation, nous avons utilisé une optimisation Adam avec une taille de mini-lots de taille 128 sur 92 époques. De plus, pour la recherche d'hyperparamètres, nous avons utilisé une recherche par quadrillage avec un jeu de validation. Les valeurs des hyperparamètres sont les mêmes que dans la sous-section précédente. Enfin, nous n'avons évalué que les performances du paradigme d'apprentissage utilisant la fonction de coût contrastive, ainsi que celui basé sur la régression logistique. En effet, ces deux paradigmes avaient obtenu des résultats supérieurs à ceux du paradigme d'apprentissage par transfert contrastif dans l'expérimentation précédente. De plus, dans la figure 3.13, on pouvait observer que la fonction de coût avait déjà atteint un plafond et qu'il ne semblait pas nécessaire de réévaluer ce paradigme avec un nombre d'époques plus important.

Le tableau 3.5 présente les valeurs des différents hyperparamètres donnant les meilleurs

résultats pour chaque paradigme d'apprentissage et le tableau 3.6 présente les meilleures performances obtenues pour chaque paradigme. Nous pouvons constater que pour cette expérimentation, la fonction de coût contrastive donne les meilleures performances avec une *accuracy* de 100% alors que le paradigme d'apprentissage basé sur la régression logistique obtient 99.99 %. Ainsi, cette expérimentation montre que l'augmentation du nombre d'époques a permis aux réseaux siamois appris d'obtenir de meilleures performances, que ce soit pour le paradigme utilisant la fonction de coût contrastif ou pour le paradigme utilisant la régression logistique. De la même manière que pour l'expérimentation précédente, il est difficile d'expliquer davantage les résultats obtenus, car dépendants d'un nombre de paramètres trop important.

Paradigme d'apprentissage	μ	l	D
Régression logistique	0.001	0.00001	80
Fonction de coût contrastive	0.001	0.00001	128

TABLE 3.5 – Valeurs des hyperparamètres pour chaque paradigme d'apprentissage (exp. 2)

Paradigme d'apprentissage	Train accuracy	Test accuracy
Régression logistique	0.9993	0.9999
Fonction de coût contrastive	0.9971	1.0

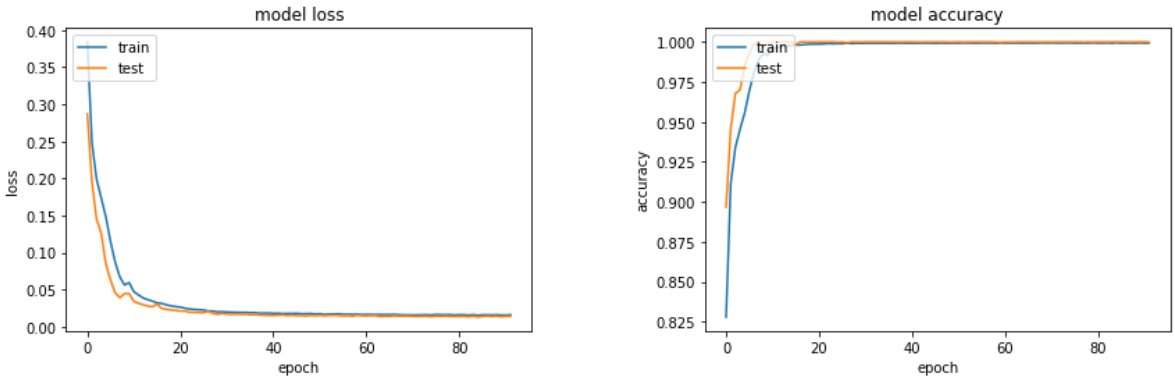
TABLE 3.6 – Performances pour chaque paradigme d'apprentissage (exp. 2)

Les historiques d'apprentissage sont présentés pour la régression logistique en figure 3.14 et ceux de la fonction de coût contrastive en figure 3.15. Tout comme pour l'expérience précédente, nous pouvons observer un décalage entre les performances sur le jeu d'entraînement et le jeu de test.

Le nombre de poids, ainsi que de multiplications de la meilleure architecture sont les mêmes que pour l'expérimentation 1, qui sont donnés dans le tableau 3.4. Ainsi, il est ainsi possible de se référer à l'analyse faite précédemment.

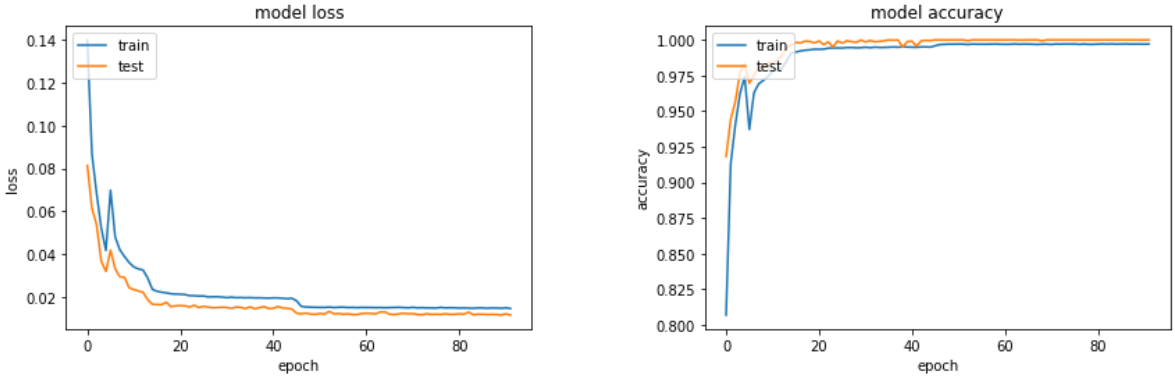
3.2.2.4 Expérimentation 3 : architecture légère (32 époques)

Une dernière expérimentation (exp. 3) a été effectuée pour évaluer les performances de l'architecture légère. Ces résultats sont présentées en figure 3.5 et ils sont comparés aux



(a) Fonction de coût (b) Justesse

FIGURE 3.14 – Historique d'apprentissage : régression logistique (exp. 2)



(a) Fonction de coût (b) Justesse

FIGURE 3.15 – Historique d'apprentissage : fonction de coût contrastive (exp. 2)

résultats de la première expérimentation basée sur l'architecture de référence, présentée en section 3.2.2.2. Les résultats de cette expérimentation, datant de fin août/début septembre 2020, n'avaient pas été incorporés à notre publication sur les réseaux siamois [128] par manque de temps.

Pour cette expérimentation, nous avons utilisé une optimisation Adam avec des mini-lots de taille 128 sur 32 époques, comme pour la première expérimentation.

L'architecture utilisée, présentée en figure 3.5, a environ 4 fois moins de paramètres que l'architecture des expérimentations précédentes. En effet, en considérant la taille de l'espace d'enchâssement égale à $D = 128$, l'architecture de référence est constituée de 1 553 402 poids à apprendre (voir tableau 3.4). Alors que l'architecture légère est constituée de 381 306 poids à apprendre, soit une réduction de 75% par rapport à l'architecture de

référence. Cette réduction de dimension est principalement due aux couches de *pooling* qui ont pour effet de réduire les dimensions des cartes de caractéristiques et donc le nombre de paramètres de la première couche dense.

Cette expérimentation, réalisée en parallèle de la seconde expérimentation, considérait le cas du paradigme de régression logistique pour $D = 128$ (voir tableau 3.7), i.e le paradigme ayant obtenu les meilleures performances lors de la première expérimentation. Nous pouvons constater sur le tableau 3.8 que les performances du réseau de neurones (99.46%) sont proches des performances obtenues lors de la première expérimentation (99.52 %). De plus, l'historique d'apprentissage pour cette expérimentation est présenté en figure 3.16. Tout comme pour les expérimentations précédentes, on voit une différence notable entre les performances obtenues lors de l'apprentissage et du test. Cette expérimentation montre qu'il est possible d'avoir de bons résultats avec une architecture plus légère que l'architecture de référence des expérimentations précédentes. En effet, lors de l'expérimentation 1 nous avions obtenu une *accuracy* de 99.52 %, ce qui correspond à une diminution de 0.06 % pour un réseau ayant environ 4 fois moins de poids.

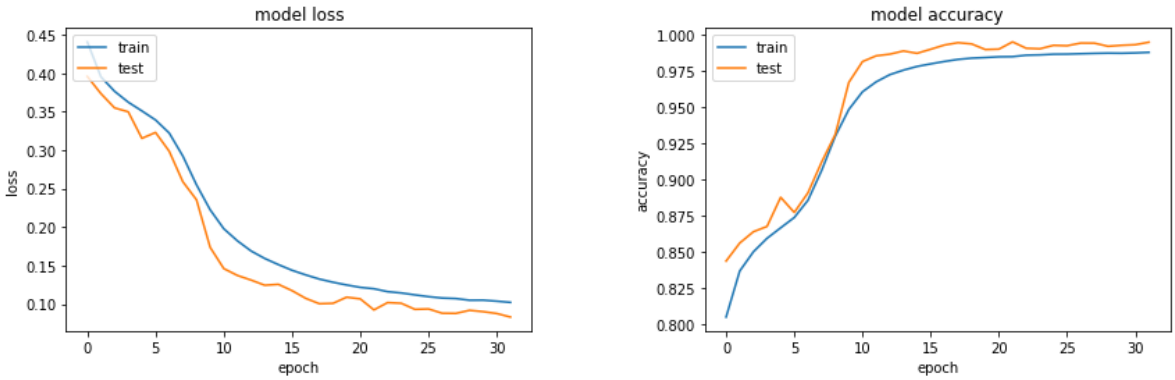
Paradigme d'apprentissage	μ	l	D
Régression logistique	0.001	0.0001	-

TABLE 3.7 – Valeurs des hyperparamètres pour chaque paradigme d'apprentissage (exp. 3)

Paradigme d'apprentissage	Train accuracy	Test accuracy
Régression logistique	0.9875	0.9946

TABLE 3.8 – Performances pour chaque paradigme d'apprentissage (exp. 3)

Le nombre de poids de l'architecture et le nombre de multiplications nécessaire à une inférence sont donnés dans le tableau 3.9. Nous pouvons remarquer que par rapport à l'architecture de référence, le nombre de poids a été réduit d'un facteur 4 et le nombre de multiplications d'un facteur 2.4. Ainsi, l'architecture légère peut être considérée comme une architecture large au sens du *TinyML*, mais qui répond aux critères de complexité définis dans le tableau 3.1. En effet, le nombre de paramètres de l'architecture est bien inférieur à 500 000 et le nombre de multiplications et bien inférieur à 80 000 000.



(a) Fonction de coût (b) justesse

FIGURE 3.16 – Historique d'apprentissage : régression logistique (exp. 3)

Nom	Type	Entrée	Paramètres	Opérations
input	Input	(2,128,1)	-	-
conv2D_1	Conv2D	(2,128,1)	400	85 400
max_pooling_1	(2,122,50)	-	-	
conv2D_2	Conv2D	(2,61,50)	35 050	1 925 000
max_pooling_2	(1,55,50)	-	-	
flatten	Flatten	(1,27,50)	-	-
dense_1	Dense	1 350	345 856	345 600
Total			381 306	2 356 000

TABLE 3.9 – Evaluation de l'implémentation logicielle de l'architecture légère

3.2.3 Comparaison avec la littérature

Dans cette sous-section, nous allons comparer nos résultats à ceux de la littérature. Pour cela, nous allons réaliser quatre comparaisons différentes. La première comparaison correspond à celle que nous avions publiée dans [128] lors du CAID 2020. La seconde, correspond à une comparaison de notre méthode avec les travaux sur les réseaux siamois en RFF publiés après la publication de notre article. Enfin, les deux dernières seront des comparaisons plus classiques, concernant la scalabilité et la complexité de notre approche.

3.2.3.1 Comparaison publiée dans le CAID 2020

Lors de la publication de notre article [128], seul un article traitait de l'utilisation des réseaux siamois pour le RFF. Dans cet article [127], Langford et al. ont utilisé un réseau siamois traitant des spectrogrammes de signaux I/Q. Leur réseau avait été entraîné à

l'aide d'une fonction de coût contrastive décrite dans [7]. Les performances que nous avions présentées dans notre article (soit 99.52 %, voir section 3.2.2.2) étaient inférieures à celles obtenues par les auteurs de [127] : 99.79%. Cependant, notre approche a plusieurs avantages par rapport à la leur. D'une part, notre architecture traite directement les signaux I/Q alors que l'architecture présentée dans [127] utilise une représentation temps-fréquence comme représentation d'entrée, nécessitant donc une phase de pré-traitement. D'autre part, nos performances ont été obtenues à partir de données réelles provenant de 16 émetteurs, alors que les travaux de Langford étaient basés sur des données provenant de 4 émetteurs et qui nous semblaient beaucoup plus contrôlées, i.e un seul défaut : le décalage fréquentiel. Ainsi, il nous semblait difficile de comparer les deux travaux, car n'ayant pas les mêmes conditions expérimentales. Cependant, les résultats que nous avons obtenus lors de l'expérimentation 2, présentés en section 3.2.2.3, sont meilleurs que ceux obtenus par Langford et al. dans leur article malgré des conditions expérimentales plus complexes : nombre d'émetteurs plus important et signaux issus d'enregistrements.

3.2.3.2 Comparaison avec les autres travaux sur les réseaux siamois en RFF

Après la parution de notre article, d'autres auteurs ont publié des articles sur l'utilisation des réseaux siamois en RFF (ou dans des domaines/appellations similaires). Nous allons d'abord présenter les différents travaux, puis essayer de faire une comparaison avec les résultats que nous avons obtenus.

Dans [69], Hazra et al. ont utilisé un réseau siamois basé sur un réseau de neurones composé à la fois de couches convolutives (1D), de couches récurrentes et de couches denses pour du *one-shot learning*. Le paradigme d'apprentissage de leur réseau siamois utilise une fonction contrastive identique à celle que l'on a utilisée. Cependant, au lieu d'utiliser la distance L_2 comme dans l'article original [122], les auteurs ont choisi d'utiliser une norme L_1 à la place. Pour évaluer les performances de leur architecture, les auteurs ont créé deux jeux de données à l'aide de signaux provenant de 54 appareils sans-fils de 4 modèles différents et respectant la norme 802.15.4. Les auteurs ont notamment étudié les performances des réseaux siamois dans le cas de classes «non-vue», i.e. sur lesquels le réseau siamois n'avait jamais été entraîné. Ils ont aussi évalué le gain de performances dans le cas d'un réentraînement du réseau siamois avec des classes sur lequel le réseau siamois n'avait jamais été entraîné.

Dans [129], Wu et al. utilisent une approche partiellement basée sur les réseaux siamois. En effet, leur fonction de coût est composée de trois sous-fonctions de coût ayant des rôles

distincts : la classification, le partitionnement et la reconstruction (à la manière d'un auto-encodeur [31]). L'entraînement du réseau siamois a été réalisé à l'aide d'un jeu de données composé de 27 émetteurs Bluetooth (smartphones). Dans l'article, les auteurs ont ainsi montré les performances de leur architecture dans une configuration *close-set* (émetteurs connus) et *open-set* (émetteurs inconnus). De plus, ils ont aussi étudié l'importance de chacun des termes de leur fonction de coût.

Dans [66], Shen et al. présentent l'utilisation d'un réseau siamois pour du RFF sur des signaux LoRa. Plus particulièrement, ils utilisent un paradigme d'apprentissage proche de l'apprentissage par transfert contrastif, mais qui apprend l'espace d'enchâssement à l'aide d'une fonction de coût, dite *triplet loss*, proposée par Shroff et al. dans [118]. Cette fonction de coût ne considère plus des paires d'entrées, mais des triplets d'entrées : une entrée de référence, une entrée provenant de la même source, une entrée provenant d'une source différente. Contrairement à notre approche, les signaux d'entrée du réseau siamois ne sont pas les signaux I/Q, mais des spectrogrammes. Dans leurs expérimentations, les auteurs ont utilisé des signaux provenant de 60 émetteurs LoRa parmis 4 modèles. Tout d'abord, ils ont étudié les performances de leur architecture pour des appareils avec lesquels le réseau n'avait pas été entraîné, afin d'évaluer la scalabilité du modèle. De plus, ils ont aussi étudié l'effet de l'augmentation de données pour la robustesse de leur modèle.

La grande hétérogénéité de ces travaux rend la comparaison avec notre approche difficile. En effet, ces travaux n'ont pas les mêmes conditions expérimentales, ni même des conditions expérimentales comparables. Cependant, ces publications montrent un intérêt de la communauté quant à l'utilisation des réseaux siamois pour le RFF. De plus, les expérimentations réalisées dans ces articles peuvent être considérées comme complémentaires aux nôtres.

3.2.3.3 Scalabilité

Les approches classiquement utilisées pour l'apprentissage de représentations nécessitent un nombre d'exemples important. Par exemple, dans [28], l'architecture utilisée est entraînée à l'aide de 720 000 signaux I/Q, soit 180 000 par classe. À l'inverse, les réseaux siamois ont besoin de beaucoup moins d'exemples pour apprendre la représentation d'une nouvelle classe, puisqu'il suffit d'un unique exemple [69], [124]. De plus, comme nous l'avons indiqué lors de notre explication de la phase de décision, un nouvel émetteur peut être facilement rajouté à la base d'empreintes à l'aide de son signal ou de sa projection. Par contre, l'apprentissage de l'espace d'enchâssement nécessite beaucoup plus d'exemples. En

effet, dans notre cas, le jeu de données utilisé pour l'entraînement du réseau siamois est composé de 360 000 paires dont 180 000 paires positives et de 180 000 paires négatives.

3.2.3.4 Complexité

Les approches classiquement utilisées pour l'apprentissage de représentations nécessitent généralement une puissance de calcul importante, ainsi qu'un espace mémoire conséquent pour stocker les modèles. Par exemple, dans [28], leur architecture est composée de plus de 400 000 paramètres. Dans la littérature, certains articles traitent de l'implémentation d'architectures de réseaux de neurones légères comme dans [72] et [71] où les auteurs présentent des réseaux de neurones constitués d'environ 200 000 paramètres. Par rapport à ces réseaux de neurones légers, notre architecture légère possède notamment un nombre de paramètres plus conséquent ($\times 1.9$) et donc occupe un espace mémoire plus important. De plus, dans [72], les auteurs précisent que leur réseau de neurones effectue 533 212 opérations flottantes, soit environ 4.4 fois moins d'opérations par inférence que notre architecture légère.

3.3 Méthode 2 : *RF eigenfingerprints*

La seconde méthode de RFF proposée, appelée *RF eigenfingerprints*, est inspirée des travaux en reconnaissance faciale sur les *eigenfaces* [8], [9]. À l'inverse des réseaux siamois pour le RFF, cette méthode se focalise majoritairement sur les propriétés d'adaptabilité et de complexité, même si nous avons aussi exploré la scalabilité de cette méthode. De plus, nous avons également exploré les propriétés d'explicabilité des *RF eigenfingerprints*.

Dans la sous-section 3.3.1, nous présentons la méthodologie puis, dans la sous-section 3.3.2.2, différentes simulations/expérimentations et enfin, dans la sous-section 3.3.3, nous faisons une comparaison de notre méthode avec celles de la littérature.

Pour rappel, les *eigenfaces* [8], [9] sont issus des travaux de Turk et Pentland en reconnaissance faciale et inspirés de travaux antérieurs de Sirovich et Kirby [130], [131] soulignant l'intérêt de l'utilisation de la transformée de Karhunen-Loève pour l'apprentissage d'un espace de projection en reconnaissance faciale. Les *eigenfaces* consistent à apprendre une projection linéaire à l'aide d'une transformée de Karhunen-Loève, qui est une Analyse en Composantes Principales (ACP) centrée [31]. Cette projection est constituée d'un ensemble de vecteurs propres, appelés *eigenfaces*, dont un exemple [13] est présenté en figure

13. Cette figure a été obtenue à partir de la version étendue de la base de données Yale Face B [132],

3.17. De plus, chaque individu (classe) est représenté par une centroïde (ou vecteur centroïde) dans l'espace de projection. Ainsi, pour réaliser une authentification à l'aide des *eigenfaces*, l'image d'entrée est projetée dans l'espace formé par les *eigenfaces* apprises, puis la projection de l'image d'entrée est comparée aux centroïdes de chaque classe. Cette méthode est considérée comme l'une des premières méthodes d'apprentissage de représentations en reconnaissance faciale [134] et elle a inspiré de nombreuses autres méthodes en reconnaissance faciale. Par exemple, dans [135], Belhumeur et al. ont proposé les *fisherfaces* basés sur l'analyse discriminante de Fisher et dans [136], Yang et al. ont présenté les *kernel eigenfaces* utilisant l'astuce du noyau [31].

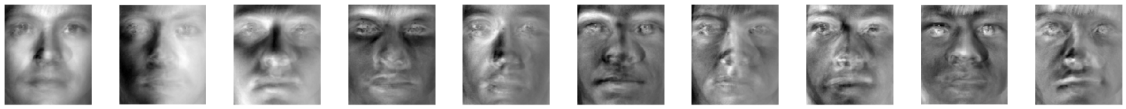


FIGURE 3.17 – *Eigenfaces* apprises sur le jeu de données Yale Face B

3.3.1 Méthodologie

La méthode des *RF eigenfingerprints* reprend le principe des *eigenfaces* mais pour l'authentification par empreinte radio. Pour cela, le préambule est utilisé pour apprendre les différents vecteurs propres ou descripteurs, appelés *RF eigenfingerprints*, qui correspondent aux signatures des émetteurs. L'utilisation du préambule permet d'avoir une «structure stable», i.e. identique à chaque émission, permettant d'éliminer en partie les problématiques d'explosion du nombre de vecteurs propres [137]. En effet, le signal radio d'un émetteur peut contenir une grande variabilité due aux données transmises qui ne sont pas forcément informatives pour le RFF. Ainsi, l'utilisation du préambule, qui est identique à chaque émission, permet de supprimer cette source de variabilité.

Dans cette partie, nous présentons les différentes étapes de la méthodologie nécessaires pour l'apprentissage et l'inférence. Il est important de noter que les étapes sont présentées dans le sens chronologique, par exemple l'apprentissage des vecteurs propres précède leur sélection. De plus, certaines étapes sont communes à la fois à l'apprentissage et à l'inférence, comme l'étape de pré-traitement ou de projection. La figure 3.18 résume les différentes étapes pour la phase d'entraînement et la phase d'inférence.

[133]

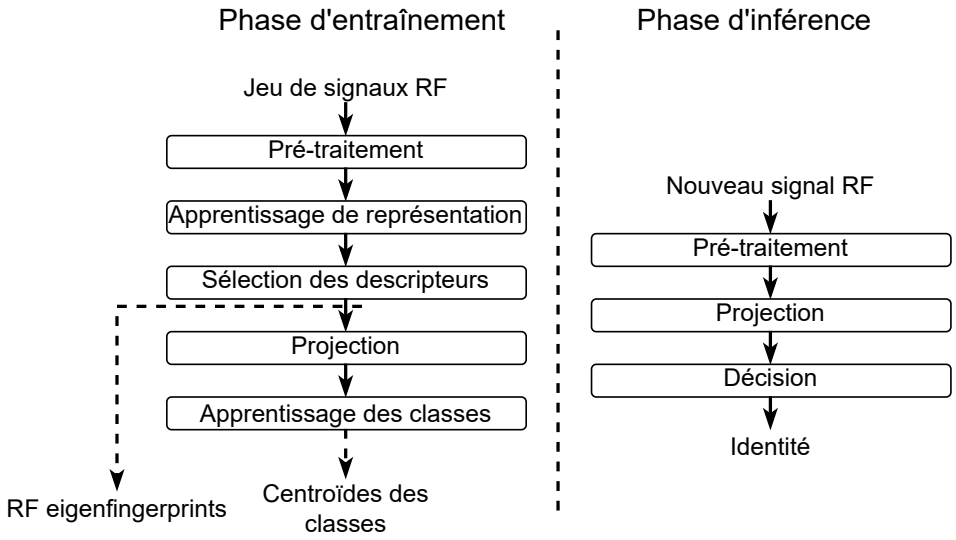


FIGURE 3.18 – Méthodologie des RF eigenfingerprints

3.3.1.1 Pré-traitement

La phase de pré-traitement est une étape primordiale permettant d'éliminer les sources de variations non informatives (ou *factors of variation* en anglais [107]). En effet, le signal émis $\tilde{s}(t)$ subit plusieurs modifications lors de la transmission, dont certaines ne sont pas informatives. La représentation en bande de base du signal reçu $\tilde{r}(t)$ a généralement la forme suivante :

$$\tilde{r}(t) = A\tilde{s}(t-\tau)e^{j2\pi\Delta ft} + \tilde{n}(t) \tag{3.7}$$

avec :

- $A \in \mathbf{C}$: l'amplitude complexe du signal.
- τ : le retard du signal.
- Δf : le décalage fréquentiel.
- $\tilde{n}(t) \sim \mathcal{CN}(0, \sigma^2)$: un bruit blanc gaussien complexe de puissance σ^2.

Nous proposons ainsi deux pré-traitements permettant de supprimer ces sources de variabilité non informatives :

- Pré-traitement n°1 : ce pré-traitement, présenté en figure 3.19, corrige le décalage fréquentiel, le décalage temporel, ainsi que l'amplitude complexe.
- Pré-traitement n°2 : ce pré-traitement, présenté en figure 3.20, corrige le décalage temporel, ainsi que l'amplitude complexe.

Pré-traitement n°1 La première étape de ce pré-traitement consiste à corriger le décalage fréquentiel pour resynchroniser le signal en fréquence. L'estimation du décalage fréquentiel $\hat{\Delta}f$ peut s'effectuer de différentes façons selon le type de modulation. Par exemple à l'aide de la séquence de conditionnement [56], [57], ou pour les modulations M-PSK grâce à la récupération par élévation à la puissance M présentée dans la section 2.1.3.3 [84]. Le signal synchronisé $\tilde{s}_1(t) = \tilde{r}(t)e^{-j2\pi\hat{\Delta}ft}$ résultant de cette étape de synchronisation a ainsi la forme suivante $\tilde{s}_1(t) = \tilde{s}(t-\tau) + \tilde{n}_2(t)$ avec $\tilde{n}_2(t)$ un bruit ayant les mêmes propriétés que $\tilde{n}(t)$.

Pour la seconde étape, il suffit de faire une intercorrélation $R_{\tilde{s}_1\tilde{p}}(\tau)$ entre le signal synchronisé $\tilde{s}_1(t)$ et le préambule $\tilde{p}(t)$ pour estimer le retard $\hat{\tau} = arg\,max_\tau R_{\tilde{s}_1\tilde{p}}(\tau)$, ainsi que l'amplitude complexe $\hat{A} = max_\tau R_{\tilde{s}_1\tilde{p}}(\tau)$.

Enfin, il suffit de corriger le décalage temporel à l'aide de $\hat{\tau}$ et de réaliser une correction d'amplitude à l'aide de $\hat{A}$. Il est important de noter que le bruit résultant $\tilde{n}'(t)$ est un bruit blanc gaussien complexe avec une variance $\sigma_n^2 = \frac{\sigma^2}{\hat{A}^2}$.

Ce pré-traitement correspond au type de pré-traitements généralement utilisé dans une chaîne de réception RF et il peut ainsi être mutualisé sans coût d'implémentation supplémentaire. De plus, des pré-traitements similaires ont déjà été utilisés dans le domaine du RFF [71].

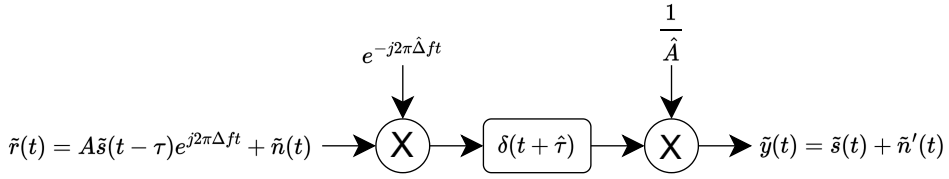


FIGURE 3.19 – Pré-traitement n° 1 pour les *RF eigenfingerprints*

Pré-traitement n°2 Ce pré-traitement ne corrige pas le décalage fréquentiel Δf, car certains auteurs le considèrent comme constant au cours du temps et, donc, signant [1], [13], [54]. Cependant, l'estimation du décalage fréquentiel $\hat{\Delta}f$ est quand même réalisée. En effet, l'intercorrélation du signal reçu et du préambule est sensible au décalage en fréquence et peut totalement échouer en présence d'un décalage fréquentiel trop important. Il faut ainsi réaliser l'intercorrélation $R_{\tilde{r}\tilde{s}_f}(\tau)$ du signal reçu et du préambule décalé en fréquence $\tilde{s}_f(t) = \tilde{p}(t)e^{j2\pi\hat{\Delta}ft}$, afin d'estimer le retard $\hat{\tau} = arg\,max_\tau R_{\tilde{r}\tilde{s}_f}(\tau)$, ainsi que l'amplitude complexe $\hat{A} = max_\tau R_{\tilde{r}\tilde{s}_f}(\tau)$. De la même manière que pour le pré-traitement n°1, le bruit $\tilde{n}'(t)$ résultant est un bruit blanc gaussien complexe avec une variance $\sigma_n^2 = \frac{\sigma^2}{\hat{A}^2}$.

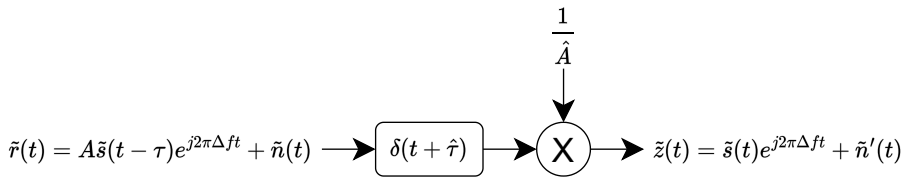


FIGURE 3.20 – Pré-traitement n°2 pour les *RF eigenfingerprints*

Autres pré-traitements Il est aussi possible de réaliser des procédures de pré-traitement prenant en compte une phase d'égalisation permettant de corriger l'effet du canal sur le signal reçu. En effet, pour les méthodes de RFF *radiometric*, le canal est une autre source de variabilités non-informatives qu'il est nécessaire de corriger. Par exemple, dans [54], Shanke et al. proposent un pré-traitement permettant de supprimer l'effet du canal à l'aide d'une phase d'égalisation.

3.3.1.2 Apprentissage des *RF eigenfingerprints*

L'apprentissage des descripteurs peut se faire à l'aide d'une analyse en composantes principales comme pour les *eigenfaces*. Cependant, nous avons choisi d'utiliser une décomposition en valeurs singulières, comme dans l'exemple de Brunton sur les *eigenfaces* [137]. En effet, la décomposition en valeurs singulières a comme particularité d'ordonner de manière décroissante les valeurs propres, ainsi que les vecteurs propres correspondants [137]. De plus, il s'agit d'une des manières les plus courantes de calculer une analyse en composantes principales [86], [107], [137].

La phase d'apprentissage des vecteurs propres nécessite une matrice de données d'apprentissage $X_{train} \in \mathbb{C}^{N \times L}$ où N correspond au nombre d'échantillons de chaque signal et L au nombre de signaux du jeu d'entraînement. Les différents signaux pré-traités sont stockés en colonne dans la matrice de données d'apprentissage X_{train}. Cette matrice est ainsi composée de nombres complexes, où x_{ij} représente le i-ème échantillon du j-ème signal.

Les différentes étapes de l'apprentissage des descripteurs sont les suivantes :

1. Calcul du vecteur moyen $\bar{x} = \begin{bmatrix} \bar{x}_1 \\ \vdots \\ \bar{x}_N \end{bmatrix}$ avec $\bar{x}_i = \sum_{j=1}^{L} x_{ij}$.
2. Calcul de la matrice de données d'apprentissage centrée $B_{train} = X_{train} - \bar{x}[1 \cdots 1]$.

3. Décomposition en valeurs singulières [137] de $B_{train} = U\Sigma V^H$.
4. Calcul de la matrice des valeurs propres $D = \frac{1}{L-1}\Sigma\Sigma^T$.

Après ces étapes, le vecteur moyen $\bar{x}$, la matrice des vecteurs propres U, ainsi que la matrice des valeurs propres D ont été apprises. Les matrices B_{train}, Σ et V ne sont que des résultats intermédiaires du calcul. De plus, le calcul de la matrice D est nécessaire seulement si l'on veut réutiliser les valeurs propres par la suite.

Deux configurations pour l'apprentissage des vecteurs propres sont donc possibles pour l'implémentation finale dans un réseau IoT :

— La première possibilité consiste à apprendre les vecteurs propres sur un jeu de données généraliste (constitué de nombreux émetteurs) et de les réutiliser lors de l'implémentation finale de l'algorithme dans un réseau IoT. Cette configuration est notamment conseillée pour l'apprentissage de nouvelles classes [9]. Par conséquent, la décomposition en valeurs singulières devra s'effectuer hors-ligne sur des machines de calculs puissantes à cause de la taille du jeu de données.

— La seconde possibilité consiste à apprendre les vecteurs propres à partir des exemples des émetteurs que l'on cherche à authentifier. Cette configuration nécessitera potentiellement moins de vecteurs propres que la configuration précédente, mais elle aura peut-être une capacité de généralisation plus faible [9]. Par conséquent, la décomposition en valeurs singulières pourra s'effectuer en-ligne sur des machines plus restreintes en capacité de calcul.

3.3.1.3 Sélection des *RF eigenfingerprints*

Nous venons de voir comment apprendre les différents vecteurs propres à l'aide d'une décomposition en valeurs singulières. Maintenant, nous allons voir comment sélectionner les plus importants pour constituer l'espace de projection constitué des *RF eigenfingerprints*. Dans la littérature, il existe de nombreuses techniques de sélection de vecteurs propres, dont les méthodes heuristiques proposées dans [31] ou les méthodes optimales présentées dans [137]. Dans la suite de ce paragraphe, nous allons décrire une nouvelle méthode de sélection des vecteurs propres basée sur le test Q de Ljung-Box.

Le test d'hypothèse Q de Ljung-Box est classiquement utilisé en statistique pour tester l'autocorrélation d'ordre supérieur à 0 [138]. Il est notamment employé en séries temporelles pour calibrer et valider les paramètres d'un modèle *AutoRegressive Integrated Moving*

Average (ARIMA) à l'aide de ses résidus. Ce test possède deux hypothèses [14] :

— H_0 : l'hypothèse nulle stipule que l'autocorrélation d'ordre 1 à R est nulle.
— H_1 : l'hypothèse alternative stipule que l'autocorrélation d'ordre 1 à R est non nulle.

L'intuition derrière l'utilisation de ce test pour la sélection des *RF eigenfingerprints* est la suivante : on cherche à déterminer les vecteurs propres correspondant aux défauts de l'émetteur pour former l'espace de projection. Notre méthode de sélection est basée sur l'hypothèse que les vecteurs propres vont former un sous-espace signal et un sous-espace bruit, de la même manière que les méthodes d'estimation spectrale comme MUSIC ou ESPRIT [102], [139]-[141]. Ainsi, les vecteurs du sous-espace bruit auront une autocorrélation similaire à celle d'un bruit blanc, à l'inverse des vecteurs propres du sous-espace bruit correspondant aux défauts de l'émetteur. De plus, les vecteurs propres du sous-espace signal sont rattachés aux valeurs singulières (ou propres) les plus importantes, comme pour les méthodes d'estimation spectrale évoquées précédemment.

Ainsi, pour la sélection des *RF eigenfingerprints*, nous proposons d'utiliser l'algorithme 2 ci-dessous, permettant de sélectionner le nombre de vecteurs propres à conserver. Son principe consiste à vérifier itérativement que les vecteurs propres, préalablement triés selon les valeurs propres décroissantes, ont une autocorrélation différente de celle d'un bruit blanc. Dès que l'algorithme évalue un vecteur propre qui a une autocorrélation similaire à un bruit blanc, il s'arrête et retourne le nombre de vecteurs propres sélectionnés. Pour cela, il fait appel au test d'hypothèse de Ljung-Box : *LjungBoxQTest*. L'ensemble des K vecteurs propres sélectionnés par l'algorithme de sélection, appelés *RF eigenfingerprints*, forment l'espace de projection $U_{proj} = u_{i \in [\![1;K]\!]}$.

Cette méthode de sélection de descripteurs est fortement influencée par le modèle de bruit. En effet, si le bruit additif n'est pas un bruit blanc, le nombre de descripteurs retourné sera potentiellement trop grand. Ainsi, une astuce consiste à ajouter du bruit blanc (gaussien) additif aux signaux enregistrés afin de remédier à ce problème. Cependant, cela va diminuer le rapport signal à bruit et donc potentiellement dégrader les performances de l'algorithme lors de la phase d'inférence, car l'estimation des centroïdes de chaque classe seront plus bruitées.

14. Nous avons réalisé le test grâce à la fonction 'lbqtest' de Matlab qui a comme valeur d'ordre $R = min(20, N-1)$

Algorithm 2: Sélection des vecteurs propres

Data: $U = \{u_1, \cdots, u_N\}$: la matrice des vecteurs propres
Result: $U_{proj} = [u_1 \cdots u_K]$: la matrice de projection
$K \leftarrow 1$;
$H \leftarrow 1$;
while $H \neq 0$ **do**
  $H \leftarrow LjungBoxQTest(u_K)$;
  $K \leftarrow K + 1$;
end
$K \leftarrow K - 1$;

3.3.1.4 Décision

La procédure de décision projette d'abord le signal d'entrée pré-traité x dans l'espace de projection à l'aide des *RF eigenfingerprints* et compare le signal projeté aux centroïdes de chaque classe. Comme pour les réseaux siamois, nous ferons une distinction entre la vérification et l'identification d'un signal. De plus, nous présentons une modélisation statistique du signal projeté, ainsi que la procédure pour l'apprentissage des centroïdes de chaque classe.

Projection La seconde étape de la décision consiste à projeter le signal pré-traité $x \in \mathbb{C}^N$ dans l'espace de projection pour obtenir $z \in \mathbb{C}^K$ de la manière suivante :

$$z = U_{proj}^H (x - \bar{x}) \tag{3.8}$$

À noter que l'obtention de la matrice des données projetées $Z \in \mathbb{C}^{K \times L}$ à partir d'une matrice de données X peut être décrite de la manière suivante :

$$Z = U_{proj}^H (X - \bar{x}[1 \cdots 1]) \tag{3.9}$$

Le tableau 3.10 récapitule la complexité de la projection en termes d'occupation mémoire et de coût de calcul. En effet, la complexité de calculs est liée au produit matriciel de $x - \bar{x}$ avec U_{proj}. De plus, la complexité mémoire est liée au nombre d'éléments de la matrice U_{proj}.

Modélisation statistique du signal projeté Pour l'étape de décision, il est ainsi nécessaire de modéliser le signal projeté z. Ainsi, nous considérons le signal $\tilde{s}(t)$ émis par

Type	Calculs	Mémoire
Projection	$O(N \times K)$	$O(N \times K)$

TABLE 3.10 – Complexité de la phase de projection des *RF eigenfingerprints*

l'émetteur c, noté $x_c \in \mathbb{C}^N$, comme constant, c'est-à-dire que son empreinte radio est identique à chaque émission. De plus, la projection de $x_c \in \mathbb{C}^N$ sera notée $\mu_c = U_{proj}^H(x_c - \bar{x})$. Enfin, comme indiqué dans [76], un bruit blanc gaussien complexe $n'(t) \sim \mathcal{CN}(0, \sigma_n^2)$ projeté dans une base orthonormée est aussi un bruit blanc gaussien complexe $n''(t) \sim \mathcal{CN}(0, \sigma_n^2)$. À l'aide de cette hypothèse et de cette propriété, la modélisation probabiliste des données appartenant à la classe c suivra une loi normale complexe multivariée [142], [143] :

$$f(z, \Theta_c) = \frac{1}{|\Sigma_c|\pi^K} e^{-(z-\mu_c)^H \Sigma_c^{-1}(z-\mu_c)} \tag{3.10}$$

avec :

- $z \in \mathbb{C}^K$: le signal projeté dans le sous-espace $\{u_1 \cdots u_K\}$.
- μ_c : le vecteur moyen de la classe c.
- $\Sigma_c = \sigma_c^2 I$: la matrice de variance-covariance de la classe c, avec σ_c^2 le bruit normalisée de la classe c [15].
- $\Theta_c = \{\mu_c, \Sigma_c\}$: l'ensemble des paramètres désignant une classe.

De plus, il est possible d'avoir un modèle probabiliste de l'ensemble des C classes, en supposant que la probabilité de classe est identique : équiprobabilité. Ainsi, la modélisation probabiliste des différentes classes est réalisé à l'aide d'un modèle de mélange gaussien :

$$g(z, \Theta) = \sum_{k=1}^{C} p_k f(z, \Theta_k) \tag{3.11}$$

avec :

- $\Theta = \{\Theta_1, \cdots, \Theta_C\}$: l'ensemble des paramètres de chaque classe.
- $p_k = \mathbb{P}(y = k)$: la probabilité d'appartenance à la classe k (ici $\frac{1}{C}$).

Apprentissage de la représentation des classes Une fois que les données sont projetées dans l'espace de projection U_{proj}, il est nécessaire d'apprendre la représentation de chaque classe dans l'espace de projection. Comme nous venons de le présenter, chaque

15. Il est possible de considérer que chaque classe un rapport signal-à-bruit différent. Par exemple, dans le cas d'une propagation directe où les émetteurs ne sont pas à la même distance du récepteur.

classe est définie par le vecteur μ_c. Ainsi, l'estimateur $\hat{\mu}_c$ du vecteur μ_c est le suivant :

$$\hat{\mu}_c = \frac{1}{\sum_{i=1}^{L} \delta(y_i, c)} \sum_{i=1}^{L} \delta(y_i, c) z_i \tag{3.12}$$

avec :

— z_i : la projection de x_i dans le sous-espace.

— y_i : la classe de x_i.

— δ : la fonction Dirac [86].

Chaque axe étant indépendant aux autres axes à cause de la projection [76], cela rend l'apprentissage du centroïde de la classe $\hat{\mu}_c$ moins sensible au fléau de la dimension [144]. En effet, en apprentissage automatique, la taille du jeu d'entraînement croit de manière exponentielle en fonction du nombre de descripteurs [31]. Cependant, l'orthogonalité entre les différentes dimensions permet d'apprendre indépendamment chaque composante de $\hat{\mu}_c$. De plus, en supposant l'hypothèse d'homoscédasticité : $\forall c \in [\![1; C]\!], \Sigma_c = \sigma_n^2 I$, il est possible de déterminer le nombre minimum d'exemples pour estimer le centroïde en connaissant le rapport signal à bruit et donc σ_n^2. Ainsi, nous donnons la formule permettant de déterminer le nombre minimum d'exemples L_c à partir d'un intervalle de confiance à $1 - \alpha$ (voir l'annexe B de [145]) :

$$L_c = \frac{t_{1-\alpha}\sigma_n^2}{2I_c} \tag{3.13}$$

avec

— $t_{1-\alpha}$: le seuil correspondant à $\mathbb{P}(X < t_{1-\alpha}) = 1 - \alpha$ (avec $X \sim \chi^2(2)$).

— I_c : la taille de l'intervalle de confiance.

L'intervalle de confiance peut être défini comme une fraction de la variance inter-classe $d = \lambda_{min} - \sigma_n^2$, où λ_{min} est la plus petite valeur propre de l'espace de projection. En effet, la valeur λ_i représente la variance totale dans la dimension i de l'espace de projection et σ_n^2 représente la variance intra-classe dans cette dimension. Finalement, définir l'intervalle de confiance comme une fraction de d permet d'éviter que les estimations des centroïdes des différentes classes ne se chevauchent trop dans la dimension correspondant à la plus petite valeur propre λ_{min}, c'est-à-dire la dimension où la variance inter-classe est la plus faible.

Vérification Comme nous l'avons précédemment introduit, la vérification consiste à comparer que le signal pré-traité x correspond à l'empreinte d'un émetteur particulier. Dans le cas des *RF eigenfingerprints*, il est ainsi possible d'effectuer une vérification de la

manière suivante :

$$\delta_1(\|z - \hat{\mu}_c\|^2 < T_c) \tag{3.14}$$

avec δ_1 : la fonction indicatrice [86].

Le seuil de classe pouvant être décrit de la manière suivante [145] :

$$T_c = \frac{t_{1-\alpha}\sigma_c^2}{2}\frac{L_c + 1}{L_c} \tag{3.15}$$

avec

— $t_{1-\alpha}$: le seuil correspondant à $\mathbb{P}(X < t_{1-\alpha}) = 1 - \alpha$ avec $X \sim \chi^2(2K)$).

— σ_c^2 : le bruit normalisé de la classe c.

Le tableau 3.11 récapitule la vérification en termes de mémoire et de calculs. La complexité de calcul de la phase de vérification est liée au calcul de la distance $\|z - \hat{\mu}_c\|^2$. De plus, la complexité mémoire est liée au stockage des empreintes $\hat{\mu}_c$ de chaque émetteur.

Type	Calculs	Mémoire
Classification	$O(K)$	$O(K \times C)$

TABLE 3.11 – Complexité de la phase de vérification des *RF eigenfingerprints*

Identification L'identification d'un émetteur à partir de z est similaire à la procédure décrite dans la section 3.2.1.3. Pour rappel, cette procédure consistait à réaliser une vérification pour chaque empreinte $\hat{\mu}_c$.

Le tableau 3.12 récapitule la complexité de l'identification en termes de mémoire et de calculs. En effet, pour l'identification, il est nécessaire de réaliser C vérifications, donc la complexité de calcul associée est $O(K \times C)$. De plus, tout comme pour la vérification, il est nécessaire de stocker l'ensemble des centroïdes $\hat{\mu}_c$.

Type	Calculs	Mémoire
Identification	$O(K \times C)$	$O(K \times C)$

TABLE 3.12 – Complexité de la phase d'identification des *RF eigenfingerprints*

Dans ce paragraphe, nous présentons une seconde manière de réaliser l'identification d'un émetteur. Cette procédure est constituée d'une phase de détection d'anomalies, suivie d'une phase de classification. Comme nous l'avions évoqué dans l'état de l'art, la procédure que nous présentons est assez courante en RFF, notamment dans [33]. Cette seconde

possibilité sera réutilisée dans nos expérimentations, notamment la phase de classification, pour nous permettre de comparer plus facilement nos résultats à ceux de la littérature.

Détection d'anomalies Il est possible d'effectuer une détection d'anomalies de la manière suivante [31], [33] :

$$\delta_1(g(z, \Theta) < T) \tag{3.16}$$

avec T : le seuil de tolérance des anomalies, permettant de déterminer si le signal provient d'un émetteur légitime ou illégitime. Il peut notamment être obtenu par l'approche présentée dans [31].

Classification Si la donnée z est considérée comme faisant partie d'une classe préalablement observée et donc provenant d'un émetteur légitime, la classification peut se faire grâce à une décision par maximum *a posteriori*[16] [86] :

$$y = arg\ max_c\ \mathbb{P}(y = c|z) \tag{3.17}$$

$$= arg\ max_c\ \frac{\mathbb{P}(y = c)\mathbb{P}(x|y = c)}{\mathbb{P}(z)} \tag{3.18}$$

$$= arg\ max_c\ \mathbb{P}(y = c)\mathbb{P}(x|y = c) \tag{3.19}$$

Or, comme nous l'avons annoncé précédemment, on suppose l'équipropabilité de chaque classe ($\forall c \in [\![1; C]\!], \mathbb{P}(y = c) = \frac{1}{C}$). De plus, en supposant l'hypothèse d'homoscédasticité ($\forall c \in [\![1; C]\!], \Sigma_c = \sigma_n^2 I$), la phase de classification peut être considérée comme une décision par maximum de vraisemblance :

$$y = arg\ max_c\ \mathbb{P}(x|y = c) \tag{3.20}$$

$$= arg\ min_c\ \|z - \mu_c\|^2 \tag{3.21}$$

La complexité de la phase de classification est présentée dans le tableau 3.13. En effet, il est toujours nécessaire de calculer les distances $\|z - \mu_c\|^2$, entre le signal projeté z et les différentes centroïdes μ_c. De plus, il est toujours nécessaire de stocker les empreintes μ_c de chaque émetteur. Enfin, la phase de détection d'anomalies a une complexité similaire, car dépendant majoritairement du calcul des différentes distances $\|z - \mu_c\|^2$.

16. La probabilité est généralement remplacée par la fonction de vraisemblance.

Type	Calculs	Mémoire
Identification	$O(K \times C)$	$O(K \times C)$

TABLE 3.13 – Complexité de la phase de classification des *RF eigenfingerprints*

3.3.2 Simulation et expérimentations

Dans cette sous-section, nous allons d'abord présenter une simulation de notre méthode de RFF avec le pré-traitement n°1. Elle a été réalisée à l'aide d'un modèle d'imperfections inspiré du modèle d'imperfections que nous avons introduit dans la section 2.1.2. De plus, nous avons réalisé une première expérimentation avec des signaux réels, nous permettant de mettre en lumière les performances de la méthode des *RF eigenfingerprints* pour les différents pré-traitements introduits. Enfin, nous présentons une seconde expérimentation évaluant l'impact du nombre de données d'apprentissage sur les performances de notre méthode.

3.3.2.1 Simulation avec le modèle d'imperfections

La première simulation [17] que nous avons réalisée est basée sur le modèle d'imperfections présenté en figure 3.21. Ce modèle d'imperfections est notamment inspiré du modèle d'imperfections présenté dans la sous-section 2.1.2. L'utilisation de ce modèle suppose que le signal reçu a été préalablement traité par le schéma de pré-traitement n°1 et présenté en figure 3.19, i.e le décalage fréquentiel, le retard et l'amplitude ont été compensés.

Les paramètres du modèle d'imperfections sont les suivants :

- Décalage I/Q :
  - $A_I \sim U(-0.01, 0.01)$
  - $A_Q \sim U(-0.01, 0.01)$
- Déséquilibre I/Q :
  - $\epsilon \sim U(-0.01, 0.01)$
  - $U \sim U(\frac{-\pi}{32}, \frac{\pi}{32})$
- Amplificateur de puissance (AM/AM) [18] :
  - $K = 3$
  - $a_1 = 1$
  - $a_2 \sim -U([-27dB, -33dB])$

17. Cette simulation servira de référence pour montrer l'influence de différents défauts sur les paramètres appris.

18. Les paramètres a_2 et a_3 sont négatifs et créent une compression de la forme d'onde.

— $a_3 \sim -U([-45dB, -55dB])$

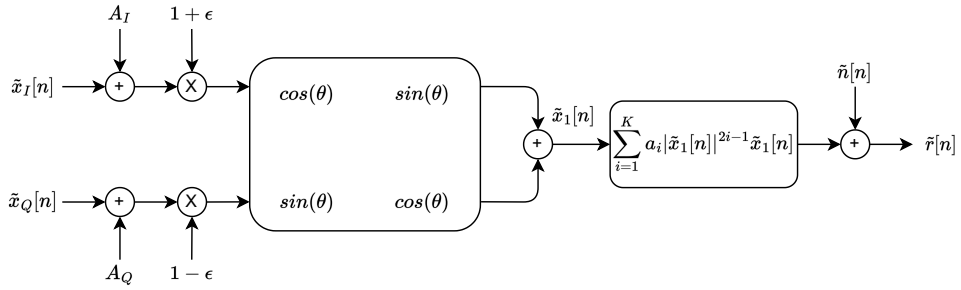


FIGURE 3.21 – Modèle d'imperfections pour la simulation des *RF eigenfingerprints*

Nous avons ainsi créé 30 signaux par classe avec 10 classes. Le préambule est composé de 50 symboles [19] QPSK générés aléatoirement avec un facteur de sur-échantillonnage de 5, soit un total de 250 échantillons. Tous les signaux d'une même classe ont les mêmes valeurs d'imperfections, seule la réalisation du bruit change d'un signal à l'autre. Le bruit $\tilde{n}[n]$ est un bruit additif blanc gaussien complexe correspondant à une valeur de SNR de 30 dB, permettant de visualiser plus facilement les *RF eigenfingerprints* appris. De plus, comme tous les signaux ont le même SNR, l'hypothèse d'homoscédascticité est respectée.

Une fois la matrice d'apprentissage de données $X_{train} \in \mathbb{C}^{(250\times300)}$ obtenue, nous avons réalisé l'apprentissage des *RF eigenfingerprints* à l'aide de la méthodologie précédemment présentée en figure 3.18. Concernant la sélection, les valeurs propres (ordonnées de manière décroissante) sont présentées en figure 3.22. Nous pouvons constater que les trois premières valeurs propres se détachent des autres. En effet, nous pouvons constater qu'à partir de la quatrième valeur propre, la décroissance des valeurs propres diminue. Ce phénomène est appelé coude ou épaulement dans la littérature de l'apprentissage automatique [86], [144]. De plus, la figure 3.23 montre les *p-values* pour la réalisation du test Q de Ljung-Box sur chaque vecteur propre. Nous pouvons observer que les 3 premiers vecteurs propres ont une *p-value* inférieure au seuil de 5% (0.05), à l'inverse des autres vecteurs propres. Donc, pour les 3 premiers vecteurs propres, l'hypothèse nulle est rejetée et l'hypothèse alternative est favorisée. Ainsi, l'algorithme 2 considère les 3 premiers vecteurs propres comme étant informatifs.

Pour la visualisation des descripteurs appris par le classifieur, nous avons utilisé une règle provenant de la théorie des probabilités : la règle des 68-95-99.7. Cette règle est la suivante, pour une variable gaussienne $X \sim \mathcal{N}(\mu, \sigma^2)$ on a :

19. En utilisant la fonction Matlab rcosdesign(0.5, 4, 5).

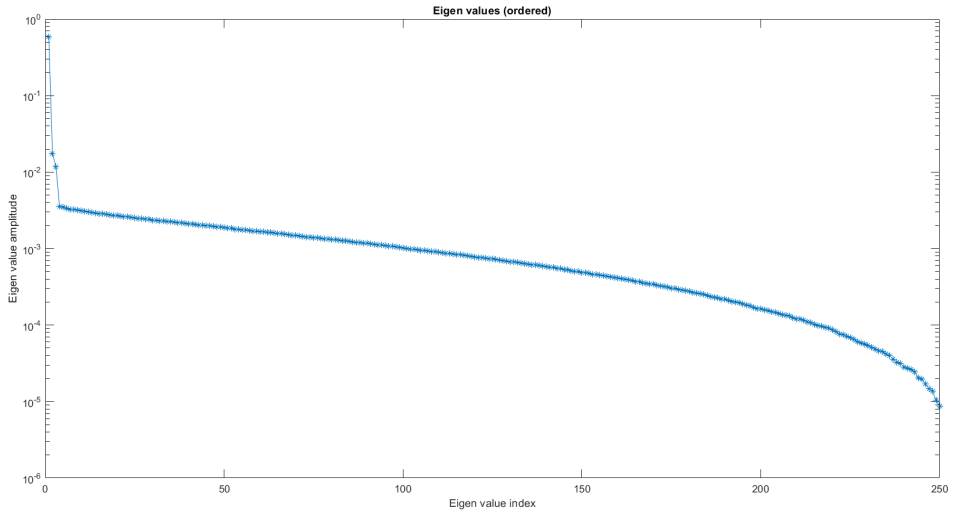


FIGURE 3.22 – Valeurs propres des *RF eigenfingerprints*

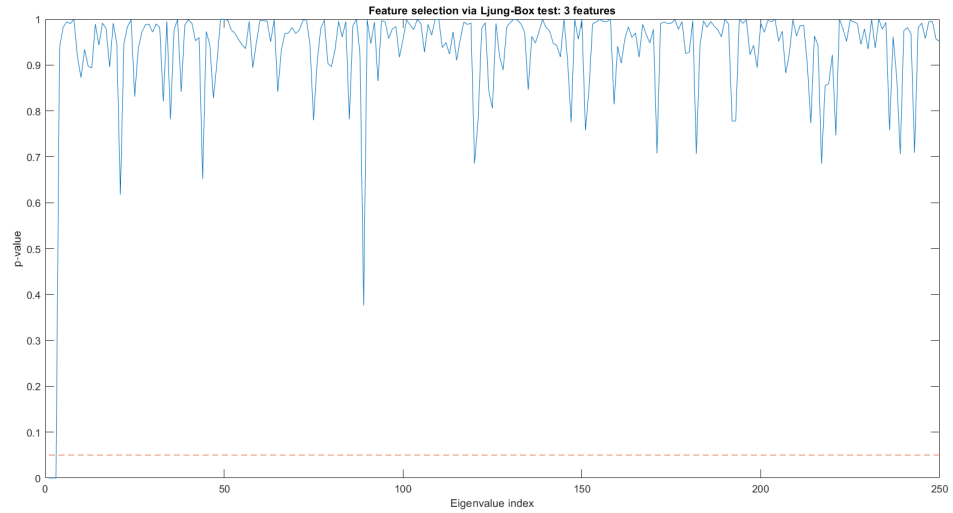


FIGURE 3.23 – *p-values* des *RF eigenfingerprints*

— $\mathbb{P}(\mu - \sigma < X < \mu + \sigma) \approx 0.6827$
— $\mathbb{P}(\mu - 2\sigma < X < \mu + 2\sigma) \approx 0.9545$
— $\mathbb{P}(\mu - 3\sigma < X < \mu + 3\sigma) \approx 0.9973$

Ainsi, dans notre cas, on va utiliser une version dérivée de cette règle [20] pour visualiser

20. Cette technique de visualisation avait déjà été utilisée dans la vidéo «Eigenfaces Example - Udacity» (youtube.com). Dans notre cas, on prendra 3 σ considérant le plus de variation, soit 99.7%.

l'impact de chaque *RF eigenfingerprint* u_k sur le signal moyen $\bar{x}$, de la manière suivante :

- $\bar{x}$ (en noir) : le signal moyen.
- u_k (en bleu) : le descripteur appris.
- $\bar{x} - 3\sqrt{\lambda_k}u_k$ (en bleu) : l'effet du vecteur propre u_k (en négatif) pondéré par 3 fois l'écart-type de cet axe sur le vecteur moyen $\bar{x}$.
- $\bar{x} + 3\sqrt{\lambda_k}u_k$ (en bleu) : l'effet du vecteur propre u_k (en positif) pondéré par 3 fois l'écart-type de cet axe sur le vecteur moyen $\bar{x}$.

Premièrement, nous pouvons constater sur la figure 3.24 que le signal moyen $\bar{x}$ est assez similaire au préambule. Cependant, il s'agit d'une version du préambule dont l'amplitude a subi un phénomène de compression AM/AM à cause de l'amplificateur de puissance.

Pour cette simulation, les *RF eigenfingerprints* sont visualisées en figure 3.25 et peuvent être interprétées de la manière suivante :

- u_1 : ce vecteur propre correspond au déséquilibre I/Q. En effet, nous pouvons observer une distorsion qui étire ou compresse la forme d'onde selon une diagonale.
- u_2 : ce vecteur propre correspond aux non-linéarités de l'amplificateur de puissance, ainsi qu'au décalage I/Q. En effet, nous pouvons observer que la forme d'onde est étirée ou compressée selon son amplitude. De plus, elle subit un décalage par rapport à l'origine.
- u_3 : ce vecteur propre correspond au décalage I/Q. En effet, la forme d'onde subit un décalage par rapport à l'origine.

Nous pouvons constater que les descripteurs appris (les *RF eigenfingerprints*) correspondent à des descripteurs interprétables comme pour les *eigenfaces*, ainsi qu'à des défauts relativement différents les un des autres. En effet, nous pouvons constater que des défauts provenant d'un modèle non-linéaire, celui de la figure 3.21, peuvent être appris sous la forme d'une combinaison linéaire de vecteurs propres représentant des défauts relativement différents les un des autres.

La figure 3.26 montre la projection des données dans le sous-espace composé des 3 vecteurs propres sélectionnés et visualisés en figure 3.25. Nous pouvons constater que les différentes classes sont relativement bien séparées dans le sous-espace de projection. Nous pouvons aussi remarquer que la variance inter-classe dépend de la valeur propre comme expliqué dans [145] et que la variance inter-classe est identique sur tous les axes, car dépendant de la puissance du bruit [145].

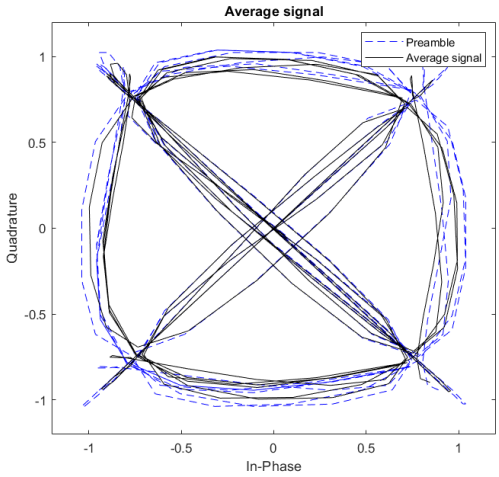


FIGURE 3.24 – Signal moyen appris pour la méthode des *RF eigenfingerprints*

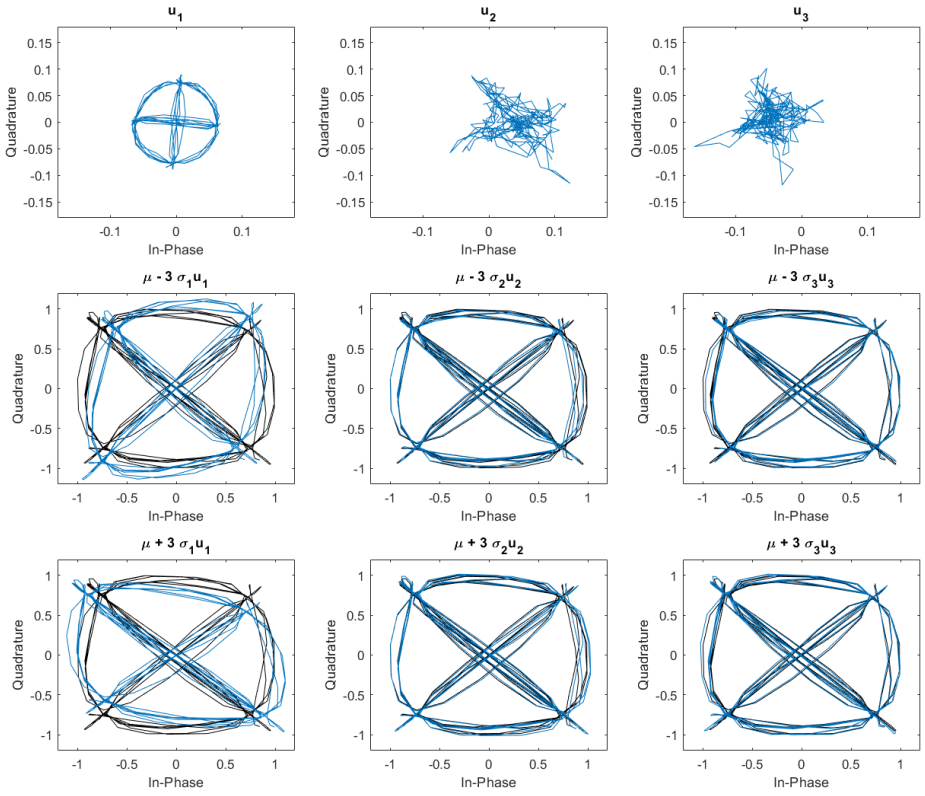


FIGURE 3.25 – Visualisation des *RF eigenfingerprints* appris

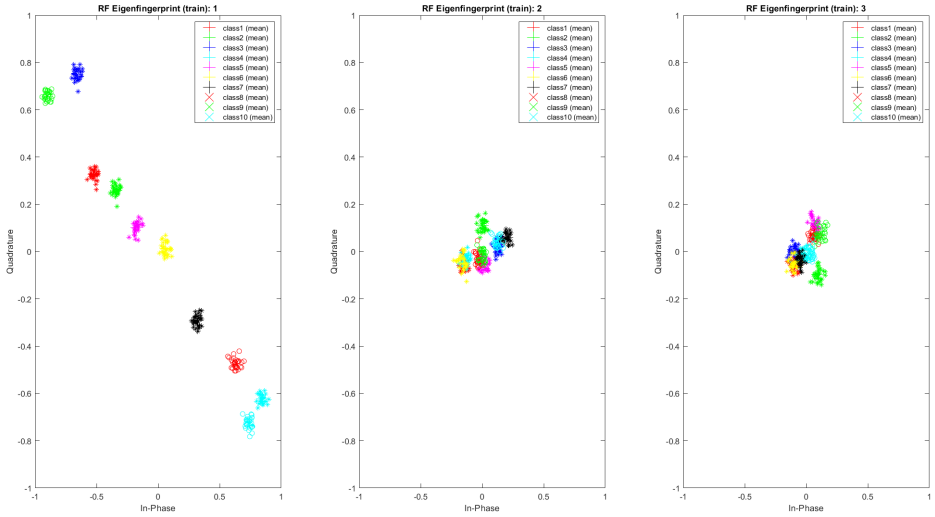


FIGURE 3.26 – Projection dans le sous-espace appris des *RF eigenfingerprints*

3.3.2.2 Expérimentation 1 : Évaluation des performances avec des données réelles

Nous avons réalisé une expérimentation avec des données réelles pour confirmer les performances de la méthode des *RF eigenfingerprints*. Plus particulièrement, nous allons comparer les performances de notre méthode avec les différents types de pré-traitements proposés, en utilisant comme métrique la justesse (*accuracy*). Cela nous permettra, par la suite, de plus facilement comparer nos résultats à ceux de la littérature, notamment aux travaux présentés dans [28], [72].

Pour cela, nous avons utilisé 4 plateformes SDR Adalm-PLUTO de Analog Devices comme émetteurs à identifier et un enregistreur de signaux I/Q BB60C de SignalHound comme récepteur [21]. Chaque émetteur émet le même préambule en boucle (avec 250 échantillons nuls entre chaque émission de préambule). Le récepteur enregistre 50 préambules (5 ms) par émetteur à une fréquence d'échantillonnage $F_e = 5$ MHz et une fréquence centrale $f_0 = 2.45$ GHz. Il s'agit du même préambule que celui qui a été utilisé dans la simulation, 50 symboles QPSK avec 250 échantillons.

L'enregistrement des signaux provenant d'une plateforme SDR est réalisé indépendamment des enregistrements de signaux des autres plateformes. Pour cela, les plateformes ADALM Pluto ont été positionnées à une distance de 45 cm de l'enregistreur de signaux

21. Les antennes utilisées sont des VERT2450 de chez Ettus Research

BB60C. La figure 3.27 présente la configuration de notre banc d'essai. De plus, nous avons fixé le rapport signal à bruit (SNR) à 30 dB pour chaque préambule reçu en ajoutant du bruit blanc gaussien complexe pour arriver à la valeur de SNR désiré SNR_{des}. La technique utilisée pour obtenir un SNR en particulier est similaire à celle utilisée dans [34], mais la puissance du bruit désirée $P_{n'}$ est calculée de la manière suivante :

$$P_{n'} = \frac{Ps}{SNR_{des}} - P_n \tag{3.22}$$

Pour rappel, P_s est la puissance du préambule et P_n est la puissance du bruit initial.

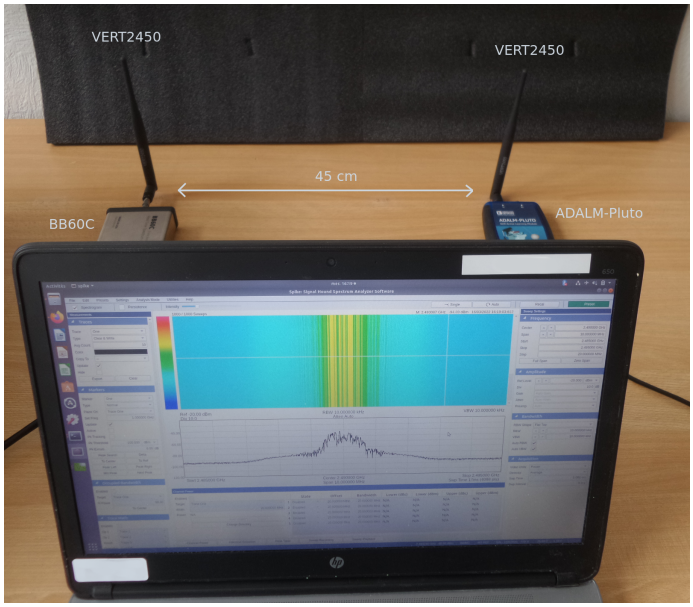


FIGURE 3.27 – Configuration du banc d'essai de l'expérimentation des *RF eigenfingerprints*

Les performances de trois classifieurs [22] ont été étudiées durant cette expérimentation :

— Classifieur 1 : un classifieur utilisant le pré-traitement n°2 (figure 3.19) suivi d'une classification par maximum de vraisemblance.
— Classifieur 2 : un classifieur utilisant le pré-traitement n°1 (figure 3.20) suivi d'une classification par maximum de vraisemblance.
— Classifieur 3 : un classifieur utilisant le pré-traitement n°1 (figure 3.19), suivie d'une classification bayésienne naïve [86] basée sur la combinaison du modèle probabiliste

22. Pour faciliter la comparaison avec notre article [145], nous avons choisi de garder les mêmes dénominations pour les classifieurs.

présenté en équation 3.10, ainsi que d'une loi normale pour modéliser le décalage fréquentiel $\hat{\Delta f}$ [23].

La matrice de données $X \in \mathbb{C}^{(250\times200)}$ est divisée en deux matrices à l'aide d'une décomposition 70% - 30 % : la matrice d'entraînement $X_{train} \in \mathbb{C}^{(250\times140)}$ (70 % des données) et une matrice de test $X_{test} \in \mathbb{C}^{(250\times60)}$ (30 % des données). Ainsi, le jeu d'entraînement est composé d'uniquement 140 signaux (soit environ 35 signaux par classe) pour apprendre les différents paramètres des classifieurs. Puis, les trois classifieurs apprennent un espace de projection composé de 8 *RF eigenfingerprints*, i.e. un vecteur $\bar{x}$ de 1×250 éléments et une matrice U_{proj} de (8×250) éléments, ainsi que les centroïdes μ_c de chaque classe, soit 2 250 éléments au total.

Une fois l'entraînement des 3 classifieurs réalisé à l'aide de la matrice X_{train}, nous avons réalisé une évaluation des performances des différents classifieurs grâce à la matrice X_{test}. Nous pouvons voir sur la figure 3.28, la comparaison des performances des différents classifieurs. Le classifieur n°1 obtient les meilleures performances de classification, cependant le classifieur n°3 offre des performances de classification acceptables pour des valeurs de SNR allant de 10 à 30 dB de SNR. Tout comme pour l'entraînement, le rapport signal à bruit de chaque signal de la matrice de données de test X_{test} est initialement fixé à 30 dB. Puis selon le rapport signal à bruit désiré, nous avons réinjecté un bruit blanc gaussien complexe permettant d'atteindre le SNR désiré. Ainsi, cela nous a permis d'évaluer les performances de classification pour des valeurs de SNR allant de -10 dB à 30 dB.

Comme nous venons de l'évoquer, le classifieur n°1 présente les meilleures performances par rapport aux deux autres classifieurs. Cependant, nous avons montré, dans l'annexe A.1, que les descripteurs appris sont moins interprétables par rapport à ceux des autres classifieurs. De plus, il est possible qu'en présence de décalages fréquentiels importants, le nombre de *RF eigenfingerprints* nécessaires explose. Ainsi, le classifieur n°3 pourrait être considéré comme un meilleur classifieur avec un bon compromis entre explicabilité, complexité et performances.

De manière plus générale, l'annexe A présente l'influence de différents défauts non pris en compte dans le modèle d'imperfections présenté en figure 3.21 sur les *RF eigenfingerprints* appris, comme le décalage fréquentiel, le bruit de phase ou le multi-trajet. Ainsi, nous pouvons constater que le nombre de *RF eigenfingerprints* est influencé par

23. La décision par maximum *a posteriori* revient à trouver $y = arg\ min_c\ \frac{1}{\sigma^2}\|z-\mu_c\|^2 + \frac{1}{2\sigma_f^2}(\hat{\Delta}f - f_c)^2$ avec σ_f^2 la variance de l'estimateur de $\hat{\Delta}f$ pour une valeur de variance de bruit σ^2 donnée et f_c la fréquence moyenne de la classe c.

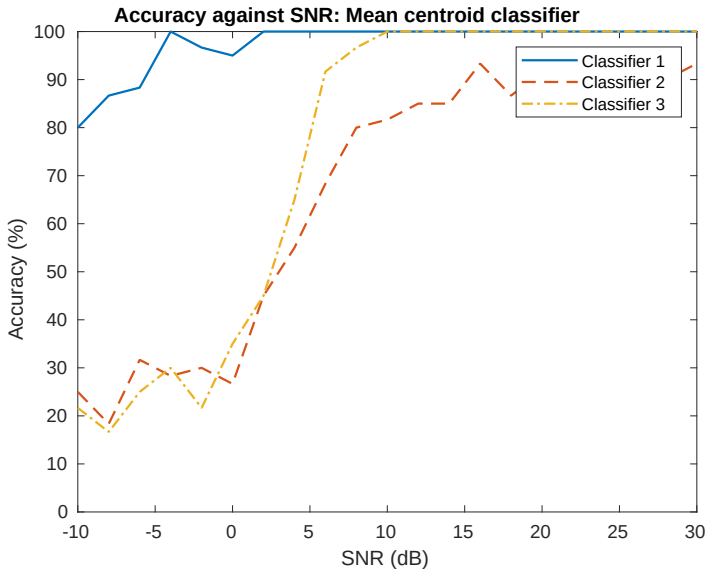


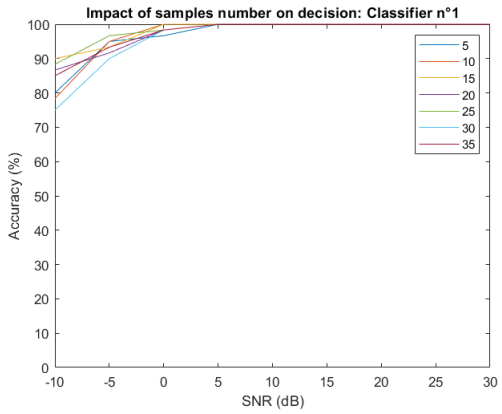
FIGURE 3.28 – Comparaison des performances des classifieurs basés sur les *RF eigenfingerprints*

ces défauts, motivant ainsi l'utilisation d'une phase de pré-traitement pour supprimer ces sources de variabilités non-informatives.

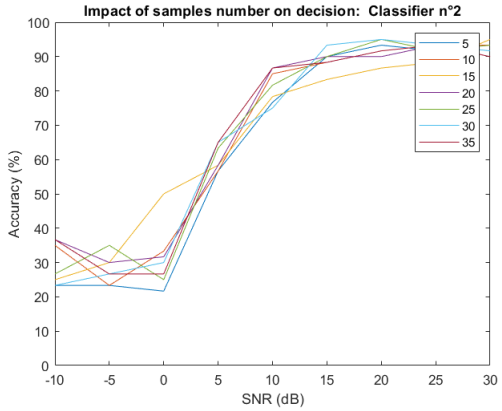
3.3.2.3 Expérimentation 2 : Étude de l'influence du nombre de données d'entraînement

Afin d'étudier les propriétés de scalabilité de la méthode des *RF eigenfingerprints*, nous avons réalisé une seconde expérimentation qui n'est pas présente dans l'article [145]. Cette expérimentation consiste à évaluer l'influence du nombre d'exemples d'entraînement sur les performances des différents classifieurs. Les résultats sont présentés en figure 3.29 et ils montrent qu'il est possible de réduire le nombre d'exemples pour chaque classe sans trop dégrader les performances de classification. Ainsi, au lieu d'utiliser 35 exemples par classe comme pour l'expérimentation précédente, il est possible d'utiliser seulement une dizaine d'exemples par classe pour l'apprentissage des différents paramètres des classifieurs.

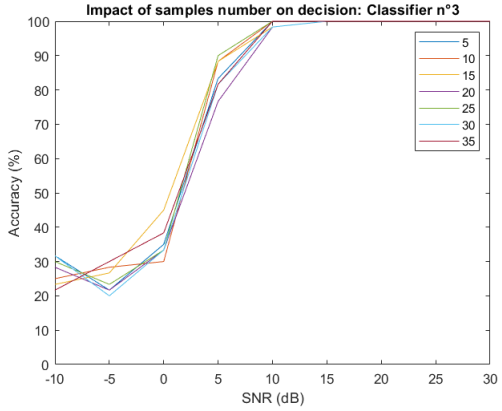
Cependant, il existe des contraintes mathématiques concernant le nombre d'exemples d'apprentissage minimums requis. D'une part, pour l'apprentissage des *RF eigenfingerprints* (et du signal moyen $\bar{x}$), il est nécessaire d'avoir plus d'exemples d'apprentissage que de vecteurs propres sélectionnés (ici 8). En effet, comme indiqué par Brunton et al. dans [137], cela est dû aux propriétés de la décomposition en valeurs singulières, car le rang de la matrice d'apprentissage dépend en partie du nombre L d'exemples utilisés. Plus



(a) Classifieur 1



(b) Classifieur 2



(c) Classifieur 3

FIGURE 3.29 – Influence du nombre d'échantillons sur les performances des *RF eigenfingerprints*

particulièrement, lors d'une décomposition en valeurs singulières, la matrice $\Sigma \in \mathbb{R}^{N \times L}$ contient $min(N, L)$ valeurs singulières. Ainsi, dans le cas où $N > L$, il y aura L valeurs singulières non nulles. D'autre part, pour l'apprentissage des centroïdes μ_c de chaque classe, l'équation 3.13 donne le nombre minimum d'exemples nécessaires par classe. Par exemple, pour le classifieur n°1, avec un SNR de 30 dB et en prenant en compte 1/10 de la variance inter-classe pour λ_{min}, ainsi qu'un seuil de 95 %, le nombre d'exemples minimum est de 10. Cependant, il ne s'agit que d'une borne mathématique et les performances des classifieurs se comportent très bien, même en deçà de cette borne, comme nous pouvons l'observer sur la figure 3.29.

3.3.3 Comparaison avec la littérature

Dans cette section, nous allons comparer la méthode des *RF eigenfingerprints* par rapport aux travaux de la littérature sur plusieurs points : les performances de classification, la scalabilité, la complexité et l'explicabilité.

3.3.3.1 Performances de classification

Nous allons tout d'abord comparer nos performances de classification par rapport aux travaux de Riyaz et al. [28]. En effet, ces travaux ont des conditions expérimentales assez proches des nôtres. Plus particulièrement, les auteurs ont évalué les performances de leur méthode avec 5 plateformes SDR (USRP B210/X310) avec une fréquence d'échantillonnage de 1.92 Msps, ainsi qu'une fréquence centrale 2.45 GHz. Pour une valeur de SNR de 20 dB, leur architecture obtient une *accuracy* de 90% alors que nos classifieurs n°1 et n°3 obtiennent des meilleures performances et que notre classifieur n°2 obtient des performances similaires.

Dans notre article [145], nous nous sommes aussi comparés aux travaux de Qing et al. [72], où ils présentent une architecture CNN légère d'authentification par empreinte radio avec 4 appareils ZigBee. De manière générale, notre classifieur n°1 présente de meilleures performances (> 93 %), même pour des valeurs de SNR faible. De plus, nos classifieurs n°2 et n°3 présentent de meilleures performances pour des valeurs de SNR fortes (10-30 dB).

3.3.3.2 Scalabilité

Les approches classiquement utilisées pour l'apprentissage de représentations nécessitent un nombre d'exemples important. Par exemple, dans [28], l'architecture utilisée est entraînée à l'aide de 720 000 signaux I/Q, i.e. 180 000 par classe. Comme nous l'avons montré dans la sous-section précédente, notre approche a besoin de peu d'exemples d'apprentissage (une dizaine d'exemples par classe) pour apprendre l'espace de projection, ainsi que la représentation d'une classe. Même si certains travaux de la littérature du RFF ont montré qu'il était possible de réaliser du *one-shot learning* avec des réseaux siamois [69].

3.3.3.3 Complexité

Les approches classiquement utilisées pour l'apprentissage de représentations nécessitent généralement une puissance de calcul importante, ainsi qu'un espace mémoire conséquent pour les stocker. Par exemple, dans [28], l'architecture présentée est composée de plus de 400 000 paramètres. Dans la littérature, certains articles traitent de l'implémentation d'architectures légères de réseaux de neurones. Par exemple, dans [72] et [71], les auteurs présentent des réseaux de neurones constitués d'environ 200 000 paramètres. Par rapport à ces réseaux de neurones légers, la méthode des *RF eigenfingerprints* a une complexité beaucoup plus faible. En effet, en prenant l'exemple des classifieurs introduits précédemment, seulement 8 vecteurs propres sont nécessaires à l'apprentissage de l'espace de projection, soit 2 250 coefficients complexes ou 4 500 coefficients réels. De la même manière, la phase de projection nécessite 2 000 multiplications complexes (ou 8 000 multiplications réelles) et la phase de classification 32 multiplications complexes (ou 128 multiplications réelles). Il s'agit ainsi d'un petit modèle selon les critères du tableau 3.1.

Le tableau 3.14 est récapitulatif de la complexité des différentes phases de la méthode des *RF eigenfingerprints*.

Type	**Calculs**	**Mémoire**
Projection	$O(N \times K)$	$O(N \times K)$
Vérification	$O(K)$	$O(K \times C)$
Identification	$O(K \times C)$	$O(K \times C)$
Classification	$O(K \times C)$	$O(K \times C)$

TABLE 3.14 – Complexité calculs et mémoire des *RF eigenfingerprints*

3.3.3.4 Explicabilité

Les approches d'apprentissage de représentations reposent, la plupart du temps, sur des architectures de réseaux de neurones profonds. Cependant, ces réseaux de neurones sont généralement considérés comme des boîtes noires, limitant ainsi leur utilisation dans des contextes critiques. Peu d'articles du domaine du RFF traitent de l'explicabilité des modèles, particulièrement pour les réseaux de neurones. À notre connaissance, seul Kuzbeda et al. [146] proposent une approche pour interpréter les descripteurs appris par leur méthode appelée RifNet. Dans cette section, nous avons montré les propriétés d'explicabilité de la méthode des *RF eigenfingerprints*, notamment grâce aux résultats présentés dans la figure 3.24 et 3.25.

3.4 Comparaison des méthodes de RFF proposées et cas d'applications

Dans cette section, nous allons comparer les méthodes de RFF proposées par rapport aux propriétés d'utilisabilité introduites dans la section 3.1. Ensuite, nous proposerons trois cas d'utilisation du RFF dans le contexte de la sécurité de l'IoT et nous positionnerons nos méthodes par rapport à ces cas d'applications.

3.4.1 Comparaison des méthodes de RFF proposées

Dans la section 3.1, nous avons proposé trois propriétés d'utilisabilité que les méthodes de RFF doivent prendre en compte pour pouvoir être utilisées dans un contexte IoT. Ainsi, les méthodes présentées dans les sections 3.2 et 3.3 prennent en compte ces propriétés, mais présentent aussi des différences vis-à-vis de ces propriétés.

3.4.1.1 Adaptabilité

Comme nous l'avons évoqué dans la section 3.1, nous avons choisi d'utiliser des méthodes basées sur l'apprentissage de représentations pour prendre en compte cette propriété. D'une part, les réseaux siamois sont des architectures d'apprentissage profond faisant partie de l'apprentissage de représentations comme indiqué dans la figure 3.1. D'autre part, la méthode des *RF eigenfingerprints* peut être vue comme une analyse en composantes principales centrée, faisant donc partie des méthodes d'apprentissage de re-

présentations à une couche selon Bengio [106]. Ainsi, les deux méthodes de RFF proposées permettent de s'adapter à l'hétérogénéité des protocoles IoT grâce à l'apprentissage de représentations.

3.4.1.2 Scalabilité

La propriété de scalabilité est évaluée en fonction de deux sous-propriétés : le FSL et la RP. Concernant la RP, les deux méthodes de RFF proposées sont capables d'ajouter/supprimer l'empreinte d'un émetteur légitime, sans réutiliser les données des classes déjà apprise par l'algorithme. Cela est rendu possible grâce à la séparation de l'espace de projection et de la représentation des classes. Concernant l'apprentissage frugal, nos deux méthodes se différencient sur ce point. D'une part, les réseaux siamois sont capables d'apprendre la représentation d'une nouvelle classe à l'aide d'une unique exemple [69] : le *one-shot learning*. D'autre part, la méthode des *RF eigenfingerprints* nécessite une dizaine d'exemples pour apprendre la centroïde d'une nouvelle classe. Cependant, l'apprentissage de l'espace de projection nécessite plus d'exemples d'entraînement pour les réseaux siamois que pour la méthode des *RF eigenfingerprints*, mais cet apprentissage n'est réalisé qu'une fois.

3.4.1.3 Complexité

La propriété de complexité est évaluée en fonction de deux sous-propriétés : la complexité mémoire et la complexité de calcul. L'architecture légère du réseau siamois, présentée en figure 3.5 et composée d'environ 380 000 paramètres, est considérée comme un grand modèle selon les critères du tableau 3.1. À l'inverse, la méthode des *RF eigenfingerprints*, composée de 4 500 paramètres (réels) dans le cas des classifieurs présentés, dans la sous-section 3.3.2, est considérée comme un petit modèle selon ces mêmes critères.

3.4.1.4 Récapitulatif de la comparaison

Le tableau 3.15 récapitule la comparaison des deux méthodes de RFF proposées. Nous pouvons constater que les réseaux siamois sont plus adaptés pour les cas d'utilisation nécessitant des bonnes propriétés de scalabilité, alors que les *RF eigenfingerprints* semblent plus adaptées à des cas d'utilisation nécessitant une complexité faible.

	Adaptabilité	**Scalabilité**		**Complexité**	
		FSL	RP	Mémoire	Calcul
Réseaux siamois	✓	Excellente	✓	Forte	Forte
RF eigenfingerprints	✓	Bonne	✓	Faible	Faible

TABLE 3.15 – Récapitulatif de la comparaison des méthodes de RFF proposées

3.4.2 Cas d'applications

Nous allons maintenant présenter différents cas d'applications du RFF dans le contexte de la sécurité de l'IoT. Plus particulièrement, nous proposons trois cas d'applications, appelés configurations, qui sont les suivants : configuration de surveillance, configuration asymétrique et configuration symétrique. Enfin, nous positionnerons les méthodes de RFF proposées par rapport à ces différents cas d'applications, en fonction des spécificités de chacune.

Dans cette section, nous utiliserons trois termes : objet, passerelle et capteur RFF. Les objets sont les appareils constituant le réseau IoT et la passerelle est une plateforme faisant le lien entre les objets et internet. Enfin, le terme capteur RFF fait référence à une plateforme exécutant une méthode de RFF. Il peut s'agir soit d'une plateforme externe au réseau, soit d'une passerelle ou d'un objet.

3.4.2.1 Cas d'application 1 : configuration de surveillance

Le premier cas d'application, appelé configuration de surveillance et décrit en figure 3.30, consiste à utiliser des capteurs RFF embarqués sur des plateformes externes au réseau IoT pour la surveillance des communications. Ainsi, selon les ressources disponibles sur les plateformes extérieures, il faudra trouver un compromis entre scalabilité et complexité. Ce cas d'application est notamment proposé dans [1], où Brik et al. présentent une approche permettant de détecter la présence d'émetteurs illégitimes à l'aide de capteurs RFF. Ce cas d'utilisation a été repris comme exemple dans l'implémentation Matlab [147] des travaux de [29]. En effet, dans cet exemple, le RFF était utilisé pour détecter une attaque *evil twin*.

3.4.2.2 Cas d'application 2 : configuration asymétrique

Le second cas d'application, appelé configuration asymétrique et décrite en figure 3.31, consiste à implémenter une méthode de RFF sur une passerelle dans le cas d'un réseau

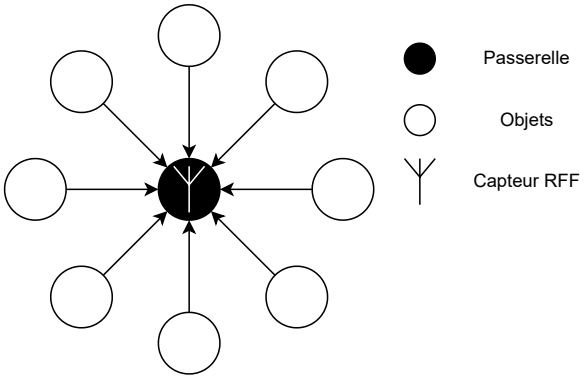


FIGURE 3.30 – Configuration de surveillance

IoT basé sur une topologie en étoile, comme pour certains réseaux de capteurs [44]. Plus particulièrement, il s'agit de considérer le cas de communications unidirectionnelles allant des objets vers la passerelle. Le terme asymétrique venant du fait que cette configuration exploite l'asymétrie de la situation : asymétrie des communications, asymétrie du RFF et asymétrie des capacités de calculs. Ainsi, comme la méthode de RFF est exécutée sur la passerelle, il n'y a pas forcément de limites quant aux ressources, mais de nombreux objets peuvent être gérés par cette configuration. Ce cas d'utilisation a notamment été évoqué par Peng et al. dans [44].

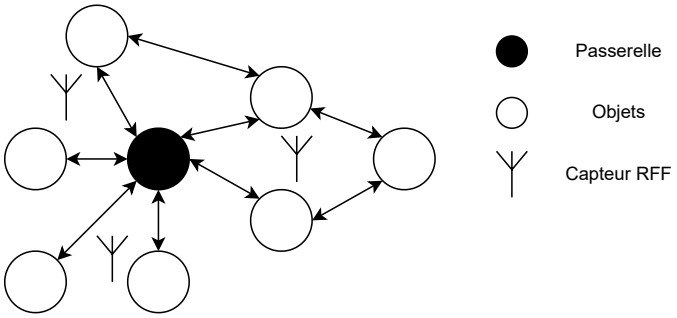


FIGURE 3.31 – Configuration asymétrique

3.4.2.3 Cas d'application 3 : configuration symétrique

Le dernier cas d'application, appelée configuration symétrique et décrite en figure 3.32, consiste à implémenter une méthode de RFF directement sur un objet, par exemple comme second facteur d'authentification. Le terme symétrique, par opposition à asymétrique, vient du fait que les communications peuvent être symétriques. Ainsi, comme la méthode

de RFF est exécutée sur l'objet, il est nécessaire qu'elle soit économe en ressources, mais l'objet n'a pas forcément besoin d'interagir avec de nombreux objets du réseau. Ce cas d'application est notamment présenté par Qing et al dans [72].

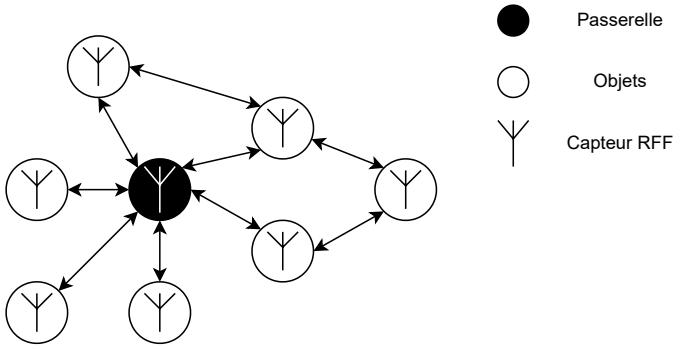


FIGURE 3.32 – Configuration symétrique

3.4.2.4 Positionnement des méthodes de RFF proposées par rapport aux configurations

Le tableau 3.16 présente les différents cas d'applications des deux méthodes de RFF proposées. Premièrement, nous pouvons remarquer qu'elles sont toutes les deux adaptées à une configuration de surveillance. En effet, selon les limitations de ressources du capteur RFF et le nombre d'appareils à gérer, l'une ou l'autre peut être envisagée. Ensuite, la configuration asymétrique nécessite l'utilisation de réseaux siamois, car la passerelle a peu de contraintes en termes de ressources, mais elle est susceptible de gérer un grand nombre d'objets, il convient donc de favoriser la scalabilité. Enfin, la configuration symétrique nécessite l'utilisation des *RF eigenfingerprints* à cause des ressources limitées de certains objets IoT, il convient donc de favoriser la complexité. Il est important de garder à l'esprit qu'il ne s'agit que d'un positionnement de notre part, mais que d'autres cas d'applications, voir d'autres utilisations peuvent être envisagées.

	Conf.de surveillance	Conf. asymétrique	Conf. symétrique
Réseaux siamois	✓	✓	×
RF eigenfingerprints	✓	×	✓

TABLE 3.16 – Cas d'applications des méthodes de RFF proposées

Chapitre 4

Implémentations logicielles et matérielles des méthodes de RFF proposées

Ce chapitre a pour but de présenter des implémentations logicielles et matérielles des méthodes de RFF introduites dans le chapitre 3 : les réseaux siamois pour le RFF et les *RF eigenfingerprints*. En effet, dans le chapitre précédent, nous avions introduit la notion de complexité, ainsi que les sous-propriétés et métriques associées : le nombre de paramètres, ainsi que le nombre de multiplications. Ces métriques avaient été notamment choisies, car elles étaient indépendantes de l'implémentation d'un modèle. Dans ce chapitre, nous choisirons des métriques permettant de comparer différentes implémentations logicielles et matérielles.

Les trois hypothèses de départ que nous ferons pour les différentes implémentations sont les suivantes :

— Nous ne considérerons que des implémentations logicielles sur CPU et des implémentations matérielles sur FPGA.

— Nous nous intéresserons à l'implémentation des phases de projection de l'inférence. En effet, comme nous l'avons précédemment expliqué dans le chapitre 3, il est d'abord nécessaire de projeter le signal d'entrée dans un espace latent avant de calculer une distance entre la projection de l'entrée et la représentation d'une classe. Cette phase de projection est celle qui nécessite le plus de calculs, ainsi que le plus d'espace mémoire pour stocker les différents coefficients. Cela permettra également d'être indépendant du type d'authentification, que ce soit une vérification ou une identification qui ne requière pas la même quantité de calculs, mais qui reposent toutes deux sur la projection du signal d'entrée dans l'espace latent.

— Nous utiliserons uniquement de la quantification post-entraînement des paramètres du modèle, i.e le modèle ne sera pas réentraîné spécifiquement pour l'implémenta-

tion. Cela permet d'avoir des implémentations plus proches des modèles présentés dans le chapitre précédent et donc de pouvoir plus facilement faire un parallèle entre les projections des modèles appris et de leurs implémentations.

4.1 Justification des plateformes, des outils et des métriques

Cette section présente les différentes plateformes, outils et métriques qui seront utilisés tout au long de ce chapitre, ainsi que les justifications de ces choix. Nous nous sommes partiellement inspirés des plateformes, des outils, voire aussi de certaines métriques proposées dans le *MLPerf Tiny Benchmark* [116], sur lequel nous nous étions également en partie basés dans le chapitre précédent.

4.1.1 Plateformes

4.1.1.1 Plateforme logicielle

La première plateforme est un ordinateur monocarte Raspberry Pi 4 modèle B (4 GB). Il est composé d'un système sur puce ou *System on Chip* (SoC) Broadcom BCM2711 intégrant 4 cœurs ARM Cortex-A72 (ARM-v8) avec 4GB de mémoire RAM. Nous l'avons utilisé avec Raspbian OS Buster basé sur un noyau Linux 5.10.

La Raspberry Pi 4B sera la plateforme utilisée pour caractériser les implémentations logicielles, i.e les implémentations CPU. Il s'agit d'une plateforme très répandue dans le monde académique, notamment grâce à son faible prix et à sa facilité de prise en main. De plus, la Raspberry Pi 4 B est l'une des plateformes du *MLPerf Tiny Benchmark* [116]. Enfin, elle permet l'acquisition de signaux radio, soit grâce à une carte d'extension, soit à l'aide d'une plateforme SDR comme l'Adalm-PLUTO [87].

4.1.1.2 Plateforme matérielle

La seconde plateforme est une carte de développement Zedboard de chez Digilent. Elle est composée d'un SoC Zynq-7020 de Xilinx/AMD intégrant 2 cœurs ARM Cortex-A9, ainsi qu'un FPGA Artix-7.

La Zedboard sera la plateforme utilisée pour caractériser les implémentations matérielles, i.e les implémentations FPGA. Tout comme la Raspberry Pi 4B, il s'agit d'une

plateforme très répandue dans le monde académique grâce à son faible prix, ainsi qu'à sa facilité de prise en main. De plus, le SoC Zynq-7020, qui compose la Zedboard, a déjà été utilisé dans certaines publications pour le *MLPerf Tiny Benchmark* [148], ainsi que dans la littérature du RFF [73]. Enfin, il est aussi possible de faire l'acquisition de signaux radio depuis la Zedboard grâce à la carte d'extension FMCOMMS3 de Analog Devices basée sur un AD9361 [149].

Pour la partie sur l'implémentation matérielle, nous n'avons pas réalisé l'implémentation sur carte et nous nous sommes limités aux estimations fournies par les différents outils. Cela permet notamment d'évaluer seulement l'implémentation sans traiter les problématiques d'intégrations finales : accélérateur matériel, bloc de traitements FPGA, ...

4.1.2 Outils

4.1.2.1 Outils pour les réseaux siamois

L'implémentation logicielle des réseaux siamois a été réalisée à l'aide de Tensorflow Lite en Python, un module de Tensorflow permettant d'implémenter des réseaux de neurones sur différentes plateformes logicielles [150], [151]. Cet outil permet de faire de la quantification post-entraînement à partir d'un modèle Keras et il s'agit également d'un des outils mentionnés[1] dans le *MLPerf Tiny Benchmark* [116]. Nous sommes contient que le choix de Python est critiquable. Cependant, il se justifie par le fait que nous voulions faire tourner à la fois le modèle Tensorflow, ainsi que son implémentation logicielle Tensorflow Lite dans un même environnement, pour faciliter leur comparaison. De plus, l'utilisation de Tensorflow Lite en Python permet aussi une meilleure reproductibilité de nos travaux, notamment via l'utilisation d'un environnement virtuel [150].

L'implémentation matérielle des réseaux siamois est réalisée à l'aide de l'outil hls4ml [152] du CERN et Vivado HLS de Xilinx/AMD [153]. hls4ml est un outil permettant de générer du code de haut niveau (C/C++), qui est ensuite traduit à l'aide de Vivado HLS dans un langage de description matériel comme VHDL ou Verilog. . Il s'agit également d'un des outils mentionnés pour le *MLPerf Tiny Benchmark* [116], [148].

1. Dans le *MLPerf Tiny Benchmark*, il s'agit de Tensorflow Lite for Microcontrollers qui est utilisé, une variante de Tensorflow Lite conçu pour les microcontrôleurs [151]. Cependant, comme la plateforme logicielle considérée est une Raspberry Pi 4 B, nous avons choisi d'utiliser Tensorflow Lite, mieux adapter aux appareils mobiles et de périphérie [150], [151].

4.1.2.2 Outils pour les *RF eigenfingerprints*

Contrairement aux réseaux siamois, l'implémentation logicielle des *RF eigenfingerprints* a été principalement guidée par l'implémentation matérielle, incluse dans notre publication [145].

L'implémentation matérielle a été réalisée avec Vivado HLS. Comme nous l'avons introduit précédemment, Vivado HLS est un outil de synthèse de haut niveau permettant de convertir un code de haut niveau écrit en C/C++ dans un langage de description matériel comme VHDL ou Verilog. Nous avons utilisé cet outil pour réaliser l'implémentation matérielle des *RF eigenfingerprints*, car il permet de faciliter l'exploration rapide de l'espace des solutions architecturales.

L'implémentation logicielle a été réalisée en C++ en s'inspirant du code de l'implémentation matérielle, pour un maximum de réutilisabilité du code utilisé avec Vivado HLS.

4.1.3 Métriques

Dans ce chapitre, nous considérons trois types de métriques. Premièrement, les métriques de reconstruction permettant de comparer le modèle, aussi appelé modèle de référence, avec son implémentation. Deuxièmement, les métriques pour la complexité de calculs, qui ont pour but d'évaluer la quantité de calculs de chaque inférence. Enfin, les métriques pour la complexité mémoire, qui ont pour but d'évaluer l'occupation de la mémoire/surface matérielle d'une implémentation.

4.1.3.1 Métriques de reconstruction

Ce premier type de métriques nous permet de comparer le modèle avec son implémentation, en mesurant l'erreur de reconstruction (ou erreur de projection) entre la projection réalisée par le modèle de référence et celle de son implémentation. Cela permet de constater l'impact de la quantification post-entraînement sur le résultat de la phase de projection. En effet, le fait de quantifier un modèle va forcément avoir un impact sur la projection puisque les coefficients du modèle seront quantifiés.

La métrique de reconstruction que nous avons définie et utilisée est la métrique dite

Mean Relative Square Error (MRSE) :

$$MRSE = \frac{1}{L}\sum_{i=1}^{L}\frac{\|z_i - \hat{z}_i\|_2^2}{\|z_i\|_2^2} \tag{4.1}$$

avec :

— L : le nombre de projections réalisées.

— z_i : la projection du i-ème signal avec le modèle de référence.

— $\hat{z}_i$: la projection du i-ème signal avec l'implémentation du modèle de référence.

Cette métrique de reconstruction dépend de l'erreur quadratique relative de projection $\|z_i - \hat{z}_i\|_2^2 \,/\, \|z_i\|_2^2$, c'est-à-dire qu'elle est indépendante de la norme de la projection à partir du modèle de référence $\|z_i\|_2^2$. Pour cela, une normalisation par la norme (L_2) au carré de la projection du modèle de référence est utilisée.

Le choix de cette métrique est justifié par le fait que son calcul ne nécessite pas de racine carrée, comme pour l'erreur absolue moyenne [144], qui peut être difficile à calculer dans le cas de représentation en virgule fixe, car basée sur des nombres entiers. De plus, l'avantage de cette métrique par rapport à d'autres métriques, comme l'erreur moyenne quadratique [86], est la normalisation par la norme au carré de la projection avec le modèle de référence. Ainsi, cette normalisation permet d'avoir une erreur relative par rapport à la norme de la projection et donc de pouvoir l'exprimer sous forme de pourcentage.

En ce qui concerne la valeur de MRSE acceptable, nous considérerons 0.25 % comme une valeur acceptable. En effet, 0.25 % de MRSE correspond au fait, qu'en moyenne, l'erreur de reconstruction ($\|z_i - \hat{z}_i\|_2$) entre la projection du modèle de référence et la projection réalisée par son implémentation représente 5 % de la norme ($\|z_i\|_2$) de la projection du modèle de référence, car $0.05^2 = 0.0025$ (0.25 %). Ce choix, bien qu'empirique, nous permettra de mesurer l'erreur de reconstruction de nos différentes implémentations.

4.1.3.2 Métrique pour la complexité de calculs

Dans le chapitre précédent, nous avions introduit le nombre de multiplications comme estimation de la complexité de calculs d'un modèle. Dans ce chapitre, nous considérerons la latence comme métrique de complexité de calculs, i.e le temps d'une inférence. En effet, comme indiqué dans [114], les performances d'un système doivent être calculées en termes de latence, même si généralement les auteurs de la littérature préfèrent une approximation en considérant le nombre de multiplications-additions.

Pour l'implémentation matérielle, nous prendrons en compte une seconde métrique :

l'*Initiation Interval* (II). Cette métrique nous permet d'avoir une estimation du débit de notre modèle, i.e le nombre d'inférences par seconde. En effet, dans les implémentations matérielles, il est courant de faire du *pipelining* de traitements. Ainsi, l'II nous indique le temps qu'il faut attendre pour débuter une nouvelle inférence, avant même que l'inférence précédente soit terminée. Cependant, pour les implémentations matérielles, la notion de latence est importante puisqu'elle correspond à la réactivité du système. Plus la latence sera faible, plus le système pourra réagir rapidement dans le cas d'une usurpation. De plus, il s'agit d'une métrique largement utilisée dans la littérature [10], [30], [73]. Nous inclurons donc à titre indicatif la latence de nos implémentations matérielles dans nos résultats.

Que ce soit la latence pour l'implémentation logicielle ou l'II pour l'implémentation matérielle, ces deux métriques ont pour but d'estimer le nombre d'inférences par seconde. Dans le chapitre précédent, les valeurs de multiplications utilisées considéraient un nombre de 10 inférences par seconde pour une cible embarquée [113]. Ainsi, dans notre cas, le tableau 4.1 permet de classer une implémentation logicielle et matérielle en fonction de son nombre d'inférences par seconde. Pour cela, la borne supérieure correspond à une implémentation lente, car reprenant le critère de 10 inférences par seconde présenté dans [113]. Puis, nous descendons à 1 000 inférences par seconde pour une implémentation modérée et 100 000 inférences par seconde pour une implémentation rapide. Le choix des valeurs pour cette métrique est essentiellement empirique et se base sur la seule contrainte disponible, celle d'une borne supérieure de 10 inférences par seconde. Cependant, la borne inférieure de 100 000 inférences par seconde signifie qu'il est possible théoriquement d'analyser 100 000 signatures par seconde (en négligeant le temps de calculs de la phase de décision), ce qui est plus rapide que certains systèmes de détection d'intrusion réseaux [154].

Type d'implémentations	Inférences/s
Lente	10
Modérée	1 000
Rapide	100 000

TABLE 4.1 – Valeurs utilisées pour l'évaluation de la complexité de calculs d'une implémentation logicielle et matérielle

4.1.3.3 Métrique pour la complexité mémoire

Dans le chapitre précédent, nous avions introduit le nombre de paramètres d'un modèle comme estimation de la complexité mémoire de celui-ci. En effet, comme nous l'avions expliqué, cela revenait à considérer une quantification 8-bits des coefficients du modèle [113], [114].

Pour l'implémentation logicielle, nous considèrerons la taille en octets d'un modèle comme métrique pour la complexité mémoire. Cette métrique étant une meilleure approximation de l'occupation mémoire d'un modèle, car dépendant de la quantification de celui-ci, ainsi que de son implémentation. Nous réutiliserons les valeurs pour l'évaluation d'une implémentation proposée dans [113] et présentée dans le tableau 4.2.

Type d'implémentations	Occupation mémoire
Petit	80 000 octets
Moyen	200 000 octets
Grand	500 000 octets

TABLE 4.2 – Valeurs utilisées pour l'évaluation de la complexité mémoire d'une implémentation logicielle

Pour l'implémentation matérielle, nous utiliserons plutôt la notion de surface matérielle [153], i.e l'ensemble des ressources matérielles d'un FPGA. En effet, les FPGA sont des composants matériels reprogrammables composés d'une multitude de ressources matérielles [2] comme des *look-up tables* (LUT), des registres (FF), des blocs de mémoire (BRAM18K) et des blocs de traitement du signal (DSP48E) [155]. Comme il est difficile de définir une métrique globale prenant en compte les différents types de ressources matérielles, nous comparerons l'utilisation des ressources à celle de la Zedboard (voir tableau 4.3). Cela correspondra donc à une métrique binaire, soit l'implémentation est possible sur Zedboard, soit l'implémentation n'est pas possible sur Zedboard. Pour faciliter la compréhension du lecteur, nous afficherons les ressources, ainsi que leur pourcentage d'utilisation (en bleu) par rapport aux ressources de la Zedboard.

BRAM18K	DSP48E	LUT	FF
280	220	53 200	106 400

TABLE 4.3 – Récapitulatif de la surface matérielle d'une Zedboard

2. Nous n'avons pas pris en compte l'*Ultra RAM*, le Zynq-7020 de la Zedboard n'en comportant pas.

4.2 Implémentation logicielle et matérielle des réseaux siamois pour le RFF

Dans cette section, nous nous focaliserons sur l'implémentation de l'architecture légère présentée en figure 3.5, car les conclusions du chapitre précédent ont montré que seul cette architecture respecte les contraintes de complexité décrites dans la section 3.1.3.

Pour les deux implémentations, nous avons considéré uniquement les $L = 1\,000$ premiers signaux du jeu de test pour accélérer les tests, enregistrés au format *pickle* (.pkl).

4.2.1 Étape préliminaire à l'implémentation matérielle et logicielle des réseaux siamois

L'étape commune à l'implémentation logicielle et matérielle de l'architecture légère proposée dans le chapitre précédent est présentée en figure 4.1 et consiste à extraire le projecteur du réseau siamois, i.e le sous-réseau de neurones ($G_W(.)$) servant à la projection des entrées dans l'espace d'enchâssement. Comme nous l'avions indiqué dans la section 3.2.2.4, l'architecture légère a été entraînée à l'aide d'un paradigme logistique. Le réseau siamois est donc composé d'un sous-réseau de neurones servant à la projection des entrées, ainsi que d'une couche dense servant à calculer la distance L_1 pondérée. Cette étape d'extraction consiste ainsi à extraire le projecteur du réseau siamois et à l'enregistrer sous la forme d'un fichier .h5, un format commun pour l'enregistrement des modèles Tensorflow.

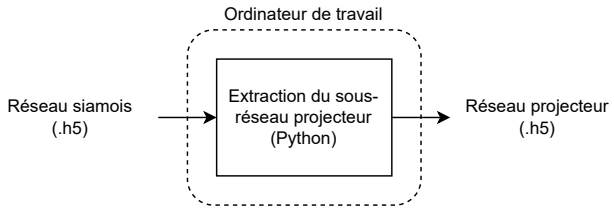


FIGURE 4.1 – Étape préliminaire à l'implémentation des réseaux siamois

4.2.2 Implémentation logicielle des réseaux siamois avec Tensorflow Lite

La figure 4.2 résume les différentes phases de l'implémentation logicielle pour les réseaux siamois à l'aide de Tensorflow Lite.

La première étape de cette implémentation consiste à convertir le réseau de neurones projecteur sauvegardé au format .h5 en un format compréhensible par Tensorflow Lite (.tflite). Pour cela, nous nous sommes basés sur l'article de Duque et al. [150] sur l'implémentation de réseau de neurones à l'aide de Tensorflow Lite en utilisant Tensorflow 2.0. Nous avons donc créé deux fichiers .tflite, correspondant à deux conversions du réseau de neurones projecteur. La première implémentation correspond à une conversion classique (en virgule flottante) du modèle sans quantification des coefficients de celui-ci, que nous appellerons DEFAULT_TFLITE. La seconde correspond à une conversion du modèle avec une quantification 8-bits des coefficients du modèle, que nous appellerons OPTIMIZED_TFLITE [151].

La seconde étape de l'implémentation consiste à évaluer, pour chaque implémentation [3], l'erreur de reconstruction, l'occupation mémoire, ainsi que sa latence sur la Raspberry Pi 4B. L'erreur de reconstruction est calculée à partir du modèle Tensorflow de référence sur la Raspberry Pi 4B. De plus, la latence a été mesurée à l'aide d'un script différent de celui utilisé pour mesurer l'erreur de reconstruction, pour éviter toute interférence entre les deux mesures.

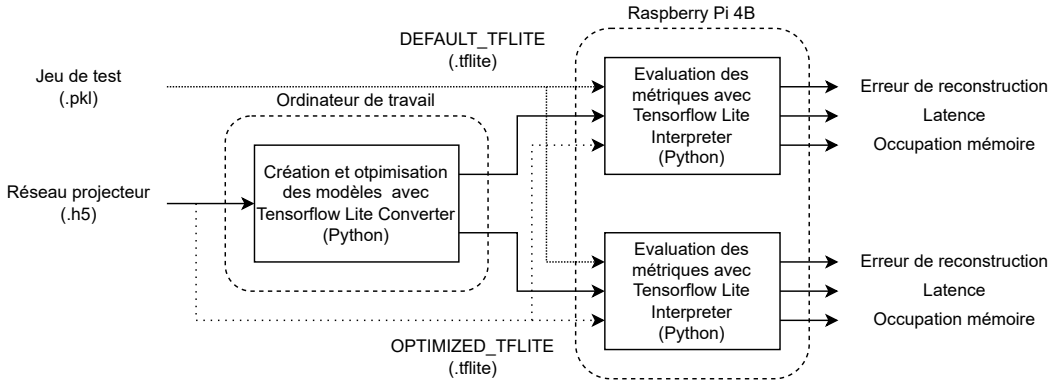


FIGURE 4.2 – Flot d'implémentation logicielle des réseaux siamois

Les résultats sont présentés dans le tableau 4.4. Nous pouvons constater que la valeur de MRSE de l'implémentation DEFAULT_TFLITE est nulle, car il n'y a pas eu de quantification des coefficients, alors que la MRSE de l'implémentation OPTIMIZED_TFLITE est proche de 0.01 % à cause de la quantification 8-bits. En revanche, nous pouvons remarquer que l'implémentation OPTIMIZED_TFLITE a une occupation mémoire beaucoup

3. À titre indicatif, la taille mémoire du modèle du réseau siamois Tensorflow (.h5) est de 4 619 664 octets et celui du réseau projecteur Tensorflow (.h5) est de 1 546 760 octets. De plus, la latence du réseau projecteur (.h5), estimée à l'aide de la fonction **predict_step**, est de 14.41 ms.

plus restreinte, ainsi qu'une latence plus faible. D'ailleurs, seule cette implémentation respecte les contraintes d'occupation mémoire introduites précédemment. Enfin, les deux implémentations peuvent être caractérisées comme des implémentations lentes, même si l'implémentation OPTIMIZED_TFLITE (937 inférences par seconde) est proche d'une implémentation modérée.

Modèle	Taille	Latence	MRSE
DEFAULT_TFLITE	1 527 180 octets	1.757 ms	0.0000 %
OPTIMIZED_TFLITE	385 472 octets	1.067 ms	0.0105 %

TABLE 4.4 – Evaluation des métriques pour l'implémentation logicielle des réseaux siamois

4.2.3 Implémentation matérielle des réseaux siamois avec hls4ml et Vivado HLS

Le flot de l'implémentation matérielle des réseaux siamois, présenté en figure 4.3, est basé sur hls4ml, qui converti le modèle de référence (.h5) en un code C/C++, ensuite synthétisé par Vivado HLS en une architecture matérielle à l'aide d'un langage de description matérielle intermédiaire comme VHDL ou Verilog.

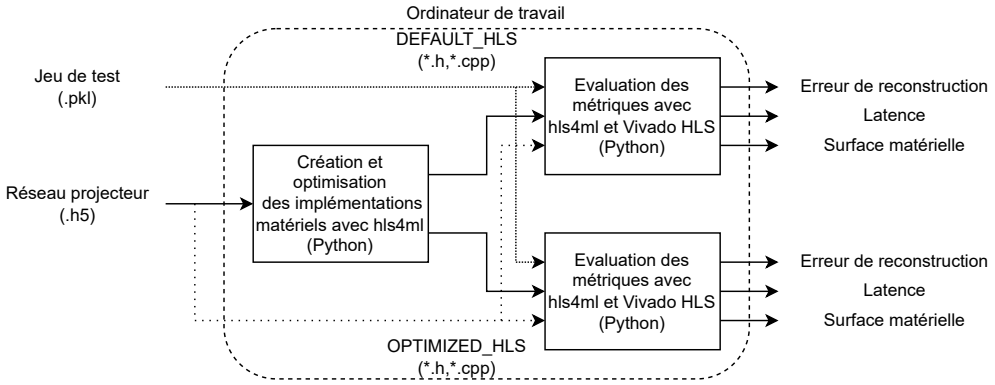


FIGURE 4.3 – Flot d'implémentation matérielle des réseaux siamois

Par défaut, hls4ml cherche à paralléliser complètement un réseau de neurones et réalise une quantification en virgule fixe sginée : ap_fixed<16, 6> [153]. Les coefficients d'un modèle sont ainsi quantifiés en 16 bits avec 6 bits de partie entière et 10 bits de partie fractionnaire. Cependant, il est possible d'agir sur l'implémentation, notamment sur le

niveau de parallélisation de chaque couche à l'aide du paramètre *ReuseFactor*[4], la stratégie d'optimisation (minimisant les ressources ou la latence) ou la quantification utilisée pour chaque couche. Pour notre implémentation, nous avons utilisé une stratégie d'optimisation minimisant l'utilisation des ressources (*Resource*) et permettant d'utiliser des *ReuseFactor* différents de 1 [152].

Lors de nos tests préliminaires, nous avons essayé de réaliser une implémentation du projecteur totalement parallèle, i.e *ReuseFactor* est égal à 1 pour chaque couche. Cependant, nous avons constaté que Vivado HLS n'était pas capable de déplier complètement le réseau de neurones de projection, en raison du niveau de parallélisme trop important. Ainsi, nous avons réalisé deux implémentations différentes, appelée DEFAULT_HLS et OPTIMIZED_HLS. D'une part, l'implémentation DEFAULT_HLS correspond à une implémentation totalement repliée, i.e complètement séquentielle utilisant la valeur maximale du *ReuseFactor* pour chaque couche. D'autre part, l'implémentation OPTIMIZED_HLS correspond à une implémentation partiellement dépliée dont les valeurs de *ReuseFactor* sont décrites dans le tableau 4.5. Ces valeurs de *ReuseFactor* ont été choisies de manière empirique pour minimiser la latence sans que le niveau de parallélisation n'empêche Vivado HLS de synthétiser l'architecture, à cause du trop grand niveau de parallélisme.

Nom de la couche	DEFAULT_HLS	OPTIMIZED_HLS
Couche convolutive 1	350	70
Couche convolutive 2	35 000	700
Couche dense	345 600	5 400

TABLE 4.5 – Valeurs des *ReuseFactor* pour l'implémentation matérielles des réseaux siamois

Les métriques pour les deux implémentations sont présentées dans le tableau 4.6. Premièrement, nous pouvons remarquer que les métriques de reconstruction des deux implémentations ne sont pas acceptables. En effet, une valeur de MRSE de 43.11 % signifie que l'erreur de reconstruction quadratique est de l'ordre de la norme au carré de la projection du modèle de référence. Pour expliquer ce phénomène, nous allons utiliser la figure 4.4 obtenue à l'aide d'un outil de profilage de réseau de neurones intégré à hls4ml [152]. Cette figure montre que la quantification en virgule fixe utilisée (boîtes grises) ne couvre pas les variations importantes des coefficients du modèle (boîtes à moustaches). Cela pourrait notamment s'expliquer par la régularisation L_2 utilisée lors de la phase

4. Le *ReuseFactor* d'une couche, permet de savoir le nombre de fois qu'un multiplieur est réutilisé.

d'apprentissage et détaillée en section 3.2.2.4. De plus, nous pouvons constater que pour l'implémentation DEFAULT_HLS, les valeurs de latence et de d'II sont bien supérieures aux valeurs acceptables. En revanche, elles sont acceptables pour l'implémentation OPTIMIZED_HLS, puisqu'il est possible de réaliser environ 1 162 inférences par seconde, ce qui correspondrait à une implémentation modérée. Enfin, les deux implémentations ne sont pas acceptables au niveau de la surface matérielle, car elles nécessitent trop de ressources par rapport à celles disponibles dans une Zedboard et décrite dans le tableau 4.3. Cela pourrait notamment s'expliquer par le fait que *hls4ml* génère une architecture dite *dataflow* à partir d'un modèle, réalisant ainsi une implémentation matérielle des différentes couches. Cependant, ce type d'architectures peut rencontrer des problématiques de passage à l'échelle pour les réseaux de neurones de tailles relativement grandes [156].

Version	BRAM18K	DSP48E	LUT	FF	Latence	II	MRSE
DEFAULT_HLS	1 800 (642 %)	3 (1 %)	82 688 (77 %)	94 919 (178 %)	965 ms	896 ms	43.11 %
OPTIMIZED_HLS	805 (287 %)	117 (53 %)	146 193 (137 %)	228 162 (428 %)	908 μs	860 μs	43.11 %

TABLE 4.6 – Occupation des ressources pour les implémentations des réseaux siamois pour le RFF (100 MHz)

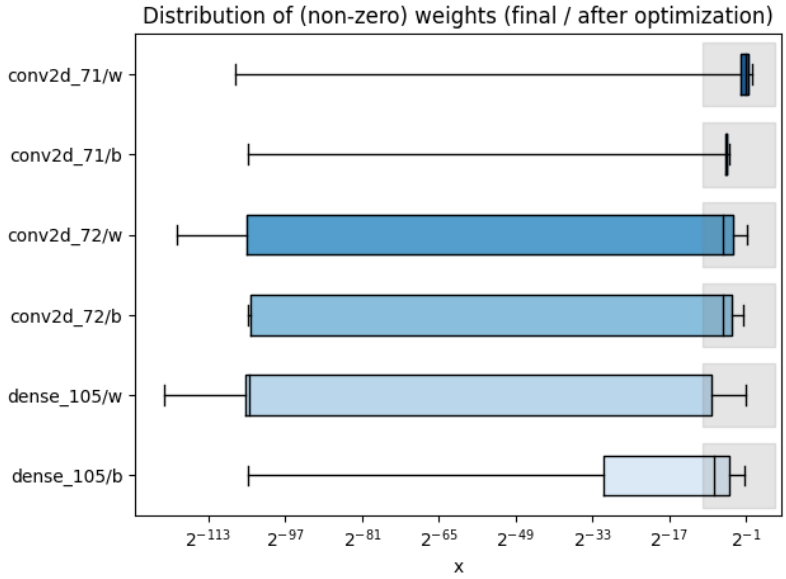


FIGURE 4.4 – Profilage des différentes couches de l'implémentation DEFAULT_HLS par hls4ml

Le tableau 4.8 montre l'implémentation OPTIMIZED_HLS du projecteur de l'architecture légère sur une carte de développement FPGA ZCU104 intégrant un SoC Zynq UltraScale+, une évolution du Zynq-7000 de la Zedboard. Ainsi, nous pouvons constater que l'implémentation ne dépasse pas les ressources de la ZCU104, présentée dans le tableau 4.7, mais que l'erreur de reconstruction est identique, car seule la plateforme cible a été changée lors de la synthèse du code généré par hls4ml. Cette implémentation sur ZCU104 montre la possibilité d'utiliser hls4ml avec des gros réseaux de neurones, à condition d'avoir une plateforme avec assez de ressources.

BRAM18K	DSP48E	LUT	FF
624	1 728	460 800	230 400

TABLE 4.7 – Récapitulatif de la surface matérielle d'une ZCU104

Version	BRAM18K	DSP48E	LUT	FF	Latence	II	MRSE
OPTIMIZED_HLS	609 (97 %)	117 (6 %)	118 365 (25 %)	205 767 (89 %)	908 μs	860 μs	43.11 % -

TABLE 4.8 – Occupation des ressources pour l'implémentation optimisée des réseaux siamois pour le RFF sur ZCU104 (100 MHz)

4.3 Implémentation logicielle et matérielle des *RF eigenfingerprints*

Pour rappel, lors de l'inférence présentée en sous-section 3.3.1, la phase de projection est située après la phase de pré-traitement et elle consiste à projeter le signal aligné $x \in \mathbb{C}^N$ dans le sous-espace de dimension K suivant l'équation suivante :

$$z = U_{proj}^{H}(x - \bar{x}) \tag{4.2}$$

avec :

— $\bar{x} \in \mathbb{C}^N$: le signal moyen.

— $U_{proj} \in \mathbb{C}^{(N \times K)}$: le sous-espace de projection.

— $z \in \mathbb{C}^K$: le signal aligné projeté dans le sous-espace de dimension K.

Dans la section 3.3.2.2, trois classifieurs avaient été introduits et leurs performances avaient été comparées dans la figure 3.28. Nous avions montré que le classifieur n°1 avaient

obtenus les meilleures performances. Dans cette section, nous avons donc choisi d'implémenter la phase de projection de ce classifieur. De plus, et contrairement à la contribution traitant des réseaux siamois, la phase de projection des trois classifieurs contient le même nombre de coefficients, i.e 8 *RF eigenfingerprints*. Il sera ainsi plus facile d'extrapoler pour les autres classifieurs, car seuls les coefficients changent et que cela n'a pas (ou peu) d'influence sur l'implémentation finale.

Comme cela a été évoqué au début de ce chapitre, l'implémentation matérielle des *RF eigenfingerprints* a été réalisée dans un premier temps, puis l'implémentation logicielle a été réalisée en s'inspirant de l'implémentation matérielle. Cependant, la taille des coefficients de la phase de projection a été choisie afin d'être aussi facilement implémentable sur une plateforme logicielle, que matérielle. Par conséquent, la quantification (en virgule fixe) des différents coefficients de la phase de projection est réalisée sur 16 bits, car suffisant vis-à-vis des métriques de reconstruction. La quantification sur 8 bits ayant été écartée lors de tests préliminaires, car les métriques de reconstruction ne correspondaient pas à nos attentes.

De plus, chaque variable complexe est représentée par deux variables, une pour la partie réelle et une pour la partie imaginaire, idem pour les vecteurs et les matrices. En effet, le type **std : :complex** de C++ ne prend pas en compte les types entiers (**int, long, ...**), seulement les types flottants (**float, double, ...**). Cependant, les types entiers sont utilisés dans l'implémentation logicielle pour les calculs en virgule fixe. Ainsi, nous avons fait ce choix pour une meilleure réutilisabilité du code entre l'implémentation matérielle et logicielle.

Pour finir, le classifieur n°1 de la méthode des *RF eigenfingerprints* ayant été obtenu à l'aide de Matlab, nous avons choisi d'exporter les différents résultats de la phase de projection du modèle sous la forme de tableau en représentation fixe ou en représentation flottante selon l'arithmétique utilisée pour l'implémentation.

Les implémentations logicielles et matérielles ont été évaluées à l'aide du jeu de test utilisé dans la section 3.3.2.2 et constitué de 60 signaux alignés.

4.3.1 Étapes préliminaires à l'implémentation matérielle et logicielle des *RF eigenfingerprints* avec Vivado HLS

Les étapes préliminaires nécessaires à l'implémentation matérielle et logicielle sont présentées en figure 4.5. Elles consistent à explorer différentes architectures matérielles.

Les résultats obtenus à la suite de ces étapes préliminaires seront aussi réutilisés pour l'implémentation logicielle des *RF eigenfingerprints*.

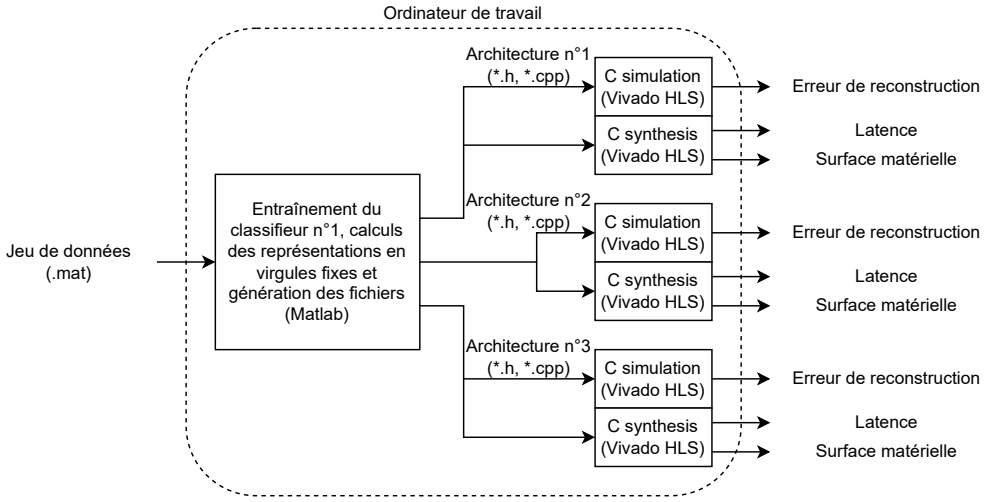


FIGURE 4.5 – Étapes préliminaires pour l'implémentation matérielle des *RF eigenfingerprints*

La première étape de l'implémentation matérielle des *RF eigenfingerprints* consiste à entraîner le classifieur n°1, à déterminer les représentations en virgule fixe et à générer les fichiers *.h et *.cpp nécessaires à la phase suivante (coefficients, jeu de test, ...). L'évaluation de l'amplitude des signaux d'entrées, des coefficients appris par le modèle, ainsi que des résultats de la projection permet de déterminer les différentes représentations en virgules fixes. En effet, pour une représentation en virgule fixe (utilisant le complément à 2) sur N bits avec N_f bits de partie fractionnaire, les valeurs représentables sont comprises entre $[-2^{N-N_f-1}, 2^{N-N_f-1} - 2^{-N_f}]$ avec un pas de 2^{-N_f} [157]. Il est ainsi nécessaire de maximiser la taille de la partie fractionnaire N_f pour limiter le plus possible les troncatures tout en évitant les dépassements. Pour cela, une analyse a été réalisée avec Matlab et elle a déterminé les représentations en virgule fixe suivantes :

— Signaux d'entrée : les signaux d'entrée x_i, ainsi que le signal moyen $\bar{x}$ sont représentés à l'aide d'une représentation en virgule fixe sur 16 bits, avec 14 bits de partie fractionnaire.

— Les *RF eigenfingerprints* : les coefficients contenus dans la matrice U_{proj} sont représentés à l'aide d'une représentation en virgule fixe sur 16 bits, avec 15 bits de partie fractionnaire.

— La projection : les résultats de la projection z_i sont représentés à l'aide d'une

représentation en virgule fixe sur 16 bits, avec 11 bits de partie fractionnaire.

La seconde étape de l'implémentation matérielle consiste à trouver l'architecture qui obtient la meilleure erreur de reconstruction en prenant en compte la métrique de reconstruction MRSE. Nous avons donc exploré plusieurs architectures :

— Architecture 1 : cette architecture, présentée à l'algorithme 3, effectue d'abord une soustraction du préambule et du signal moyen, puis la projette dans l'espace de décision.

— Architecture 2 : cette architecture, présentée à l'algorithme 4, effectue la soustraction entre le préambule et le signal moyen et calcule sa projection au fur à mesure.

— Architecture 3 : cette architecture, présentée à l'algorithme 5, est assez similaire à l'architecture 2. Cependant, elle utilise une astuce qui consiste à décomposer la projection de la manière suivante : $U_{proj}^{H}(x - \bar{x}) = U_{proj}^{H}x + b$ (avec $b = -U_{proj}^{H}\bar{x}$).

Algorithm 3: Pseudocode de l'implémentation FPGA de l'architecture 1

Data: x : signal d'entrée, $\bar{x}$: signal moyen, $U_{proj} = \{u_1, \cdots, u_N\}$: matrice de projection
Result: z : signal projeté
for $i = 1 : K$ **do**
  $z^{(i)} \leftarrow 0$;
end
for $i = 1 : N$ **do**
  $y^{(i)} \leftarrow x^{(i)} - \bar{x^{(i)}}$;
end
for $i = 1 : K$ **do**
  for $j = 1 : N$ **do**
    $z^{(i)} \leftarrow z^{(i)} + u_{j,i}^{*} y^{(j)}$;
  end
end

Dans le tableau 4.9, nous pouvons constater que les trois architectures respectent bien les contraintes de 0.25% vis-à-vis de la métrique de reconstruction [5]. Cependant, les deux premières architectures ont des valeurs de métriques de reconstruction plus faibles que l'architecture 3. De plus, nous pouvons remarquer que l'architecture 2 est la plus rapide, avec une latence de 84.50 μs. Pour la suite de l'implémentation matérielle, nous choisirons donc l'architecture 2 comme architecture à optimiser.

5. Les coefficients du modèle ont été quantifiées à l'aide du type **ap_fixed** de Vivado HLS.

Algorithm 4: Pseudocode de l'implémentation FPGA de l'architecture 2

Data: x : signal d'entrée, $\bar{x}$: signal moyen, $U_{proj} = \{u_1, \cdots, u_N\}$: matrice de projection
Result: z : signal projeté
for $i = 1 : K$ **do**
  $z^{(i)} \leftarrow 0$;
end
for $i = 1 : N$ **do**
  $y^{(i)} \leftarrow x^{(i)} - \bar{x^{(i)}}$;
  for $j = 1 : K$ **do**
    $z^{(j)} \leftarrow z^{(j)} + u^*_{i,j} y^{(i)}$;
  end
end

Algorithm 5: Pseudocode de l'implémentation FPGA de l'architecture 3

Data: x : signal d'entrée, $b = -U^H_{proj}\bar{x}$: le terme de biais, $U_{proj} = \{u_1, \cdots, u_N\}$: matrice de projection
Result: z : signal projeté
for $i = 1 : K$ **do**
  $z^{(i)} \leftarrow b^{(i)}$;
end
for $i = 1 : N$ **do**
  for $j = 1 : K$ **do**
    $z^{(j)} \leftarrow z^{(j)} + u^*_{i,j} y^{(i)}$;
  end
end

Version	BRAM18K	DSP48E	FF	LUT	Latence	II	MRSE
Arch. 1	8 (2 %)	4 (1 %)	277 (0 %)	448 (0 %)	90.76 μs	90.76 μs	0.221 %
Arch. 2	6 (2 %)	4 (1 %)	251 (0 %)	377 (0 %)	84.50 μs	84.50 μs	0.221 %
Arch. 3	6 (2 %)	4 (1 %)	260 (0 %)	334 (0 %)	84.60 μs	84.60 μs	0.227 %

TABLE 4.9 – Comparaison des architectures pour les *RF eigenfingerprints* (80 MHz)

4.3.2 Implémentation matérielle des *RF eigenfingerprints* avec Vivado HLS

Le flot d'implémentation matérielle des *RF eigenfingerprints* est présentée en figure 4.6. Il consiste donc à optimiser [6] l'architecture 2 pour améliorer sa latence. Pour cela, nous avons utilisé des **#pragma** de Vivado HLS, qui agissent comme des directives pour guider l'implémentation. Le pseudo-code de l'optimisation est présenté dans l'algorithme 6. Principalement, deux optimisations ont été réalisées :

— L'initialisation de la projection par des valeurs nulles a été réalisée en parallèle à l'aide de **#pragma HLS unroll**. Ce pragma a donc pour effet d'initialiser chaque valeur de z à 0 en un coup d'horloge.

— La projection a été pipelinée (et parallélisée) à l'aide de **#pragma HLS pipeline**. Ce pragma a pour effet de pipeliner la boucle externe, mais aussi de paralléliser la boucle interne. En effet, comme il est indiqué dans la documentation de Vivado HLS 2019.2 [153], le pragma **#pragma HLS pipeline** parallélise toutes les boucles internes, probablement pour faciliter le *pipeling* de la boucle externe.

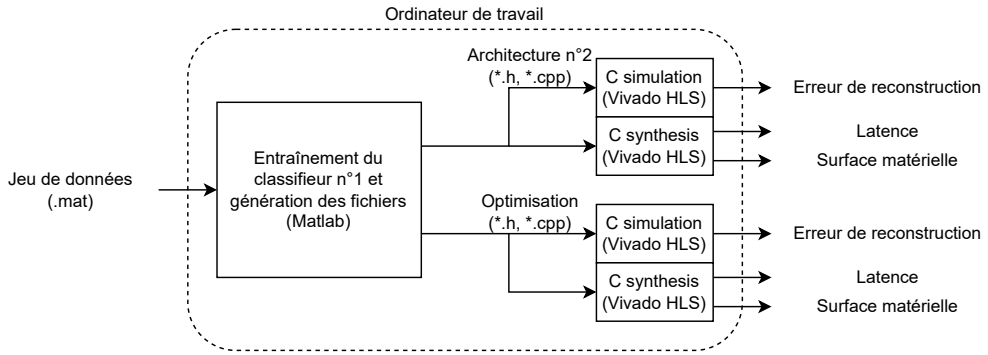


FIGURE 4.6 – Flot d'implémentation matérielle des *RF eigenfingerprints*

En réalité, il est nécessaire de rajouter d'autres **#pragma** comme **#pragma HLS array_partition** pour permettre la parallélisation des traitements. Plus particulièrement, comme on cherche à accéder en parallèle aux différents éléments de z, ainsi qu'aux différentes *RF eigenfingerprints*, il est nécessaire de partitionner ces éléments à l'aide de **#pragma HLS array_partition**. Enfin, les entrées et sorties sont stockées dans des *First-In, First Out* (FIFO) à l'aide de **#pragma HLS interface ap_fifo**.

6. Cette optimisation est beaucoup plus efficace que notre optimisation précédente publiée dans [145] avec une réduction de la latence d'un facteur proche de 6.6.

Algorithm 6: Pseudocode de l'implémentation FPGA

Data: x : signal d'entrée, $\bar{x}$: signal moyen, $U_{proj} = \{u_1, \cdots, u_N\}$: matrice de projection
Result: z : signal projeté
for $i = 1 : K$ **do** `// Boucle d'initialisation`
  #pragma HLS unroll;
  $z^{(i)} \leftarrow 0$;
end
for $i = 1 : N$ **do** `// Boucle de projection externe`
  #pragma HLS pipeline;
  $y^{(i)} \leftarrow x^{(i)} - \bar{x^{(i)}}$;
  for $j = 1 : K$ **do** `// Boucle de projection interne`
    $z^{(j)} \leftarrow z^{(j)} + u^*_{i,j} y^{(i)}$;
  end
end

Dans le tableau 4.10, nous pouvons observer la comparaison entre l'implémentation de l'architecture 2 et de son optimisation. Plus particulièrement, la latence de l'implémentation optimisée a été réduite d'un facteur proche de 25.7, tout en ayant une occupation matérielle acceptable par rapport aux ressources matérielles disponibles sur la Zedboard. Ainsi, l'implémentation matérielle optimisée des *RF eigenfingerprints* est capable de réaliser plus de 300 000 inférences par seconde, ce qui en fait une implémentation rapide. Comme nous l'avions précédemment mentionné dans la section 3.3.3, la complexité de calcul de la phase de projection est en $O(N \times K)$. Cependant, en projetant en parallèle sur chaque axe de z, la latence est en $O(N)$, soit en 263 coups d'horloge.

Version	BRAM18K	DSP48E	FF	LUT	Latence	II	MRSE
Arch. 2	6 (2 %)	4 (1 %)	251 (0 %)	377 (0 %)	84.50 μs	84.50 μs	0.221 %
Opti.	18 (6 %)	32 (14 %)	1 057 (0 %)	1 127 (2 %)	3.28 μs	3.28 μs	0.221 %

TABLE 4.10 – Occupation des ressources pour les implémentations matérielles des *RF eigenfingerprints* (80 MHz)

4.3.3 Implémentation logicielle des *RF eigenfingerprints* en C++

Cette implémentation, présentée en figure 4.7, est basée sur l'architecture 2, comme pour l'implémentation matérielle. Ainsi, pour cette sous-section, nous allons donc comparer trois implémentations logicielles basées sur l'architecture 2 :

— Implémentation 1 : la première implémentation logicielle réalisée est une implémentation en virgule flottante à l'aide du type **float** de C++ (représentation en virgule flottante simple précision).

— Implémentation 2 : la seconde implémentation réalisée est une implémentation en virgule fixe (en utilisant le complément à 2, [157]) basé sur l'architecture 2 précédemment introduite. Cependant, il ne s'agit pas d'une implémentation identique de celle-ci, seulement d'une implémentation présentant des performances similaires en termes d'erreur de reconstruction. De plus, cette implémentation utilise un accumulateur 16 bits pour stocker le calcul de projection.

— Implémentation 3 : la troisième implémentation logicielle réalisée est une implémentation en virgule fixe (en utilisant le complément à 2) cherchant à améliorer l'erreur de reconstruction de l'implémentation 2. Pour cela, elle utilise un accumulateur 32 bits pour calculer le résultat de la projection à la place d'un accumulateur sur 16 bits. Cependant, une fois le calcul de la projection terminé, le résultat stocké dans l'accumulateur est converti dans le format de représentation des résultats de la projection : 16 bits avec 11 bits de partie fractionnaire.

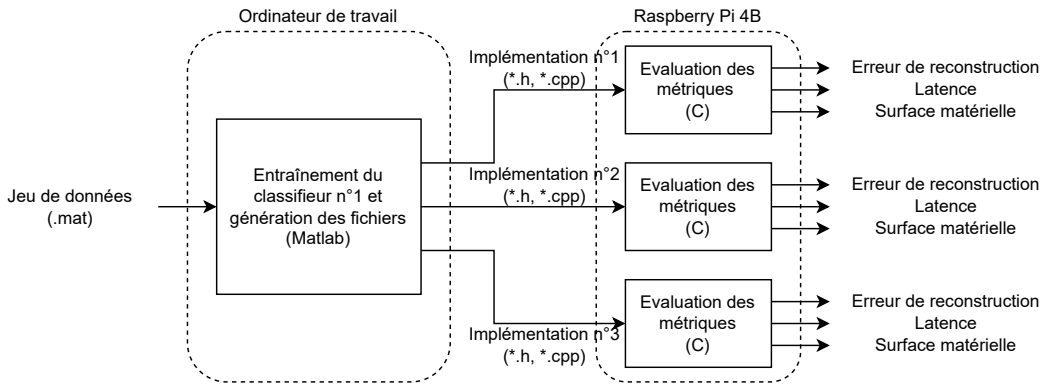


FIGURE 4.7 – Flot d'implémentation logicielle des *RF eigenfingerprints*

Le résultat des différentes implémentations est présenté dans le tableau 4.11. Nous pouvons constater que les erreurs de reconstructions des trois implémentations sont acceptables par rapport à la valeur de 0.25 %. Concernant la complexité mémoire, l'occupa-

tion mémoire [7] des trois implémentations est bien inférieure à la limite des 80 000 octets définie dans le tableau 4.2. Enfin, la latence [8] des trois modèles est inférieure à la limite de 1 ms correspondant à 1 000 inférences par seconde (implémentation modérée). Cependant, nous pouvons remarquer que l'implémentation 1 est la plus rapide avec environ 18 868 inférences par seconde, même si l'implémentation 3 est capable de réaliser environ 18 760 inférences par seconde. Ce phénomène, bien que contre-intuitif, peut s'expliquer facilement par les multiples conversions de type 16 bits vers 32 bits (et vice-versa) qu'il est nécessaire de faire pour éviter les dépassements d'entier lors du calcul des projections dans l'implémentation 2 et 3. Ainsi, comme l'implémentation 3 utilise un accumulateur 32 bits, elle réalise beaucoup moins de conversion de types que l'implémentation 2 qui utilise un accumulateur 16 bits. Pour finir, nous pouvons considérer que ces trois implémentations sont des implémentations petites au sens de l'occupation mémoire, modérées au sens de la complexité de calculs et avec des erreurs de reconstruction satisfaisantes. Il serait tout de même préférable de choisir l'implémentation 3 qui offre un compromis mémoire, latence et erreur de reconstruction intéressant sans utiliser une représentation en virgule flottante, qui nécessiterait un processeur de calculs en virgule flottante.

Implémentation	Taille	Latence	MRSE
Implémentation 1	18 614 octets	53.01 μs	2.136e-08 %
Implémentation 2	9 590 octets	75.38 μs	0.221 %
Implémentation 3	9 590 octets	53.36 μs	2.066e-06 %

TABLE 4.11 – Implémentation logicielle des *RF eigenfingerprints* en C++

4.4 Comparaison des implémentations avec la littérature

Dans cette section, une comparaison des implémentations logicielles et matérielles des méthodes de RFF proposées avec les implémentations de méthodes de RFF de la littérature introduites en section 1.5 sera effectuée. Plus particulièrement, dans l'état de l'art, nous avions présenté trois articles de la littérature [10], [30] et [73]. De plus, nous

7. L'occupation mémoire est estimée à partir du fichier objet (.o), obtenu à partir du fichier (.cpp) et contenant les différents coefficients $(\bar{x}, U_{proj})$.

8. Pour la mesure de latence, nous avons réalisé 1 000 itérations sur le jeu de test, pour avoir une meilleure estimation de la latence.

en profiterons pour comparer nos différentes implémentations entre elles.

4.4.1 Comparaison des implémentations logicielles

À notre connaissance, seul Jian et al. [30] ont réalisé une implémentation logicielle sur CPU de méthodes de RFF. Pour cela, ils implémentent plusieurs réseaux de neurones entraînés sur différents jeux de données (WiFi, ADS-B, ...) et les évaluent sur le CPU d'un Galaxy S10. Les réseaux de neurones ont une architecture commune appelée «ResNet50-1D» [40], comportant initialement aux alentours de 16 à 17 millions de paramètres (selon le jeu de données). Cependant, Jian et al. ont réalisé du *pruning* progressif, consistant à supprimer les paramètres avec une amplitude faible au fur et à mesure de l'entraînement. Ainsi, le plus petit modèle implémenté dans leur article comporte environ 740 000 coefficients, soit deux fois plus que notre architecture légère présentée dans la figure 3.5. L'implémentation la plus rapide sur CPU présentée pour leur article correspond à l'implémentation du réseau de neurones avec 740 000 paramètres et elle obtient un temps d'inférence de 16.31 ms. D'une part, l'implémentation logicielle la plus rapide des réseaux siamois effectue une inférence en 1.07 ms, soit environ 15 fois plus rapidement. D'autre part, l'implémentation logicielle la plus rapide des *RF eigenfingerprints* effectue une inférence en environ 0.053 ms, soit environ 307 fois plus rapidement. Nous pouvons tout de même constater que les implémentations logicielles des *RF eigenfingerprints* sont les plus rapides (< 1 ms) et que leurs occupations mémoire sont extrêmement faibles ($<$ 80 000 octets).

4.4.2 Comparaison des implémentations matérielles

Dans cette partie, nous ne discuterons pas de l'implémentation matérielle des réseaux siamois à cause des problématiques d'implémentations que nous avons expliquées précédemment, i.e une erreur de reconstruction trop importante et une occupation matérielle non compatible avec les ressources disponibles sur Zedboard. Ainsi, nous comparerons uniquement les implémentations matérielles des *RF eigenfingerprints* avec celles de la littérature.

Dans [30], Jian et al. utilisent une carte de développement FPGA ZCU104, comme celle utilisée pour l'implémentation du projecteur de l'architecture légère et décrite dans le tableau 4.8. Ils implémentent les mêmes réseaux de neurones que ceux discutés dans la comparaison des implémentations logicielles. Ainsi, l'implémentation la plus rapide

de leurs réseaux de neurones réalise une inférence en 530 μs. En comparaison, notre implémentation matérielle des *RF eigenfingerprints* la plus rapide effectue une inférence en 3.28 μs, soit 161.5 plus rapidement. Enfin, les auteurs n'ont pas fourni d'évaluation de l'occupation matérielle de leurs implémentations.

Dans [10], Lowder a implémenté son algorithme de RFF sur la partie FPGA (Kintex-7) d'une plateforme SDR USRP X310. Le temps d'inférence de son implémentation est de 137 μs avec un signal à analyser d'une durée de 130 μs. Cependant, son calcul termine seulement 7 μs après la réception du dernier échantillon du signal à analyser. En prenant le cas de 7 μs de latence (meilleur cas), notre implémentation matérielle des *RF eigenfingerprints* la plus rapide effectue une inférence en 3.28 μs, soit 2.138 fois plus rapidement, sinon il s'agirait d'un facteur environ égal à 41.8 pour 137 μs (pire cas). Tout comme pour l'article précédent, les auteurs ne fournissent pas d'évaluation complète de la surface matérielle de leur implémentation.

Dans [73], Peng et al. réalisent l'implémentation de deux réseaux de neurones sur la partie FPGA (Artix-7) d'une carte de développement PYNQ-Z1 intégrant un Zynq-7020. Les auteurs ne présentent pas l'occupation des ressources matérielles de leurs implémentations, mais ils fournissent la latence, le débit, ainsi que la consommation énergétique de chaque implémentation. Ainsi, leur implémentation la plus rapide à une latence de 219 μs, ainsi qu'un débit de 12 192 inférences par seconde (II = 82μs). En comparaison, notre implémentation matérielle optimisée des *RF eigenfingerprints* est toujours la plus rapide avec une latence de 3.28 μs, soit un facteur d'accélération d'environ 66.8, et un nombre d'inférences par seconde d'environ 304 878, soit un facteur d'accélération d'environ 45.

Comme pour l'implémentation logicielle, notre implémentation matérielle optimisée des *RF eigenfingerprints* sur FPGA est la plus rapide avec un nombre d'inférences par seconde supérieur à 100 000. De plus, son occupation matérielle ne dépasse pas les ressources disponibles sur la Zedboard et son erreur de reconstruction est acceptable au seuil des 0.25 %. Ainsi, nous pouvons constater l'avantage, en termes d'implémentation, des *RF eigenfingerprints* sur les réseaux siamois pour le RFF.

Chapitre 5

Perspectives

Dans ce chapitre, nous distinguerons les perspectives de nos travaux selon deux aspects : les perspectives des méthodes de RFF proposées et de leurs implémentations (en sous-section 5.1) et des perspectives plus générales concernant le domaine du RFF dans un contexte IoT (en sous-section 5.2).

5.1 Perspectives des méthodes de RFF proposées

Les perspectives de chaque méthode de RFF proposée dans le chapitre 3 et de leurs implémentations seront présentées indépendamment. Pour chaque méthode, nous présenterons les perspectives selon les trois aspects suivants : son amélioration, l'amélioration de ses implémentations et la possibilité de réentraîner l'espace latent pour cette méthode : l'apprentissage progressif [31]. De plus, nous discuterons de la possibilité d'une expérimentation rendant possible une meilleure comparaison des deux méthodes de RFF proposées.

5.1.1 Perspectives de la méthode 1 : réseaux siamois pour le RFF

Notre première contribution évalue les performances de différents paradigmes d'apprentissage de réseaux siamois, ainsi que de différentes architectures pour le RFF.

Tout d'abord, il serait possible d'améliorer cette méthode en testant de nouveaux paradigmes d'apprentissage comme la *triplet loss* [118], que nous avons déjà évoqué dans la section 3.2.3, ou de nouvelles architectures plus légères comme celles présentées dans [71], [72], voire prenant mieux en compte la nature complexe des signaux [65]. Cela permettrait d'explorer leur impact sur les performances, ainsi que sur les propriétés de la section 3.1.

Il serait aussi possible d'améliorer l'implémentation matérielle en utilisant du *Quantize-Aware Training* à la place de la quantification post-entraînement [152]. Ce type d'apprentissage permet d'apprendre les poids d'un réseau de neurones en prenant en compte la

quantification en virgule fixe, limitant ainsi l'influence des troncatures dans l'implémentation finale. De plus, l'évaluation des implémentations matérielles de réseaux de neurones sur FPGA obtenues avec différents outils comme FINN [158] ou Vitis AI [156] pourrait être une piste intéressante, car permettant d'explorer des stratégies d'implémentation différentes de celles utilisées par hls4ml [148], [156].

Enfin, il serait possible d'explorer des méthodes d'entraînement progressif pour mettre à jour les coefficients d'un réseau siamois. Par exemple, pour prendre en compte l'évolution des empreintes radios dans le temps [64] ou pour l'ajout de nouveaux émetteurs, voire pour s'adapter aux défauts du récepteur. Notamment, Hazra et al. ont montré la possibilité de réentraîner des réseaux siamois dans [69] pour l'ajout de nouveaux émetteurs et ils ont montré une amélioration des performances sans coût de réentraînement prohibitif.

5.1.2 Perspectives de la méthode 2 : *RF eigenfingerprints*

Notre seconde méthode de RFF, appelée *RF eigenfingerprints*, est inspirée des travaux sur les *eigenfaces* en reconnaissance faciale [9].

Une première piste d'amélioration de notre méthode consisterait à s'inspirer des travaux de la littérature ayant découlés des *eigenfaces*. Notamment, dans [135], Belhumeur et al. proposent une méthode, appelée *fisherfaces*, basée sur l'analyse discriminante linéaire. Dans [136], Yang et al. proposent une méthode, appelée *kernelfaces*, basée sur l'astuce du noyau [31], permettant de projeter les données dans un espace non-linéaire, tout en limitant la quantité de calculs nécessaire. Il serait ainsi intéressant d'évaluer l'applicabilité de ces méthodes au RFF et de voir si elles permettent d'améliorer les performances d'authentification sans trop décroître leur scalabilité, ni accroître leur complexité.

L'implémentation matérielle des *RF eigenfingerprints* utilise actuellement deux variables indépendantes pour représenter un nombre complexe. Il serait donc possible d'utiliser le type **std : :complex** de C++, utilisé pour représenter des nombres complexes et compatible avec la représentation en virgule fixe **ap_fixed** utilisée dans Vivado HLS, pour étudier son impact sur la surface matérielle, ainsi que sur la latence de l'implémentation.

Enfin, il serait possible d'utiliser de l'apprentissage progressif pour réapprendre les *RF eigenfingerprints*, notamment en s'inspirant des méthodes d'analyse en composantes principales incrémentales déjà utilisées pour les *eigenfaces* [159], [160]. Ces méthodes permettraient, comme pour les réseaux siamois, de mettre à jour les coefficients servant à la projection pour s'adapter aux différents changements précédemment évoqués.

5.1.3 Comparaison des méthodes de RFF proposées

Le dernier point concernant les perspectives de nos contributions porte sur une comparaison approfondie entre nos deux méthodes. En effet, dans le chapitre 3, les expérimentations réalisées pour chaque méthode de RFF proposée n'ont pas les mêmes conditions. Ainsi, il serait possible de réaliser une expérimentation permettant de mieux comparer ces deux méthodes, notamment en utilisant le modèle d'imperfections présenté dans la section 2.1.2. Cela permettrait de limiter l'influence des conditions expérimentales dans notre comparaison et de mieux évaluer ces deux méthodes vis-à-vis des métriques proposées.

5.2 Perspectives générales

Il semble aussi important de discuter des perspectives générales pour le RFF dans un contexte IoT. Principalement, nous nous focaliserons sur trois points : la transférabilité, l'intégration et la sécurité.

5.2.1 Transférabilité

Cette notion de transférabilité est directement lié aux propriétés introduites dans la section 3.1 et plus particulièrement au concept de scalabilité. En effet, la scalabilité permet de prendre en compte un grand nombre d'émetteurs dans un contexte fortement dynamique. Ainsi, dans le chapitre 3, nous avons proposé deux méthodes pour répondre à cette problématique. Cependant, ces méthodes sont basées sur un pré-entraînement de l'espace latent à partir d'un jeu de données ne contenant pas forcément les émetteurs finaux. Il est donc possible que les défauts du récepteur, présentés dans la section 2.3, aient une influence sur les performances du système final. En effet, il serait possible que le récepteur utilisé pour la capture des signaux servant à entraîner l'espace latent ne soit pas le même que celui utilisé pour l'application finale. Le terme transférabilité venant du transfert de l'espace latent appris sur le récepteur initial à celui de l'application finale.

Pour répondre à cette problématique de transférabilité, nous proposons trois pistes. La première serait d'utiliser des «récepteurs quasi-parfaits», i.e des récepteurs d'assez bonne qualité pour que leurs défauts n'aient pas d'influence sur l'entraînement de l'espace latent. Cependant, il reste à définir cette notion de qualité et son impact sur le coût du récepteur. Une seconde piste consiste à réaliser des corrections des défauts du récepteur comme nous l'avons proposé en section 2.3. Enfin, il serait possible de réentraîner l'espace latent

pour prendre en compte les défauts du récepteur, notamment à l'aide de l'apprentissage progressif, que nous avons évoqué dans la sous-section 5.1.

5.2.2 Intégration

Il est nécessaire de mieux penser l'intégration d'une méthode de RFF dans un appareil IoT, ainsi que dans le protocole visé. En effet, même si des efforts ont été fournis pour étudier l'implémentation de ces méthodes dans un contexte embarqué, il est nécessaire de proposer des métriques, ainsi que des valeurs associées adaptées à notre problématique, notamment en prenant compte les restrictions des appareils IoT et les débits des protocoles visés. Dans un second temps, il serait intéressant d'évaluer les consommations énergétiques et l'occupation mémoire/surface matérielle de ces implémentations, ainsi que leur impact sur le cas d'utilisation finale. Cela permettrait d'évaluer l'intérêt d'utiliser du RFF comme mécanisme de sécurité pour l'IoT. Enfin, il serait nécessaire de proposer des protocoles d'amorçage, permettant d'ajouter un nouvel émetteur légitime dans la base d'empreintes lors de la phase d'enrôlement. Cette problématique n'est d'ailleurs pas spécifique au RFF, car il s'agit une problématique que l'on retrouve aussi dans le monde de l'IoT [161].

5.2.3 Sécurité

Le dernier point pour l'adoption du RFF dans le contexte de l'IoT, et plus généralement pour les réseaux sans-fils, est son niveau de sécurité. En effet, il est considéré comme mal évalué par certains auteurs de la littérature [15], [32].

D'une part, il est nécessaire d'évaluer le niveau de sécurité lors d'une attaque *zero-effort* [162], c'est-à-dire lorsqu'un utilisateur illégitime usurpe l'identité d'un utilisateur légitime sans chercher à modifier son empreinte radio. Cette problématique est largement traitée dans le domaine de la biométrie, notamment via l'utilisation du *False Acceptance Rate* [27], définissant la probabilité qu'un utilisateur illégitime soit considéré comme un utilisateur légitime, ainsi que du *False Rejection Rate*, définissant la probabilité qu'un utilisateur légitime soit considéré comme un utilisateur illégitime. La prise en compte de ces deux métriques amenant à un compromis sécurité-utilisabilité [27].

D'autre part, il semble aussi nécessaire d'évaluer la robustesse des méthodes de RFF aux attaques les ciblant comme le *signal replay* [11], [163], le *feature replay* [11], [163] ou les *adversarial examples* [35], [164], [165]. Ainsi, il faudrait proposer des contre-mesures, ainsi que des métriques pour évaluer l'impact de ces attaques.

Conclusion

Ce manuscrit traite de l'authentification par empreinte radio (RFF) dans le contexte de l'*Internet of Things* (IoT). Plus particulièrement, nous nous sommes intéressés aux propriétés nécessaires à l'utilisation du RFF dans un contexte IoT, puis nous avons proposé deux méthodes de RFF respectant ces propriétés, ainsi que différentes implémentations de ces méthodes.

Dans un premier temps, nous avons introduit trois propriétés d'utilisabilité pour le RFF dans le contexte de l'IoT : l'adaptabilité, la complexité et la scalabilité. Ces propriétés ont été choisies en étudiant les spécificités et les défis de l'IoT exposés dans la littérature, ainsi qu'en analysant les propriétés évoquées dans l'état de l'art du RFF pour l'IoT. D'une part, nous avons proposé l'utilisation de l'apprentissage de représentations pour la prise en compte de la propriété d'adaptabilité. D'autre part, nous avons présenté des métriques et les valeurs associées nous permettant d'évaluer les propriétés de scalabilité et de la complexité d'une méthode de RFF pour l'IoT.

Ensuite, nous avons introduit deux méthodes de RFF adaptées à un contexte IoT : les réseaux siamois pour le RFF et la méthode des *RF eigenfingerprints*. La première méthode se focalise sur les propriétés d'adaptabilité et de scalabilité, même si nous avons montré que certaines architectures de réseaux siamois étaient compatibles avec les contraintes de complexité préalablement introduites. À l'inverse de la première méthode, la seconde se focalise sur les propriétés d'adaptabilité et de complexité, mais nous avons aussi réussi à montrer les propriétés de scalabilité de cette méthode. Ces deux méthodes permettent de présenter un compromis intéressant entre scalabilité et complexité, que nous avons exploité pour leurs implémentations.

Enfin, nous avons étudié l'implémentation de ces méthodes. Plus particulièrement, nous avons fait la différence entre l'implémentation d'une méthode de RFF dans un réseau IoT et son implémentation logicielle ou matérielle sur un appareil IoT. Pour l'implémentation d'une méthode de RFF dans un réseau IoT, nous avons proposé trois cas d'applications dépendant de différentes topologies de réseaux IoT, ainsi que du type d'appareil sur lequel la méthode de RFF est exécutée, puis nous avons positionné les méthodes proposées par rapport à ces différents cas d'applications. Pour l'implémentation d'une méthode

de RFF au sein d'un IoT, nous avons présenté pour chaque méthode de RFF différentes implémentations logicielles sur CPU et matérielles sur FPGA, en les évaluant à l'aide de plusieurs métriques.

Du fait de leur scalabilité, les réseaux siamois pour le RFF semblent être plus adaptés à être implémentés sur des appareils qui ont peu de restrictions en termes de puissance de calculs, mais qui doivent gérer un grand nombre d'appareils, comme une passerelle IoT. À l'inverse, grâce à sa faible complexité, la méthode des *RF eigenfingerprints* semble être plus adaptée à être implémentée sur des appareils qui ne communiquent pas avec beaucoup d'appareils, mais qui ont des ressources de calculs limitées, comme un objet IoT (capteur, actionneur, ...).

Bien que les méthodes de RFF proposées semblent être bien adaptées pour être utilisées dans un contexte IoT, différentes pistes d'améliorations ont été proposés, portant par exemple sur l'amélioration de la méthode ou de leurs implémentations logicielles et matérielles. De plus, la fin du manuscrit est dédiée à la présentation de perspectives plus globales, qu'il semble nécessaire d'étudier pour l'adoption massive du RFF dans un contexte IoT. Ainsi, même si le RFF semble être une piste intéressante pour assurer l'authenticité des messages ou pour la détection d'intrusions dans un réseau IoT, il reste des progrès à faire pour s'assurer de sa viabilité. Par exemple, il serait nécessaire de mieux évaluer le niveau de sécurité des méthodes de RFF, notamment à l'aide de métriques plus adaptées par rapport à celles actuellement utilisées.

Chapitre 6

Liste des publications

Publications dans une conférence

[128] **L. Morge-Rollet**, D. Le Jeune, F. Le Roy et R. Gautier, «Siamese Network on I/Q Signal for RF Fingerprinting», dans *Conference on Artificial Intelligence for Defense* 2020 (CAID 2020), 2020

[166] G. De Broglie, **L. Morge-Rollet**, D. Le Jeune, F. Le Roy, C. Roland, C. Canaff et J.-P. Diguet, «New Methods for Fast Detection for Embedded Cognitive Radio», dans *Asia-Pacific Signal and Information Processing Association Annual Summit and Conference* 2022 (APSIPA 2022), 2022

[167] C. Slimani, **L. Morge-Rollet**, L. Lemarchand, D. Espes, F. Le Roy, J. Boukhobza, «Characterizing Intrusion Detection Systems On Heterogeneous Embedded Platforms», dans *26th Euromicro Conference on Digital System Design* (DSD'2023), 2023

Publications dans un journal

[145] **L. Morge-Rollet**, F. Le Roy, D. Le Jeune, C. Canaff et R. Gautier, «RF eigenfingerprints, an Efficient RF Fingerprinting Method in IoT Context», dans *MDPI Sensors*, 2022

[168] **L. Morge-Rollet**, D. Le Jeune, F. Le Roy, C. Canaff et R. Gautier, «Drone Detection and Classification Using Physical-Layer Protocol Statistical Fingerprint», dans *MDPI Sensors*, 2022

[97] **L. Morge-Rollet**, D. Le Jeune, F. Le Roy, C. Canaff et R. Gautier, «From Modeling to Sensing of Micro-Doppler in Radio Communications», dans *MDPI Remote Sensing*, 2022

Annexe A

Influence de défauts sur les *RF eigenfingerprints*

Le but de cette annexe est de présenter l'influence de différents défauts non pris en compte dans la sous-section 3.3.2.1 sur les *RF eigenfingerprints*, notamment sur le nombre de descripteurs appris, ainsi que leur explicabilité. Plus particulièrement, nous avons choisi d'étudier l'impact du décalage fréquentiel, du bruit de phase ainsi que de multi-trajets.

Pour chacun de ces défauts, deux expérimentations sont réalisées. La première expérimentation consiste à étudier uniquement l'influence du défaut en question sur le nombre de vecteurs propres sélectionnés. La seconde expérimentation consiste à étudier l'impact du défaut sur les *RF eigenfingerprints* à l'aide d'un modèle d'imperfections spécifique et de comparer les résultats obtenus avec ceux présentés en section 2.1.2.

En plus de montrer l'influence des défauts sur la méthode des *RF eigenfingerprints*, cette annexe montre l'intérêt de l'utilisation d'un modèle d'imperfections pour l'étude des propriétés d'une méthode de RFF.

A.1 Influence du décalage fréquentiel

Nous avons réalisé une première expérimentation consistant à évaluer l'influence du décalage fréquentiel sur le nombre de descripteurs appris. En effet, comme nous l'avons préalablement expliqué, l'analyse en composantes principales ou la décomposition en valeurs singulières sont assez sensibles aux sources de variations [137]. Cependant, dans le cas du classifieur n°1, le décalage fréquentiel n'a pas été corrigé et le nombre de descripteurs appris pourrait potentiellement être plus important que dans le cas du classifieur n°2.

Les paramètres de cette expérimentation sont les suivants :

— $f_0 = 2.45$ GHz : la fréquence centrale.

— $F_e = 5$ MHz : la fréquence d'échantillonnage.

— $\delta f_{ppm} = \{0, 0.22, 20, 200\}$ ppm : les différentes valeurs de la précision fréquentielle.
— $f_{max}^{(k)} = \frac{\delta f_{ppm}^{(k)}}{10^6} f_0$: la fréquence maximale correspond à la précision fréquentielle $\delta f_{ppm}^{(k)}$.
— $N_{signals} = 200$: le nombre de signaux pour chaque valeur de précision fréquentielle.
— $SNR = 30$ dB : le rapport signal à bruit.

Le but de cette expérimentation est de mesurer l'impact de la précision fréquentielle δf_{ppm} sur le nombre de descripteurs sélectionnés par l'algorithme 2. Ainsi, pour chaque valeur de la précision fréquentielle ($\delta f_{ppm}^{(k)}$ avec $k \in [\![1; Card(\delta f_{ppm})]\!]$), $N_{signals}$ signaux ($l \in [\![1; N_{signals}]\!]$) sont générés suivant un modèle de signal particulier décrit par l'équation A.1. Puis, un apprentissage des descripteurs est réalisé grâce à une décomposition en valeurs singulières et le nombre de descripteurs utile est déterminé par l'algorithme 2, qui est présenté en sous-section 3.3.1.

$$\tilde{r}_{(l)}^{(k)}(t) = \tilde{s}(t) e^{j2\pi f_{max}^{(k)} (\frac{2\times(l-1)}{N_{signals}} - 1)t} + \tilde{n}^{k,l}(t) \tag{A.1}$$

avec $\tilde{n}^{k,l}(t)$: le bruit additif blanc gaussien complexe pour la réalisation l lié à la précision fréquentielle $\delta f_{ppm}^{(k)}$.

Il est possible d'observer, sur la figure A.1, que le nombre de valeurs propres avec une amplitude importante augmentent en fonction de la précision fréquentielle. De même, sur la figure A.2 qui présente le nombre de descripteurs sélectionnés par l'algorithme 2 en fonction de la précision fréquentielle, il est possible de constater une augmentation similaire.

Nous avons réalisé une seconde expérimentation, basée sur le modèle d'imperfections présenté en figure A.3 et reprenant les défauts du modèle de la figure 3.21, mais en y ajoutant un décalage fréquentiel propre à chaque classe. Les paramètres du modèle d'imperfections sont les suivants :

— Décalage I/Q :
  — $A_I \sim U(-0.01, 0.01)$
  — $A_Q \sim U(-0.01, 0.01)$
— Déséquilibre I/Q :
  — $\epsilon \sim U(-0.01, 0.01)$
  — $U \sim U(\frac{-\pi}{32}, \frac{\pi}{32})$
— Amplificateur de puissance (AM/AM) [1] :

1. Les paramètres a_2 et a_3 sont négatifs et créent une compression de la forme d'onde.

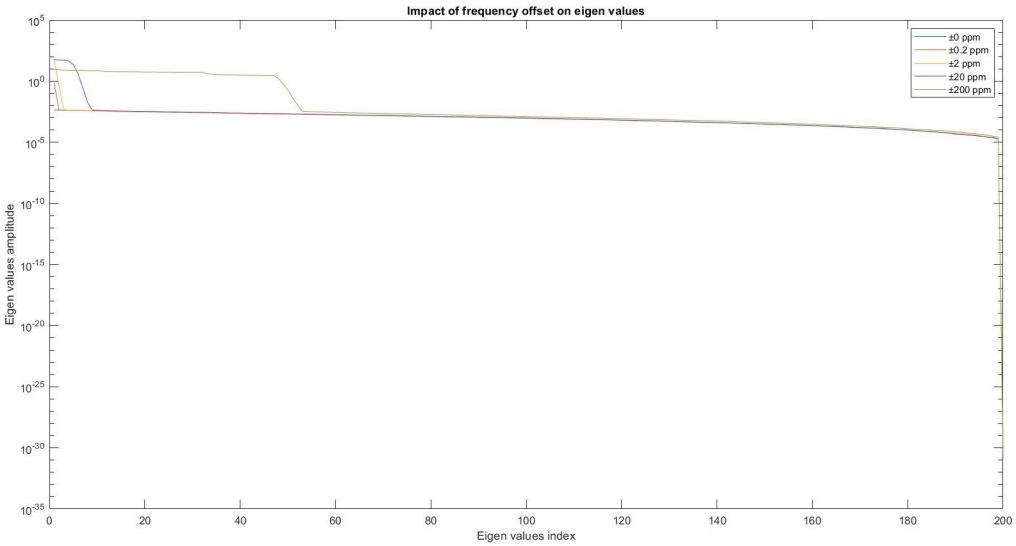


FIGURE A.1 – Évolution des valeurs propres des *RF eigenfingerprints* en fonction de la précision fréquentielle

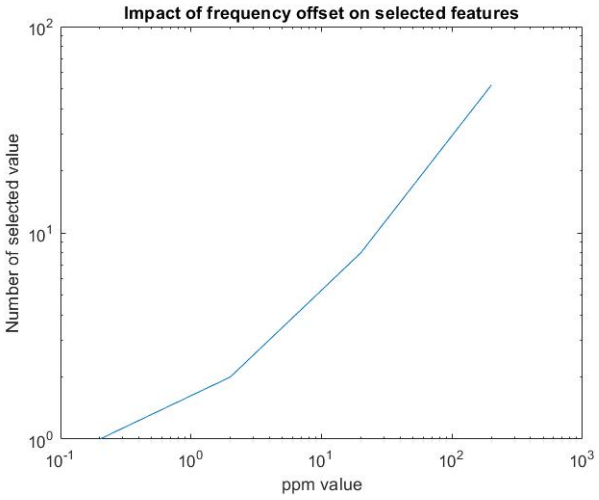


FIGURE A.2 – Évolution du nombre de *RF eigenfingerprints* en fonction de la précision fréquentielle

- $K = 3$
- $a_1 = 1$
- $a_2 \sim -U([-27dB, -33dB])$
- $a_3 \sim -U([-45dB, -55dB])$

— Décalage fréquentiel :

- $f_0 = 2.45$ GHz

— $\delta f_{ppm} = 2$ ppm
— $\Delta f \sim U(-4\,900, 4\,900)$

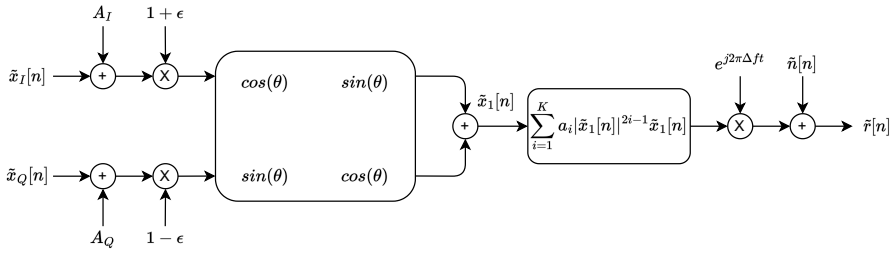


FIGURE A.3 – Modélisation d'imperfections pour les *RF eigenfingerprints* (avec le décalage fréquentiel)

Premièrement, il est possible constater que le signal moyen, présenté en figure A.4, n'est pas similaire au préambule comme c'était le cas dans la figure 3.24. Il est possible aussi d'observer, sur la figure A.5, que huit descripteurs ont été sélectionnés, ce qui correspond à cinq descripteurs de plus que pour la figure 3.23. De plus, les descripteurs appris et présentés en figure A.6 sont beaucoup moins interprétables que ceux présentés en figure 3.25. En effet, ces descripteurs correspondent principalement aux effets du décalage fréquentiel. Cependant, comme on peut le voir sur la figure A.7, les classes sont beaucoup plus séparables dans le sous-espace de projection appris que pour les données du modèle d'imperfections précédent. Ainsi, cela corrobore ce qui a été présenté dans les résultats de l'expérimentation présentée dans la sous-section 3.3.2, c'est-à-dire que prendre en compte le décalage fréquentiel nécessite plus de descripteurs et les rend moins interprétables. En revanche, cela permet une meilleure séparabilité des classes dans l'espace de sous-projection et apporte donc un gain en performance de classification, à condition que le décalage fréquentiel de chaque classe soit constant.

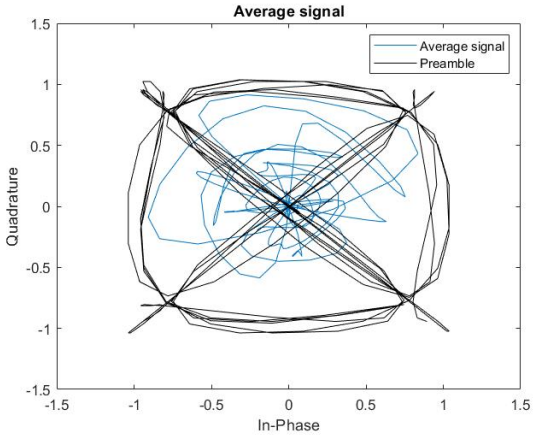


FIGURE A.4 – Signal moyen des *RF eigenfingerprints* (en présence d'un décalage fréquentiel)

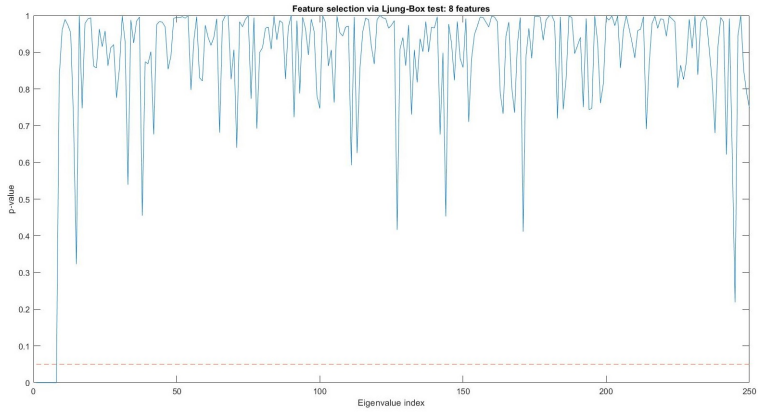


FIGURE A.5 – *p-values* des *RF eigenfingerprints* (pour le décalage fréquentiel)

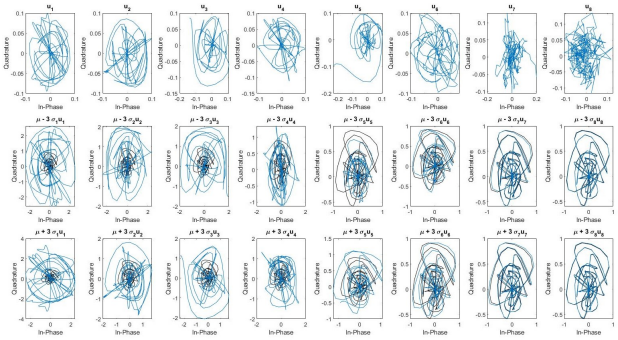


FIGURE A.6 – Visualisation des *RF eigenfingerprints* (pour le décalage fréquentiel)

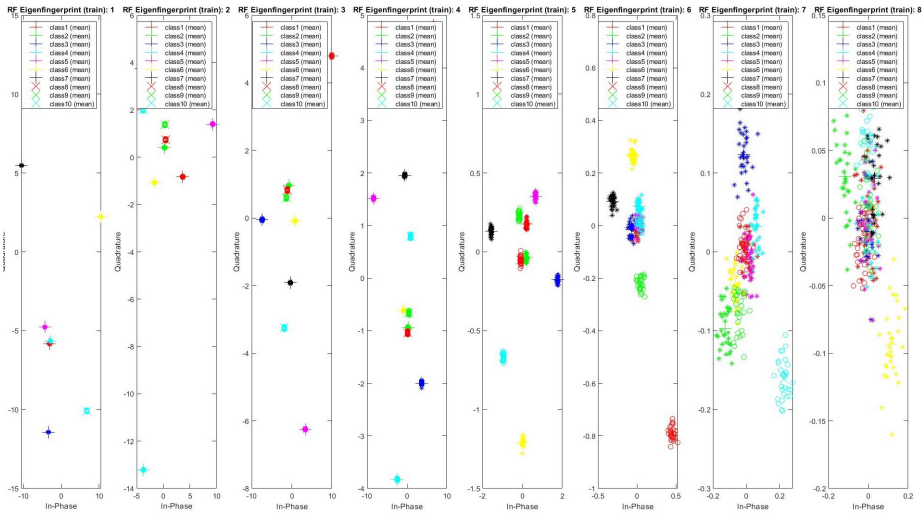


FIGURE A.7 – Projection des données dans le sous-espace des *RF eigenfingerprints* (pour le décalage fréquentiel)

A.2 Influence du bruit de phase

Nous avons réalisé une première expérimentation consistant à évaluer l'influence du bruit de phase sur le nombre de descripteurs appris. En effet, tout comme pour le décalage fréquentiel, le niveau du bruit de phase peut influencer le nombre de descripteurs appris. Les paramètres de cette expérimentation sont les suivants :

- $Level = \{-80, -85, -90, -95, -100\}$ dB : le niveau du bruit de phase
- $N_{signals} = 200$: le nombre de signaux pour chaque valeur de niveau de bruit de phase
- $SNR = 30$ dB : le rapport signal à bruit

Le but de cette expérimentation est de mesurer l'impact du niveau de bruit de phase sur le nombre de descripteurs sélectionnés par l'algorithme 2. Ainsi, pour chaque valeur de niveau de bruit de phase ($Level^{(k)}$ avec $k \in [\![1; Card(Level)]\!]$) , $N_{signals}$ signaux ($l \in [\![1; N_{signals}]\!]$) sont générés suivant un modèle de signal particulier [2] et décrit par l'équation A.2. Puis, un apprentissage des descripteurs est réalisé grâce à une décomposition en valeurs singulières et le nombre de descripteurs utiles est déterminé par l'algorithme 2.

$$\tilde{r}_{(l)}^{(k)}(t) = \tilde{s}(t)e^{j\phi^{k,l}(t)} + \tilde{n}^{k,l}(t) \tag{A.2}$$

2. Pour cette expérience, nous avons utilisé la fonction Matlab **comm.PhaseNoise('Level, Level$^{(k)}$, 'FrequencyOffset', 2000)**

avec :

— $\phi^{k,l}(t)$: le bruit de phase pour la réalisation l liée au niveau de bruit de phase $Level^{(k)}$.

— $\tilde{n}^{k,l}(t)$: le bruit additif blanc gaussien complexe pour la réalisation l liée au niveau de bruit de phase $Level^{(k)}$.

Il est possible d'observer, sur la figure A.8, que le nombre de valeurs propres avec une amplitude importante augmentent en fonction du niveau de bruit de phase. De même, sur la figure A.9 qui présente le nombre de descripteurs considérés sélectionnés par l'algorithme 2, il est possible de constater une augmentation similaire. Cependant, cet impact est moindre que pour le décalage fréquentiel.

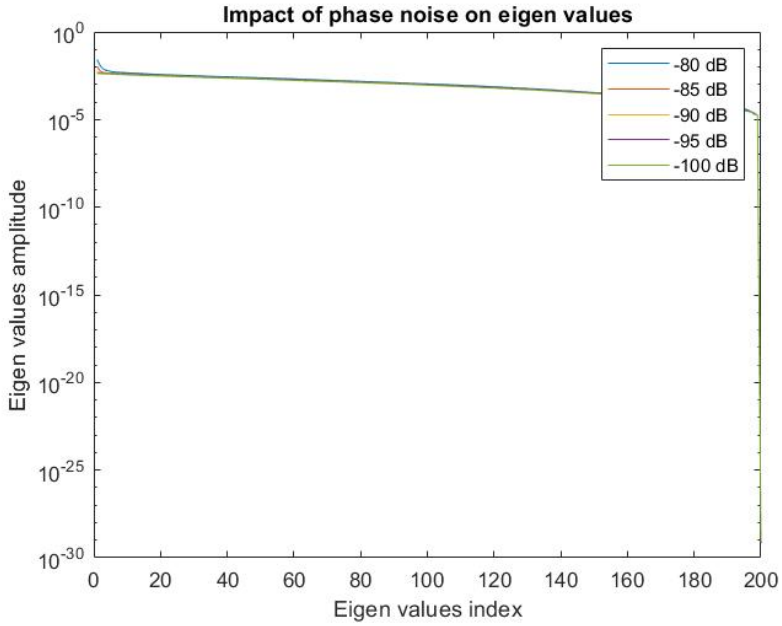


FIGURE A.8 – Évolution des valeurs propres des *RF eigenfingerprints* en fonction du niveau du bruit de phase

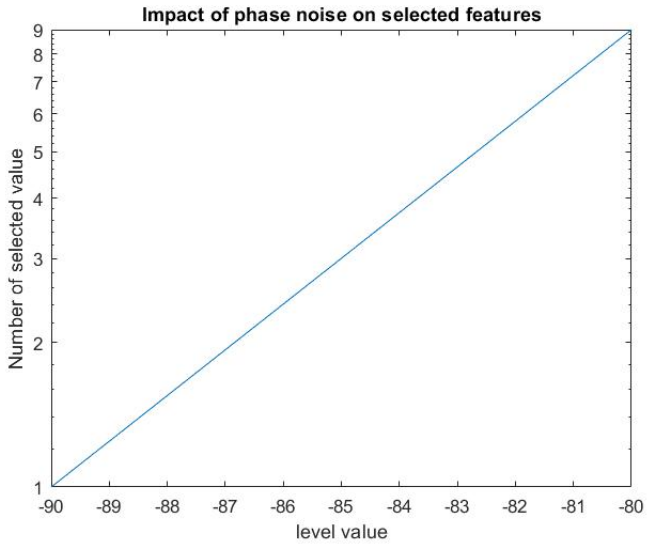


FIGURE A.9 – Évolution du nombre de *RF eigenfingerprints* en fonction du niveau du bruit de phase

Nous avons réalisé une seconde expérimentation basée sur le modèle d'imperfections présenté en figure A.10 et reprenant les défauts du modèle de la figure 3.21, mais en y ajoutant un bruit de phase. Les paramètres du modèle d'imperfections sont les suivants :

- Décalage I/Q :
  - $A_I \sim U(-0.01, 0.01)$
  - $A_Q \sim U(-0.01, 0.01)$
- Déséquilibre I/Q :
  - $\epsilon \sim U(-0.01, 0.01)$
  - $U \sim U(\frac{-\pi}{32}, \frac{\pi}{32})$
- Amplificateur de puissance (AM/AM) [3] :
  - $K = 3$
  - $a_1 = 1$
  - $a_2 \sim -U([-27dB, -33dB])$
  - $a_3 \sim -U([-45dB, -55dB])$
- Bruit de phase [4] :
  - $Level = -85$ dB

Premièrement, il est possible de constater, sur la figure A.11, que le signal moyen est

3. Les paramètres a_2 et a_3 sont négatifs et créent une compression de la forme d'onde.

4. Pour cette expérience, nous avons utilisé la fonction Matlab **comm.PhaseNoise('Level, -85, 'FrequencyOffset', 2000)**.

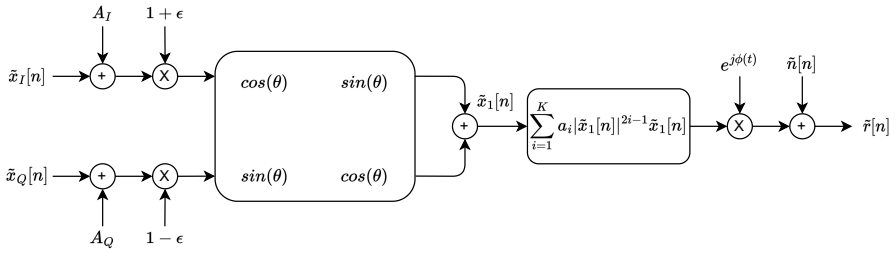


FIGURE A.10 – Modélisation d'imperfections pour les *RF eigenfingerprints* (avec le bruit de phase)

assez similaire à celui de la figure 3.24, car comme $\mathbb{E}(\tilde{s}(t)) = \tilde{s}(t)$ et $\mathbb{E}(e^{j\phi(t)}) = 1$ donc $\mathbb{E}(\tilde{s}(t)e^{j\phi(t)}) = \tilde{s}(t)$. Il est possible aussi d'observer, sur la figure A.12, que six descripteurs ont été sélectionnés, ce qui correspond à trois descripteurs de plus que pour la figure 3.23. Nous pouvons remarquer, sur la figure A.6, que les trois premiers descripteurs appris sont similaires à ceux présentés en figure 3.25. Cependant, les trois derniers descripteurs sont beaucoup moins interprétables, car il semble correspondre au bruit de phase. Il est intéressant de remarquer, que le nombre de descripteurs appris, présentés en figure 3.23, est trois et que le nombre de descripteurs appris pour un niveau de bruit de phase (*Level*) de -85 dB dans la figure A.9 est de trois. Cela pourrait s'expliquer par le fait que si $\phi(t)$ a une amplitude relativement faible, alors $e^{j\phi(t))} \approx 1 + j\phi(t)$ (en faisant un développement limité à l'ordre 1). Ainsi, cela paraîtrait logique que les trois premiers descripteurs correspondent à des défauts similaires à ceux présentés en 3.25 et que les trois derniers correspondent à du bruit de phase. De plus, il est possible d'observer, sur la figure A.14, que les classes sont relativement bien séparables sur les trois premiers axes, similairement à ce que nous avions pu observer en figure 3.26, mais que les trois derniers axes, correspondant au bruit de phase, n'apportent aucune séparabilité des classes. Il est possible d'en conclure que le bruit de phase peut faire augmenter le nombre de descripteurs considéré comme informatif (au sens du test Q de Ljung-Box), mais que cela n'apporte pas de gain en termes de séparabilité.

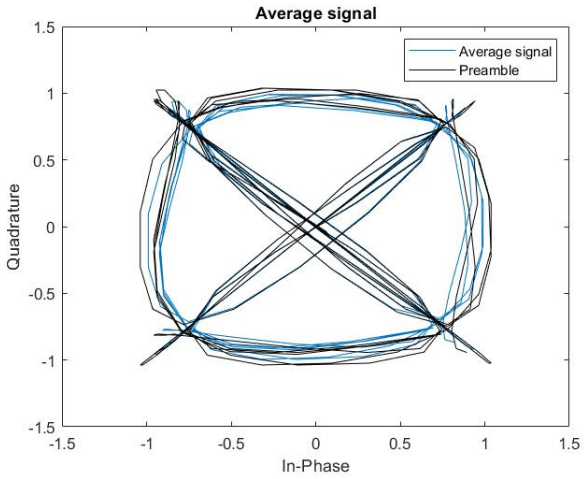


FIGURE A.11 – Signal moyen des *RF eigenfingerprints* (en présence de bruit de phase)

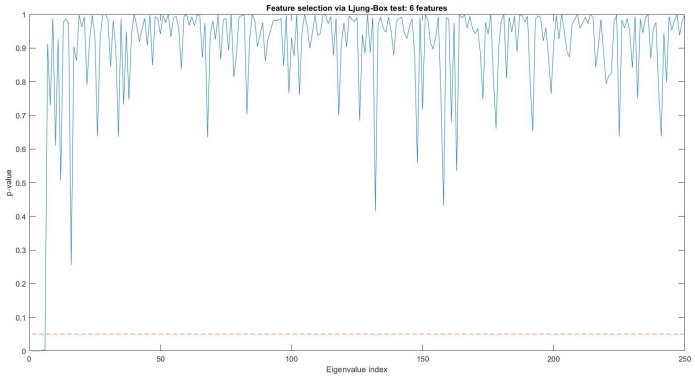


FIGURE A.12 – *p-values* des *RF eigenfingerprints* (pour le bruit de phase)

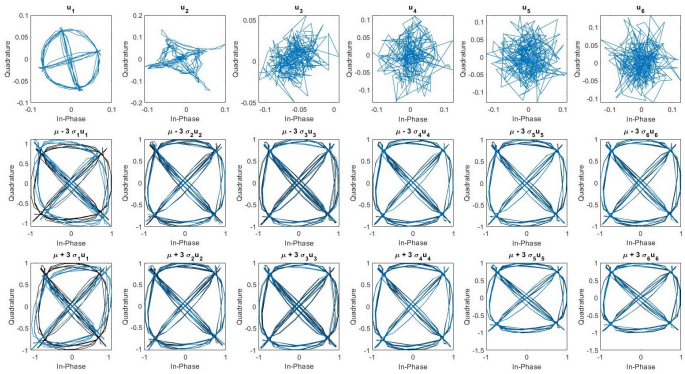


FIGURE A.13 – Visualisation des *RF eigenfingerprints* (pour le bruit de phase)

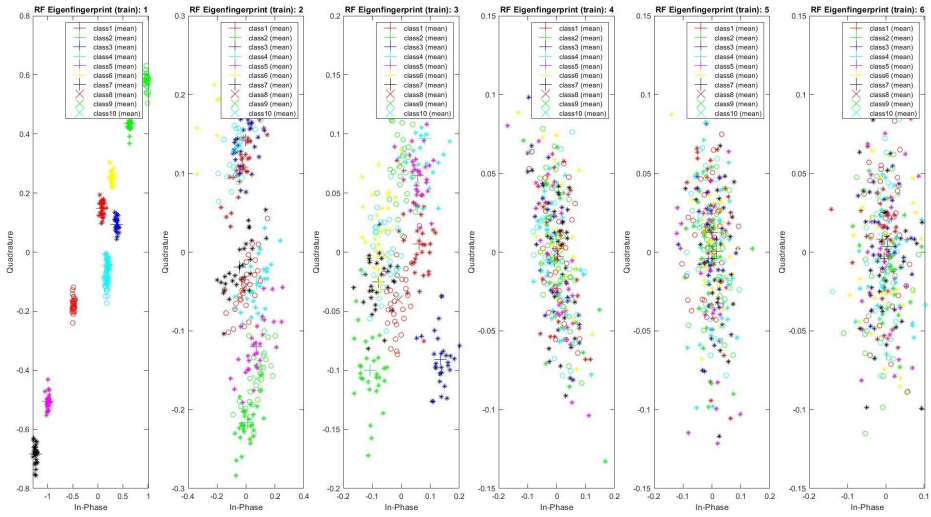


FIGURE A.14 – Projection des données dans le sous-espace des *RF eigenfingerprints* (pour le bruit de phase)

A.3 Influence de multi-trajets

Nous avons réalisé une première expérimentation consistant à évaluer l'influence de multi-trajets sur le nombre de descripteurs appris. Les paramètres de cette expérimentation sont les suivants :

— $N_{path} = \{1, 3, 5, 7, 9\}$: le nombre de trajets.
— $h_1 = 1$: le coefficient du premier trajet.
— $h_{i(i>1)} \sim \mathcal{CN}(0, 1)$: les coefficients du canal [169].
— $N_{signals} = 200$: le nombre de signaux pour chaque valeur de précision fréquentielle.
— $SNR = 30$ dB : le rapport signal à bruit.

Le but de cette expérimentation est de mesurer l'impact du nombre de trajet N_{path} sur le nombre de descripteurs sélectionnés par l'algorithme 2. Ainsi, pour chaque valeur du nombre de multi-trajets ($N_{path}^{(k)}$ avec $k \in [\![1; Card(N_{path})]\!]$), $N_{signals}$ signaux ($l \in [\![1; N_{signals}]\!]$) sont générés suivant un modèle de signal particulier décrit par l'équation A.3. Puis, un apprentissage des descripteurs est réalisé grâce à une décomposition en valeurs singulières et le nombre de descripteurs utile est déterminé par l'algorithme 2.

$$\tilde{r}_{(l)}^{(k)}(t) = h^{(k,l)}(t) * \tilde{s}(t) + \tilde{n}^{k,l}(t) \tag{A.3}$$

avec :

— $\tilde{n}^{k,l}(t)$: la réalisation d'un bruit additif blanc gaussien complexe pour la réalisation

l liée au nombre de trajet $N_{path}^{(k)}$.

— $h^{(k,l)}(t) = \sum_{i=0}^{N_{path}^{(k)}-1} h_i^(k,l)\delta(t-kT_e)$: le canal (filtre à réponse impulsionnelle finie).

— $h_i^(k,l)$: le coefficient assosié au $i+1$-ème trajet et à la réalisation l liée au nombre de trajets $N_{path}^{(k)}$.

— $T_e = 1/Fe$: la période d'échantillonnage.

— F_e : la fréquence d'échantillonnage.

Il est possible d'observer, sur la figure A.15, que le nombre de valeurs propres avec une amplitude importante augmentent linéairement en fonction du nombre de trajets. De même, sur la figure A.16 qui présente le nombre de descripteurs considérés comme nécessaires par l'algorithme 2, il est possible de constater une augmentation similaire. Cette augmentation linéaire est logique, puisqu'il s'agit d'une combinaison linéaire du signal $y(t) = \sum_{i=1}^{N_{Path}} h_i \tilde{s}(t-kTe)$.

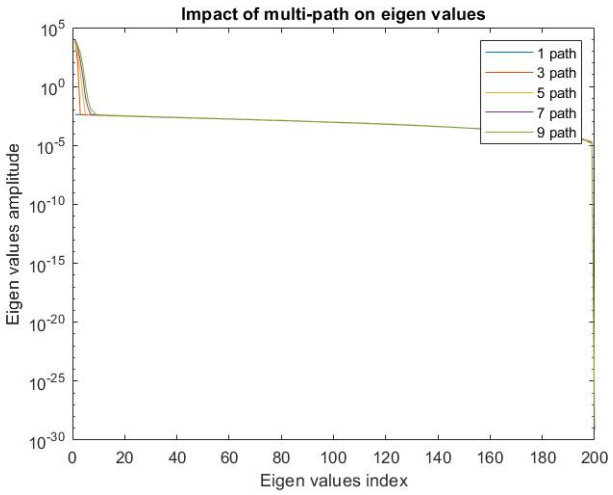


FIGURE A.15 – Évolution des valeurs propres des *RF eigenfingerprints* en fonction du nombre de trajet

Nous avons réalisé une seconde expérimentation basée sur le modèle d'imperfections présenté en figure A.17, reprenant les défauts du modèle de la figure 3.21, mais en y ajoutant des multi-trajets propres à chaque classe. Les paramètres du modèle d'imperfections sont les suivants :

— Décalage I/Q :
  — $A_I \sim U(-0.01, 0.01)$
  — $A_Q \sim U(-0.01, 0.01)$

— Déséquilibre I/Q :

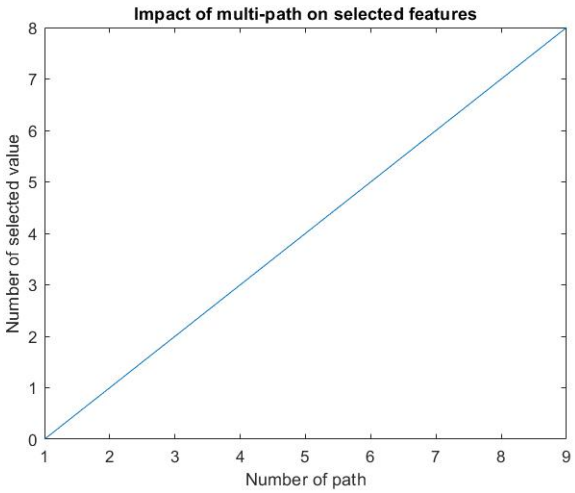


FIGURE A.16 – Évolution du nombre de *RF eigenfingerprints* en fonction du nombre de trajet

- $\epsilon \sim U(-0.01, 0.01)$
- $U \sim U(\frac{-\pi}{32}, \frac{\pi}{32})$

— Amplificateur de puissance (AM/AM) [5] :
- $K = 3$
- $a_1 = 1$
- $a_2 \sim -U([-27dB, -33dB])$
- $a_3 \sim -U([-45dB, -55dB])$

— Multi-trajets :
- N_{path}=3
- h_1=1
- $h_{i(i>1)} \sim \mathcal{CN}(0, 1)$

Premièrement, il est possible peut constater sur la figure A.18, que comme pour le cas de la fréquence (cf. figure A.4), le signal moyen appris n'est pas similaire au préambule. Il est possible aussi d'observer, sur la figure A.19, que sept descripteurs ont été sélectionnés, ce qui correspond à quatre descripteurs de plus que pour la figure 3.23. Tout comme pour le cas du décalage fréquentiel, les descripteurs appris et présentés en figure A.20 sont beaucoup moins interprétables que ceux présentés en figure 3.25. En effet, ces descripteurs correspondent principalement aux effets des multi-trajets. Cependant, les classes sont beaucoup plus séparables dans le sous-espace de projection présenté en figure A.21, que

5. Les paramètres a_2 et a_3 sont négatifs et créent une compression de la forme d'onde.

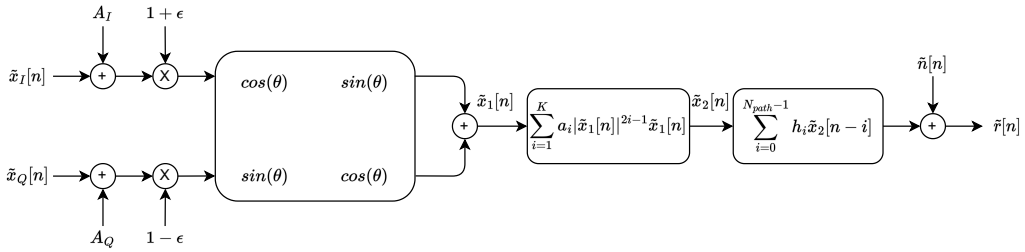


FIGURE A.17 – Modélisation d'imperfections pour les *RF eigenfingerprints* (avec les multi-trajets)

pour les données du précédent modèle d'imperfections. Ainsi, même si prendre en compte les effets des multi-trajets nécessite plus de descripteurs et les rend moins interprétables. En revanche, cela permet une meilleure séparabilité des classes dans l'espace de sous-projection et apporte donc un gain en performance, à condition que le canal soit invariant dans le temps et différent pour chaque émetteur.

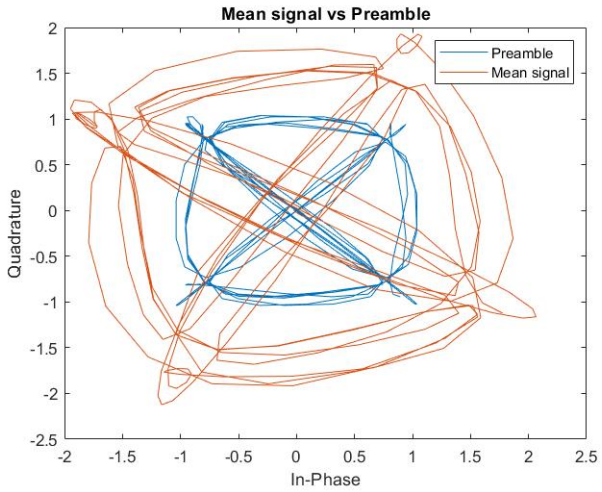


FIGURE A.18 – Signal moyen des *RF eigenfingerprints* (en présence de multi-trajets)

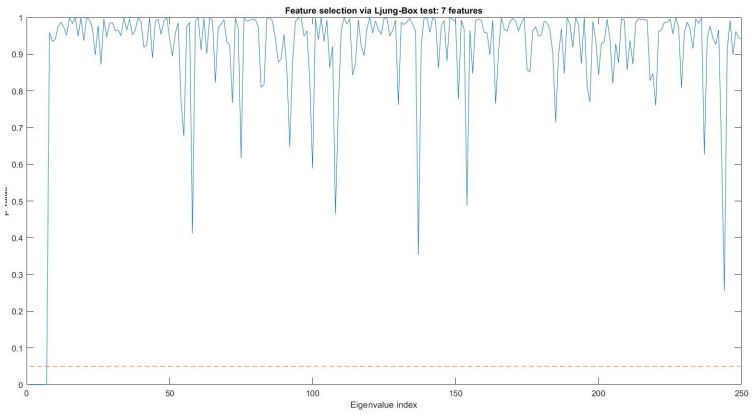


FIGURE A.19 – *p-values* des *RF eigenfingerprints* (pour les multi-trajets)

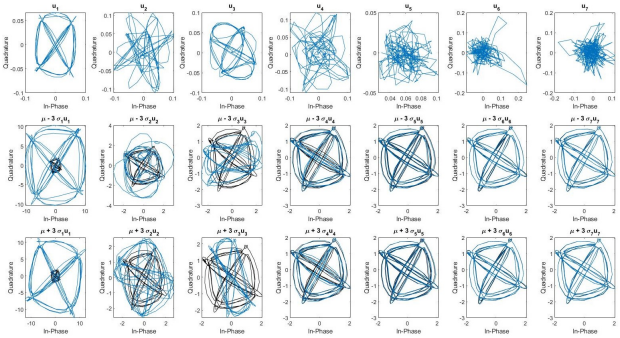


FIGURE A.20 – Visualisation des *RF eigenfingerprints* (pour les multi-trajets)

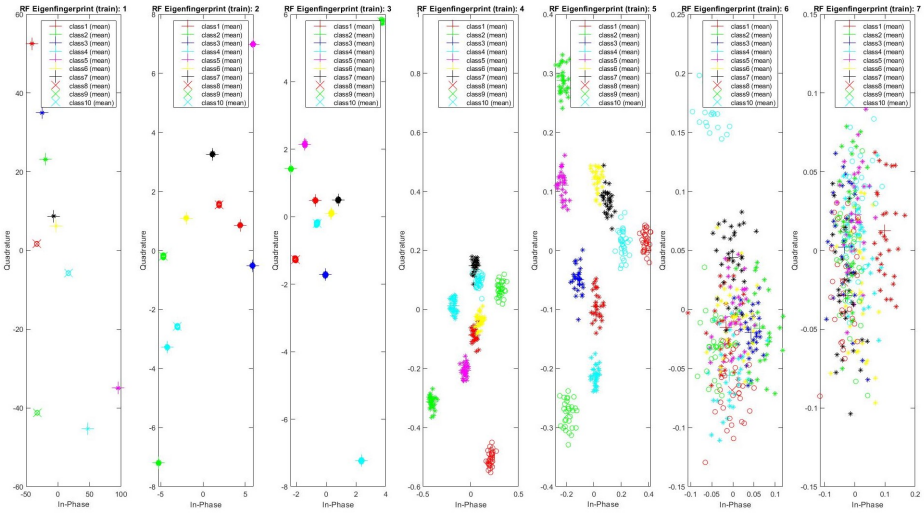


FIGURE A.21 – Projection des données dans le sous-espace des *RF eigenfingerprints* (pour les multi-trajets)

BIBLIOGRAPHIE

[1] V. Brik, S. Banerjee, M. Gruteser et S. Oh, « Wireless Device Identification with Radiometric Signatures », in *Proceedings of the 14th ACM international conference on Mobile computing and networking*, 2008, p. 116-127.

[2] Oracle, *Qu'est-ce que l'Internet of Things (IoT) (dernière consultation le 25/06/2023)*, 2014. adresse : `https://www.oracle.com/fr/internet-of-things/what-is-iot/`.

[3] M. I. Naas, « Placement des données de l'internet des objets dans une infrastructure de fog », thèse de doct., MathSTIC, 2019.

[4] Ericsson, *Ericsson Mobility Report (dernière consultation le 25/06/2023)*, 2023. adresse : `https://www.ericsson.com/49dd9d/assets/local/reports-papers/mobility-report/documents/2023/ericsson-mobility-report-june-2023.pdf`.

[5] K. K. Patel et S. M. Patel, « Internet of Things-IOT : Definition, Characteristics, Architecture, Enabling Technologies, Application and Future Challenges », t. 6, 2016.

[6] J. Bromley, J. W. Bentz, L. Bottou et al., « Signature Verification Using A "Siamese" Time Delay Neural Network », *Advances in neural information processing systems*, t. 6, 1993.

[7] S. Chopra, R. Hadsell et Y. Le Cun, « Learning a similarity metric discriminatively, with application to face verification », in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2005.

[8] M. Turk et A. Pentland, « Face recognition using eigenfaces », in *IEEE Conference on Computer Vision and Pattern Recognition*, 1991, p. 586-587.

[9] M. Turk et A. Pentland, « Eigenfaces for recognition », *Journal of Cognitive Neuroscience*, t. 3, p. 71-86, 1991.

[10] W. M. Lowder, « Real-Time RF-DNA Fingerprinting of ZigBee Devices Using a Software-Defined Radio with FPGA Processing », mém. de mast., Air Force Institute of Technology, 2015.

[11] B. Danev, H. Luecken, S. Capkun et K. M. E. Defrawy, « Attacks on physical-layer identification », in *WiSec '10*, 2010, p. 89-98.

[12] O. Ureten et N. Serinken, « Wireless security through RF fingerprinting », *Canadian Journal of Electrical and Computer Engineering*, t. 32, p. 27-33, 2007.

[13] L. Peng, A. Hu, J. Zhang, Y. Jiang, J. Yu et Y. Yan, « Design of a Hybrid RF Fingerprint Extraction and Device Classification Scheme », *IEEE Internet of Things Journal*, t. 6, p. 349-360, 2019.

[14] N. Soltanieh, Y. Norouzi, Y. Yang et N. C. Karmakar, « A Review of Radio Frequency Fingerprinting Techniques », *IEEE Journal of Radio Frequency Identification*, t. 4, p. 222-223, 2020.

[15] B. Danev, D. Zanetti et S. Capkun, « On physical-layer identification of wireless devices », *ACM Computing Surveys*, t. 45, p. 29, 2012.

[16] F. Zhuo, Y. Huang et J. Chen, « Radio Frequency Fingerprint Extraction of Radio Emitter Based on I/Q Imbalance », *Procedia computer science*, p. 472-477, 2017.

[17] A. Jagannath, J. Jagannath et P. S. P. V. Kumar, « A comprehensive survey on radio frequency (RF) fingerprinting : Traditional approaches, deep learning, and open challenges », *Computer Networks*, t. 219, 2022.

[18] Y. Zou, J. Zhu, X. Wang et L. Hanzo, « A Survey on Wireless Security : Technical Challenges, Recent Advances, and Future Trends », *Proceedings of the IEEE*, t. 104, p. 1727-1765, 2016.

[19] X. S. Zhou, X. Wang et M. Bloch, *Best Readings in Physical-Layer Security*. IEEE ComSoc, 2018.

[20] L. Bai, L. Zhu, J. Liu, J. Choi et W. Zhang, « Physical Layer Authentication in Wireless Communication Networks : A Survey », *Journal of Communications and Information Networks*, t. 5, p. 237-264, 2020.

[21] N. Xie, Z. Li et H. Tan, « A Survey of Physical-Layer Authentication in Wireless Communications », *IEEE Communications Surveys & Tutorials*, t. 23, p. 282-310, 2021.

[22] K. Zeng, K. Govindan et P. Mohapatra, « Non-cryptographic authentication and identification in wireless networks », *IEEE Wireless Communications*, t. 17, p. 56-62, 2010.

[23] Q. Xu, R. Zheng, W. Saad et Z. Han, « Device Fingerprinting in Wireless Networks : Challenges and Opportunities », *IEEE Communications Surveys & Tutorials*, t. 18, p. 94-104, 2016.

[24] M. Katagi, *Lightweight Cryptography for the Internet of Things (dernière consolutation le 25/03/2023)*, 2011. adresse : `https://www.iab.org/wp-content/IAB-uploads/2011/03/Kaftan.pdf`.

[25] J. Jean et T. Peyrin, « La cryptographie symétrique à bas coût : Comment protéger des données avec peu de ressources », *MISC Hors Série - Sécurité des objets connectés*, t. 15, p. 104-115, 2017.

[26] M. El-Abed, R. Giot, B. Hemery, J.-J. Schwartzmann et C. Rosenberger, « Towards the Security Evaluation of Biometric Authentication Systems », *International journal of engineering and technology*, p. 315-320, 2011.

[27] R. M. Bolle, J. H. Connell, S. Pankanti, N. K. Ratha et A. W. Senior, *Guide to Biometrics*. Springer Professional Computing, 2004, p. 3-30, 70, 90-91.

[28] S. Riyaz, K. Sankhe, S. Ioannidis et K. Chowdhury, « Deep Learning Convolutional Neural Networks for Radio Identification », *IEEE Communications Magazine*, t. 56, p. 146-152, 2018.

[29] K. Sankhe, M. Belgiovine, F. Zhou, S. Riyaz, S. Ioannidis et K. R. Chowdhury, « ORACLE : Optimized Radio clAssification through Convolutional neuraL nEtworks », 2019.

[30] T. Jian, Y. Gong, Z. Zhan et al., « Radio Frequency Fingerprinting on the Edge », *IEEE Transactions on Mobile Computing*, t. 21, p. 4078-4093, 2021.

[31] A. Géron, *Hands-on Machine Learning with Scikit-Learn, Keras and Tensorflow - 2nd edition*. O'Reilly, 2019, p. 1-174, 213-276, 331-374, 445-524.

[32] R. M. Gerdes, « Physical layer identification : methodology, security, and origin of variation », thèse de doct., Iowa State University, 2011.

[33] P. Robyns, E. Marin, W. Lamotte, P. Quax, D. Singelée et B. Preneel, « Physical-Layer fingerprinting of LoRa devices using supervised and zero-shot learning », in *ACM Conference on Security and Privacy in Wireless and Mobile Networks*, 2017, p. 58-63.

[34] E. Ozturk, F. Erden et I. Guvenc, « RF-based low-SNR classification of UAVs using convolutional neural networks », in *arXiv*, 2020.

[35] S. Karunaratne, E. Krijestorac et D. Cabric, « Penetrating RF Fingerprinting-based Authentication with a Generative Adversarial Attack », in *IEEE International Conference on Communications*, 2021, p. 1-6.

[36] K. I. Talbot, P. R. Duley et M. H. Hyatt, « Specific Emitter Identification and Verification », *Technology Review Journal*, t. 113, 2003.

[37] J. Matuszewski, « Specific Emitter Identification », in *International Radar Symposium*, 2008, p. 1-4.

[38] N. Serinken, K. Ellis et E. Lavigne, *An Evaluation of the MoTron TxID-1 Transmitter Fingerprinting System (dernière consultation le 25/06/2023)*, 1997. adresse : `https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=ecbcba18aeff4072d35aad4bf38b8946d440d25e`.

[39] M. Sun, L. Zhang, J. Bao et Y. Yan, « RF fingerprint extraction for GNSS anti-spoofing using axial integrated Wigner bispectrum », *Journal of Information Security and Applications*, t. 35, p. 51-54, 2017.

[40] T. Jian, B. C. Rendon, E. Ojuba et al., « Deep Learning for RF Fingerprinting : A Massive Experimental Study », *IEEE Internet of Things Magazine*, t. 6, p. 50-57, 2020.

[41] N. T. Nguyen, G. Zheng, Z. Han et R. L. Zheng, « Device fingerprinting to enhance wireless security using nonparametric Bayesian method », in *IEEE INFOCOM*, 2011, p. 1404-1412.

[42] K. B. Rasmussen et S. Capkun, « Implications of radio fingerprinting on the security of sensor networks », in *2007 Third International Conference on Security and Privacy in Communications Networks and the Workshops*, 2007, p. 331-340.

[43] V. Lakafosis, A. Traille, H. Lee et al., « RF Fingerprinting Physical Objects for Anticounterfeiting Applications », *IEEE Transactions on Microwave Theory and Techniques*, t. 59, p. 504-514, 2011.

[44] L. Peng, A. Hu, Y. Jiang, Y. Yan et C. Zhu, « A differential constellation trace figure based device identification method for ZigBee nodes », in *International Conference on Wireless Communications and Signal Processing (WCSP)*, 2016, p. 1-6.

[45] T. D. Vo-Huu, T. D. Vo-Huu et G. Noubir, « Fingerprinting Wi-Fi Devices Using Software Defined Radios », in *Proceedings of the 9th ACM Conference on Security & Privacy in Wireless and Mobile Networks*, 2016, p. 3-14.

[46] K. Yang, J. Kang, J. Jang et H.-n. Lee, « Multimodal Sparse Representation-Based Classification Scheme for RF Fingerprinting », *IEEE Communications Letters*, t. 23, p. 867-870, 2019.

[47] I. Mohamed, Y. Dalveren, F. O. Catak et A. Kara, « On the Performance of Energy Criterion Method in Wi-Fi Transient Signal Detection », *MDPI Electronics*, t. 11, 2022.

[48] M. Ezuma, F. Erden, C. K. Anjinappa, O. Ozdemir et I. Guvenc, « Detection and Classification of UAVs Using RF Fingerprints in the Presence of Interference », *IEEE Open Journal of the Communications Society*, t. 1, p. 60-76, 2019.

[49] S. Dolatshahi, A. C. Polak et D. L. Goeckel, « Identification of wireless users via power amplifier imperfections », in *Conference Record of the Forty Fourth Asilomar Conference on Signals, Systems and Computers*, 2010, p. 1553-1557.

[50] A. C. Polak, S. Dolatshahi et D. L. Goeckel, « Identifying Wireless Users via Transmitter Imperfections », *IEEE Journal on Selected Areas in Communications*, t. 29, p. 1469-1479, 2011.

[51] A. C. Polak et D. L. Goeckel, « Wireless Device Identification Based on RF Oscillator Imperfections », in *IEEE International Conference on Acoustic, Speech and Signal Processing*, 2014, p. 2679-2683.

[52] Y. Le Cun, Y. Bengio et G. Hinton, « Deep Learning », *Nature*, t. 521, p. 436-444, 2015.

[53] Q.-y. Wu, C. Feres, D. Kuzmenko et al., « Deep learning based RF fingerprinting for device identification and wireless security », *Electronics Letters*, t. 54, p. 1405-1407, 2018.

[54] K. Sankhe, M. Belgiovine, F. Zhou et al., « No Radio Left Behind : Radio Fingerprinting Through Deep Learning of Physical Hardware Impairments », *IEEE Transactions on Cognitive Communications and Networking*, t. 6, p. 165-178, 2019.

[55] M. Villegas, *Radio-communications numériques - Tome 2*. Dunod, 2007.

[56] J. Palicot, *Radio Engineering - From Shotware to Cognitive Radio*. Wiley, 2011, p. 172-185, 208-210, 229-243.

[57] G. Baudoin, *Communications numériques - Tome 1*. Dunod, 2013, p. 23-59, 242-269, 433-590.

[58] R. Khanzadi, « Modeling and Estimation of Phase Noise in Oscillators with Colored Noise Sources », mém. de mast., Chalmers, 2013.

[59] M. Isaksson, D. Wisell et D. Rönnow, « A Comparative Analysis of Behavioral Models for RF Power Amplifiers », *IEEE Transactions on Microwave Theory and Techniques*, t. 54, p. 348-359, 2006.

[60] H. Ghozlan et G. Kramer, « Models and Information Rates for Wiener Phase Noise Channels », *IEEE Transactions on Information Theory*, t. 63, p. 2376-2393, 2017.

[61] W. Wang, Z. Sun, S. Piao, B. Zhu et K. Ren, « Wireless Physical-Layer Identification : Modeling and Validation », *IEEE Transactions on Information Forensics and Security*, t. 11, p. 2091-2106, 2016.

[62] J. Zhang, R. Woods, M. Sandell, M. Valkama, A. Marshall et J. Cavallaro, « Radio Frequency Fingerprint Identification for Narrowband Systems, Modelling and Classification », *IEEE Transactions on Information Forensics and Security*, t. 16, p. 3974-3987, 2021.

[63] H. Yuan, Z. Bao et A. Hu, « Power Ramped-up Preamble RF Fingerprintsof Wireless Transmitters », *Radioengineering*, t. 20, p. 703-709, 2011.

[64] S. U. Rehman, K. W. Sowerby, S. Alam et I. T. Ardekani, « Radio frequency fingerprinting and its challenges », in *2014 IEEE Conference on Communications and Network Security*, 2014, p. 496-497.

[65] E. Mattei, C. Dalton, A. Draganov et al., « Feature Learning for Enhanced Security in the Internet of Things », 2019, p. 1-5.

[66] G. Shen, J. Zhang, A. J. Marshall et J. Cavallaro, « Towards Scalable and Channel-Robust Radio Frequency Fingerprint Identification for LoRa », *IEEE Transactions on Information Forensics and Security*, t. 17, p. 774-787, 2022.

[67] A. Aghnaiya, Y. Dalveren et A. Kara, « On the Performance of Variational Mode Decomposition-Based Radio Frequency Fingerprinting of Bluetooth Devices », *Sensors*, t. 7, 2020.

[68] E. Uzundurukan, Y. Dalveren et A. Kara, « A Database for the Radio Frequency Fingerprinting of Bluetooth Devices », *Data*, t. 5, 2020.

[69] S. Hazra, T. Voigt et W. Yan, « PLIO : Physical Layer Identification using One-shot Learning », in *2021 IEEE 18th International Conference on Mobile Ad Hoc and Smart Systems (MASS)*, 2021, p. 335-343.

[70] M. S. Kavre, A. Gadekar et Y. Gadhade, « Internet of Things (IoT) : A Survey », in *IEEE Pune Section International Conference*, 2019, p. 1-6.

[71] J. R. G. del Arroyo, B. J. Borghetti et M. A. Temple, « Considerations for Radio Frequency Fingerprinting across Multiple Frequency Channels », *Sensors*, t. 22, 2022.

[72] G. Qing, H. Wang et T. Zhang, « Radio frequency fingerprinting identification for Zigbee via lightweight CNN », *Physical Communication*, t. 44, 2021.

[73] H.-T. Peng, J. Lederman, L. Xu et al., « A Photonic-Circuits-Inspired Compact Network : Toward Real-Time Wireless Signal Classification at the Edge », in *arXiv*, 2021.

[74] DARPA, *Radio Frequency Machine Learning Systems (RFMLS) (dernière consultation le 25/06/2023)*, 2019. adresse : `https://www.darpa.mil/program/radio-frequency-machine-learning-systems`.

[75] F. Restuccia, S. D'oro, A. Al-Shawabka et al., « DeepRadioID : Real-Time Channel-Resilient Optimization of Deep Learning-based Radio Fingerprinting Algorithms », 2019, p. 51-60.

[76] D. Tse, *Fundamentals of radio Communications*. Cambridge University Press, 2005, p. 21-63, 578-585.

[77] I. Scott, *Analogue IQ Error Correction For Transmitters - Off Line Method (dernière consultation 25/06/2023)*, 2007. adresse : `https://vaedrah.angelfire.com/tx_iq_correction.htm`.

[78] F. Cottet, *Traitement des signaux et acquisition des données*. Dunod, 2015, p. 128-130, 287-290.

[79] N. J. Kasdin, « Discrete Simulation of Colored Noise and Stochastic Processes and $1/(f^{a}lpha)$; $PowerLawNoiseGeneration$ », *The Proceedings of the IEEE*, t. 83, p. 802-827, 1995.

[80] M. Windisch, *Estimation and Compensation of I/Q imbalance in Broadband Communications Receivers*. VOGT, 2007, p. 1-38.

[81] Z. Zhu, H. Leung et X. Huang, « Challenges in reconfigurable radio transceivers and application of nonlinear signal processing for RF impairment mitigation », *IEEE Circuits and Systems Magazine*, t. 13, p. 44-65, 2013.

[82] D. P. Bertsekas, *Nonlinear Programming*. Athena Scientific, 2002, p. 8-9.

[83] B. A. D. H. Brandwood, « A complex gradient operator and its application in adaptive array theory », *IEE Proceedings F - Communications, Radar and Signal Processing*, t. 130, p. 11-16, 1983.

[84] M. Joindot et A. Glavieux, *Introduction aux communications numériques*. Dunod, 2007, p. 6-9, 203-206.

[85] S. Kessentini et D. Barchiesi, *Initiation à l'optimisation : métaheuristiques - Problèmes à variables continues*. Ellipses, 2020.

[86] C.-A. Azencott, *Introduction au Machine Learning*. Dunod, 2019, p. 11, 34-35, 39, 44, 52-53, 169-175.

[87] A. Devices, *ADALM-PLUTO - SDR Active Learning Module (dernière consultation le 25/06/2023)*, 2017. adresse : `https://www.analog.com/media/en/news-marketing-collateral/product-highlight/adalm-pluto-product-highlight.pdf`.

[88] *AD9364 Reference Manual (dernière consultation le 25/06/2023)*, 2016. adresse : `https://www.analog.com/media/cn/technical-documentation/user-guides/ad9364_reference_manual_ug-673.pdf`.

[89] J.-M. Brossier et G. Jourdain, « Algorithmes adaptatifs auto-optimisés pour l'égalisation et la récupération de porteuse - Application aux transmissions acoustiques sous-marines », *Traitement du signal*, t. 11, p. 327-336, 1994.

[90] B. Lyonnet, « Traitement du signal pour les communications numériques au travers de canaux radio-mobiles », thèse de doct., Ecole doctorale d'Electronique, Electrotechnique, Automatique et Traitement du Signal, 2011.

[91] A. Mours, « Localisation de cible en sonar actif », thèse de doct., Ecole doctorale Terre, Univers, Environnement, 2017.

[92] R. H. Clarke, « A statistical theory of mobile-radio reception », *The Bell System Technical Journal*, t. 47, p. 957-1000, 1968.

[93] R. H. Clarke et W. L. Khoo, « 3-D Mobile Radio Channel Statistics », *IEEE Transactions on Vehicular Technology*, t. 56, p. 798-799, 1997.

[94] E. Simon, « Estimation de canal et Synchronisation pour les systèmes OFDM en présence de mobilité », thèse de doct., Université de Lille 1 Sciences et Technologies, 2015.

[95] L. Ros, « Traitement du signal pour les communications numériques au travers de canaux radio-mobiles », thèse de doct., Ecole doctorale d'Electronique, Electrotechnique, Automatique et Traitement du Signal, 2015.

[96] V. Chen, F. Li, S.-S. Ho et H. Wechsler, « Micro-Doppler effect in radar : phenomenon, model, and simulation study », *IEEE Transactions on Aerospace and Electronic Systems*, t. 42, p. 2-21, 2006.

[97] L. Morge-Rollet, D. Le Jeune, F. Le Roy, C. Canaff et R. Gautier, « From Modeling to Sensing of Micro-Doppler in Radio Communications », *MDPI Remote Sensing*, t. 14, 2022.

[98] A.C.McCormick et A.K.Nandi, « Cyclostationarity in rotating machine », *Mechanical Systems and Signal Processing*, t. 2, p. 225-242, 1998.

[99] K. Kim, I. A. Akbar, K. K. Bae, J.-S. Urn, C. M. Spooner et J. H. Reed, « Cyclostationary Approaches to Signal Detection and Classification in Cognitive Radio », in *IEEE International Symposium on New Frontiers in Dynamic Spectrum Access Networks*, 2007, p. 212-215.

[100] J. Chen, A. Gibson et J. Zafar, « Cyclostationary spectrum detection in cognitive radios », in *IET Seminar on Cognitive Radio and Software Defined Radios : Technologies and Techniques*, 2008.

[101] D. Mototolea, R. Youssef, E. Radoi et I. Nicolaescu, « Non-Cooperative Low-Complexity Detection Approach for FHSS-GFSK Drone Control Signals », *IEEE Open Journal of the Communications Society*, t. 1, p. 401-412, 2020.

[102] R. Badeau, « Méthode à haute résolution pour l'estimation et le suivi des sinusoïdes modulées. Applications aux signaux de musique. », thèse de doct., Telecom Paris, 2005.

[103] G. Blanchet et M. Charbit, *Signaux et images sous matlab*. Hermes, 2001.

[104] S. G. Mallat et Z. Zhang, « Matching pursuits with time-frequency dictionaries », *IEEE Transactions on Signal Processing*, t. 41, p. 3397-3415, 1993.

[105] I. O. Kennedy, P. Scanlon, F. J. Mullany, M. M. Buddhikot, K. E. Nolan et T. W. Rondeau, « Radio Transmitter Fingerprinting : A Steady State Frequency Domain Approach », in *IEEE 68th Vehicular Technology Conference*, 2008, p. 1-5.

[106] Y. Bengio, A. C. Courville et P. Vincent, « Representation learning : A review and new perspectives », *IEEE transactions on pattern analysis and machine intelligence*, t. 35, p. 1798-1828, 2013.

[107] I. Goodfellow, Y. Bengio et A. Courville, *L'apprentissage profond*. Quantmetry, 2018, p. 27-36, 65-74, 344.

[108] A. Ghasemi, C. Parekh et P. Guinand, « Spectrum Awareness Under Co-Channel Usage Via Deep Temporal Convolutional Networks », in *WinCOMM*, 2020, p. 1-7.

[109] Y. Wang, Q. Yao, J. T.-Y. Kwok et L. M.-s. Ni, « Generalizing from a few examples : A survey on few-shot learning », *ACM computing surveys*, t. 53, p. 1-34, 2020.

[110] Scikit-Learn, *Choosing the right estimator (dernière consultation le 25/06/2023)*, 2013. adresse : `https://scikit-learn.org/stable/tutorial/machine_learning_map/index.html`.

[111] P. Warden, *How many images do you need to train a neural network ? (dernière consultation le 25/06/2023)*, 2017. adresse : `https://petewarden.com/2017/12/14/how-many-images-do-you-need-to-train-a-neural-network/`.

[112] V. Lakshmanan, S. Robinson et M. Munn, *Machine Learning Design Patterns*. O'Reilly, 2020, p. 160-161.

[113] Y. Zhang, N. Suda, L. Lai et V. Chandra, « Hello Edge : Keyword Spotting on Microcontrollers », in *arXiv*, 2017.

[114] A. Chowdhery, P. Warden, J. Shlens, A. G. Howard et R. Rhodes, « Visual Wake Words Dataset », in *arXiv*, 2019.

[115] C. R. Banbury, V. J. Reddi, M. Lam et al., « Benchmarking TinyML Systems : Challenges and Direction », in *arXiv*, 2020.

[116] C. R. Banbury, V. J. Reddi, P. Torelli et al., « MLPerf Tiny Benchmark », in *arXiv*, 2021.

[117] Y. Taigman, M. Yang, M. Ranzato et L. Wolf, « DeepFace : Closing the Gap to Human-Level Performance in Face Verification », in *2014 IEEE Conference on Computer Vision and Pattern Recognition*, 2014, p. 539-546.

[118] F. Schroff, D. Kalenichenko et J. Philbin, « FaceNet : A unified embedding for face recognition and clustering », in *2015 IEEE Conference on Computer Vision and Pattern Recognition*, 2015.

[119] Y. Le Cun, *Quand la machine apprend - La révolution des neurones artificiels et de l'apprentissage profond*. Odile Jacob, 2019, p. 228-231.

[120] S. Rochette et C. Bridon, « Keras, l'outil privilégie des data scientists », t. 106, p. 70-86, 2020.

[121] T. O'Shea, J. Corgan et T. C. Clancy, « Convolutional Radio Modulation Recognition Networks », in *International conference on engineering applications of neural networks*, 2016, p. 213-226.

[122] R. Hadsell, S. Chopra et Y. Le Cun, « Dimensionality Reduction by Learning an Invariant Mapping », in *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2006, p. 815-823.

[123] A. Gresse, M. Quillot, R. Dufour, V. Labatut et J.-F. Bonastre, « Similarity Metric Based on Siamese Neural Networks for Voice Casting », in *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2019, p. 6585-6589.

[124] G. Koch, R. Zemel et R. Salakhutdinov, « Siamese neural networks for one-shot image recognition », in *ICML deep learning workshop*, 2015.

[125] J. Brownlee, *A Gentle Introduction to Transfer Learning for Deep Learning (dernière consultation le 25/06/2022)*, 2017. adresse : https://machinelearningmastery.com/transfer-learning-for-deep-learning/.

[126] A. Ng, *Convolutional Neural Networks - C4W4 (dernière consultation le 25/06/2023)*, 2017. adresse : https://www.youtube.com/watch?v=-FfMVnwXrZ0&list=PL57aAW1NtY5gameWemtP_74odYnQ-ETdA&ab_channel=DeepLearningAI.

[127] Z. L. Langford, L. Eisenbeiser et M. Vondal, « Robust signal classification using siamese networks », in *Proceedings of the ACM Workshop on Wireless Security and Machine Learning*, 2019, p. 1-5.

[128] L. Morge-Rollet, F. Le Roy, D. Le Jeune et R. Gautier, « Siamese Network on I/Q Signal for RF Fingerprinting », in *CAID 2020*, 2020, p. 152-159.

[129] Y. Wu, Z. Sun et G. Yue, « Siamese Network-based Open Set Identification of Communications Emitters with Comprehensive Features », in *2021 6th International Conference on Communication, Image and Signal Processing (CCISP)*, 2021, p. 408-412.

[130] L. Sirovich et M. Kirby, « Low-dimensional procedure for the characterization of human faces », *Journal of Optical Society of America*, t. 43, p. 512-524, 1987.

[131] M. Kirby et L. Sirovich, « Application of the Karhunen-Loeve procedure for the characterization of human faces », *IEEE Transactions on Pattern Analysis and Machine Learning*, t. 12, p. 103-108, 1990.

[132] A. S. Georghiades, P. N. Belhumeur et D. J. Kriegman, « From Few to Many : Illumination Cone Models for Face Recognition under Variable Lighting and Pose », *IEEE Trans. Pattern Anal. Mach. Intell.*, t. 23, p. 643-660, 2001.

[133] K.-c. Lee, J. Ho et D. J. Kriegman, « Acquiring linear subspaces for face recognition under variable lighting », *IEEE Transactions on Pattern Analysis and Machine Intelligence*, t. 27, p. 684-698, 2005.

[134] A. K. Jain, A. A. Ross et K. Nandakumar, *Introduction to biometrics*. Springer, 2011, p. 118-119.

[135] P. Belhumeur, J. Hespanka et D. Kriegman, « Eigenfaces vs Fisherfaces : recognition using classs specific linear projection », *IEEE Transactions on Pattern Analysis and Machine Intelligence*, t. 19, p. 711-720, 1997.

[136] M.-H. Yang, N. Ahuja et D. J. Kriegman, « Face recognition using kernel eigenfaces », in *International Conference on Image Processing*, 2000, p. 37-40.

[137] S. Brunton et J. Kutz, *Data-Driven Science and Engineering : Machine Learning, Dynamical Systems, and Control*. Cambridge University Press, 2019, p. 3-46.

[138] P. J. Brockwell et R. A. Davis, *Introduction to time series and forecasting*. Springer, 2002, p. 36.

[139] P. Nayman, *Bases et techniques avancées en traitement du signal*. Ellipses, 2017, p. 651-653.

[140] D. Declerq et A. Quinquis, *Détection et estimation des signaux*. Hermes, 1996.

[141] G. Fleury, *Analyse spectrale*. ellipses, 2001.

[142] N. Goodman, « Statistical analysis based on a certain multivariate complex gaussian distribution », *The annals of Mathematical Statistics*, t. 34, p. 153-177, 1963.

[143] R. Hankin, « The complex multivariate gaussian distribution », *The R journal*, t. 7, p. 73, 2015.

[144] A. Geron, *Machine Learning avec Scikit-Learn*. Dunod, 2017, p. 39, 196-197, 205.

[145] L. Morge-Rollet, F. Le Roy, D. Le Jeune, C. Canaff et R. Gautier, « RF eigenfingerprints, an efficient method of RF Fingerprinting in IoT context », *MDPI Sensors*, t. 22, 2022.

[146] S. Kuzdeba, J. M. Carmack et J. Robinson, « RF Fingerprinting with Dilated Causal Convolutions–An Inherently Explainable Architecture », in *Asilomar Conference on Signals, Systems, and Computers*, 2021, p. 292-299.

[147] Matlab, *Design a Deep Neural Network with Simulated Data to Detect WLAN Router Impersonation (dernière consultation 25/06/2023)*, 2020. adresse : `https://fr.mathworks.com/help/comm/ug/design-a-deep-neural-network-with-simulated-data-to-detect-wlan-router-impersonation.html`.

[148] H. Borras, G. D. Guglielmo, J. M. Duarte et al., « Open-source FPGA-ML codesign for the MLPerf Tiny Benchmark », in *arXiv*, 2022.

[149] A. Devices, *AD9361 - Data Sheet (dernière consultation le 25/06/2023)*, 2016. adresse : `https://www.analog.com/media/en/technical-documentation/data-sheets/ad9361.pdf`.

[150] A. Duque, « Machine Learning sur des objets connectés avec Tensorflow Lite pour l'agriculture verticale », *Linux Magazine*, t. 239, p. 6-14, 2020.

[151] P. Warden et D. Situyake, *TinyML : Machine Learning With Tensorflow Lite on Arduino and Ultra-Low-Power Microcontrollers*. O'Reilly, 2020, p. 1-4, 404-406.

[152] F. Fahim, B. Hawks, C. Herwig et al., « hls4ml : An Open-Source Codesign Workflow to Empower Scientific Low-Power Machine Learning Devices », in *arXiv*, 2021.

[153] Xilinx/AMD, *Vivado Design Suite User Guide - High Level Synthesis (dernière consultation le 25/06/2023)*, 2020. adresse : `https://docs.xilinx.com/v/u/2019.2-English/ug902-vivado-high-level-synthesis`.

[154] S. A. R. Shah et B. Issac, « Performance Comparison of Intrusion Detection Systems and Application of Machine Learning to Snort System », *Future Generation Computer Systems*, t. 80, p. 157-170, 2017.

[155] J. Sérot, *Les circuits FPGA et le langage VHDL : une introduction pour les programmeurs et par l'exemple*. ellipses, 2019, p. 63-76.

[156] F. Hamanaka, T. Odan, K. Kise et T. V. Chu, « An Exploration of State-of-the-Art Automation Frameworks for FPGA-Based DNN Acceleration », *IEEE Access*, 2023.

[157] L. R. Rabiner et B. Gold, *Theory and application of digital signal processing*. Prentice-Hall, 1975, p. 302-303.

[158] M. Blott, T. B. Preusser, N. J. Fraser, G. Gambardella, K. O'Brien et Y. Umuroglu, « FINN-R : An End-to-End Deep-Learning Framework for Fast Exploration of Quantized Neural Networks », in *arXiv*, 2018.

[159] H. Zhao, P. C. Yuen, J. T.-Y. Kwok et J.-Y. Yang, « Incremental PCA based face recognition », in *ICARCV 2004 8th Control, Automation, Robotics and Vision Conference*, 2004, p. 687-691.

[160] H. Zhao, P. C. Yuen et J. T.-Y. Kwok, « A novel incremental principal component analysis and its application for face recognition », *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, t. 36, p. 873-886, 2006.

[161] E. Baccelli, N. Anciaux, K. Bhargavan et al., *Internet des Objets - Défis sociétaux et domaines de recherche scientifique pour l'Internet des Objets (IoT)*. Inria, 2022, p. 81.

[162] V. Matyás et Z. Ríha, « Security of biometric authentication systems », 2010, p. 19-28.

[163] M. Edman et B. Yener, *Active Attacks Against Modulation-based Radiometric Identification (dernière consultation le 25/06/2023)*, 2009. adresse : `http://www.cs.rpi.edu/research/pdf/09-02`.

[164] Y. Shi, K. Davaslioglu et Y. E. Sagduyu, « Generative Adversarial Network for Wireless Signal Spoofing », in *Proceedings of the ACM Workshop on Wireless Security and Machine Learning*, 2019, p. 55-60.

[165] F. Restuccia, S. D'oro, A. Al-Shawabka et al., « Hacking the Waveform : Generalized Wireless Adversarial Deep Learning », in *arXiv*, 2020.

[166] G. De Broglie, L. Morge-Rollet, D. Le Jeune et al., « New Methods for Fast Detection for Embedded Cognitive Radio », in *International Conference on Image Processing*, 2022, p. 2007-2014.

[167] C. Slimani, L. Morge-Rollet, L. Lemarchand, D. Espes, F. L. Roy et J. Boukhobza, « Characterizing Intrusion Detection Systems On Heterogeneous Embedded Platforms », in *26th Euromicro Conference on Digital System Design*, 2023.

[168] L. Morge-Rollet, D. L. Jeune, F. L. Roy, C. Canaff et R. Gautier, « Drone Detection and Classification Using Physical-Layer Protocol Statistical Fingerprint », *MDPI Sensors*, t. 22, 2022.

[169] N. Y. Soltani, K. Sankhe, J. G. Dy, S. Ioannidis et K. R. Chowdhury, « More is Better : Data Augmentation for Channel-Resilient RF Fingerprinting », *IEEE Communications Magazine*, t. 58, p. 66-72, 2020.

COLLEGES SCIENCES
BRETAGNE POUR L'INGENIEUR
LOIRE ET LE NUMERIQUE

ENSTA BRETAGNE

Titre : Authentification par empreinte radio pour l'IoT

Mot clés : RF Fingerprinting, *Internet of Things*, Feature learning, Deep learning

Résumé :

L'internet des objets ou *Internet of Things* (IoT) désigne les objets reliés à internet, qui intègrent des capteurs et/ou des actionneurs. Leur prolifération augmente considérablement le risque de cybermenaces, il est donc nécessaire de proposer des contre-mesures afin de sécuriser leurs communications, notamment au niveau de la couche physique. Une des possibilités consiste à s'assurer de l'authenticité des messages transmis en utilisant de l'authentification par empreinte radio ou *Radio Frequency Fingerprinting* (RFF). Plus particulièrement, ce type de méthodes exploitent les imperfections des composants d'un émetteur radio, voire celles du canal de propagation, qui sont considérées comme uniques.

L'approche défendue dans ce manuscrit consiste à étudier les propriétés nécessaires à l'utilisation du RFF dans un contexte IoT, mais aussi les méthodes qui en découlent, ainsi que leurs implémentations. Tout d'abord, trois propriétés ont été introduites : l'adaptabilité, la scalabilité et la complexité. Ensuite, deux méthodes de RFF prenant en compte ces propriétés ont été proposées : les réseaux siamois pour le RFF et les *RF eigenfingerprints*. Celles-ci présentent notamment un compromis entre scalabilité et complexité, permettant d'adresser différents cas applicatifs. Pour finir, l'implémentation de ces méthodes a été considérée, que ce soit au sein d'un réseau ou au niveau d'un appareil IoT.

Title: Radio Frequency Fingerprinting in IoT context

Keywords: RF Fingerprinting, *Internet of Things*, Feature learning, Deep Learning

Abstract: The Internet of Things (IoT) defines the objects connected to the Internet, which integrate sensors and/or actuators. Their proliferation considerably increases the risk of cyber threats. Therefore, it is necessary to propose countermeasures in order to secure their communications, especially at the physical layer. One of the possibilities consists in ensuring the authenticity of the messages transmitted by using Radio Frequency Fingerprinting (RFF). Particularly, this type of method exploits the imperfections of the components of a radio transmitter, or even those of the propagation channel, which are considered as unique.

The approach defended in this manuscript consists in studying the properties necessary for the use of the RFF in an IoT context, but also the methods which result from it, as well as their implementations. First, three properties were introduced: adaptability, scalability and complexity. Then, two RFF methods taking into account these properties have been proposed: the siamese networks for RFF and the *RF eigenfingerprints* method. These present a compromise between scalability and complexity, allowing possibility to address different use cases. Finally, the implementation of these methods was considered, either within a network or on IoT device.