Approches d'apprentissage profond pour la détection en couche physique de télécommunications multi-accès

Cyrille Morin

N° d'ordre NNT : 2021LYSEI049

THÈSE de DOCTORAT DE L'UNIVERSITÉ DE LYON
opérée au sein de
INRIA – Équipe MARACAS

École Doctorale N° 160
Électronique, Électrotechnique et Automatique
(ED EEA)

Spécialité / discipline de doctorat :
Traitement du signal et des images

Soutenue publiquement le 22/07/2020, par :
Cyrille Morin

Approches d'apprentissage profond pour la détection en couche physique de télécommunications multi-accès

---

Devant le jury composé de :

Rapporteurs :		
Marco Di Renzo	Directeur de Recherche	CentraleSupélec
Symeon Chatzinotas	Professeur des Universités	University of Luxembourg
Examinateurs :		
Marwa Chafii	Maître de Conférence	ENSEA
Christophe Moy	Professeur des Universités	Université de Rennes 1
Catherine Douillard	Professeur des Universités	IMT Atlantique
Directeurs de Thèse :		
Jean-Marie Gorce	Professeur des Universités	Université de Lyon
Leonardo Sampaio Cardoso	Maître de Conférence	Université de Lyon
Jakob Hoydis	Chercheur Principal	Nvidia

UNIVERSITY OF LYON
Doctoral School of Electronics, Electrotechnics and Automation
(ED EEA)

THESIS

presented in partial fulfilment of the requirements
for the degree of Doctor of Philosophy from the University of Lyon,
the 22/07/2020.

Specialization: Electrical Engineering

Cyrille Morin

Approches d'apprentissage profond pour la détection en couche physique de télécommunications multi-accès

Members of the Jury:

Reviewers:		
Marco Di Renzo	Research Director	CentraleSupélec
Symeon Chatzinotas	Professor	University of Luxembourg
Examiners:		
Marwa Chafii	Associate Professor	ENSEA
Christophe Moy	Professor	Université de Rennes 1
Catherine Douillard	Professor	IMT Atlantique
Thesis supervisors:		
Jean-Marie Gorce	Professor	Université de Lyon
Leonardo Sampaio Cardoso	Associate Professor	Université de Lyon
Jakob Hoydis	Principal Research Scientist	Nvidia

UNIVERSITÉ DE LYON
Électronique, Électrotechnique et Automatique
(ED EEA)

THÈSE

présentée publiquement pour l'obtention
du diplôme de Docteur de l'Université de Lyon,
le 22/07/2020.

Spécialité : Traitement du signal et des images

Cyrille Morin

Approches d'apprentissage profond pour la détection en couche physique de télécommunications multi-accès

Devant le jury composé de :

Rapporteurs :		
Marco Di Renzo	Directeur de Recherche	CentraleSupélec
Symeon Chatzinotas	Professeur des Universités	University of Luxembourg
Examinateurs :		
Marwa Chafii	Maître de Conférence	ENSEA
Christophe Moy	Professeur des Universités	Université de Rennes 1
Catherine Douillard	Professeur des Universités	IMT Atlantique
Directeurs de Thèse :		
Jean-Marie Gorce	Professeur des Universités	Université de Lyon
Leonardo Sampaio Cardoso	Maître de Conférence	Université de Lyon
Jakob Hoydis	Chercheur Principal	Nvidia

Département FEDORA – INSA Lyon - Ecoles Doctorales

SIGLE	ECOLE DOCTORALE	NOM ET COORDONNEES DU RESPONSABLE
CHIMIE	**CHIMIE DE LYON** **https://www.edchimie-lyon.fr** Sec. : Renée EL MELHEM Bât. Blaise PASCAL, 3e étage secretariat@edchimie-lyon.fr	**M. Stéphane DANIELE** C2P2-CPE LYON-UMR 5265 Bâtiment F308, BP 2077 43 Boulevard du 11 novembre 1918 69616 Villeurbanne **directeur@edchimie-lyon.fr**
E.E.A.	**ÉLECTRONIQUE, ÉLECTROTECHNIQUE, AUTOMATIQUE** **https://edeea.universite-lyon.fr** Sec. : Stéphanie CAUVIN Bâtiment Direction INSA Lyon Tél : 04.72.43.71.70 secretariat.edeea@insa-lyon.fr	**M. Philippe DELACHARTRE** INSA LYON Laboratoire CREATIS Bâtiment Blaise Pascal, 7 avenue Jean Capelle 69621 Villeurbanne CEDEX Tél : 04.72.43.88.63 **philippe.delachartre@insa-lyon.fr**
E2M2	**ÉVOLUTION, ÉCOSYSTÈME, MICROBIOLOGIE, MODÉLISATION** **http://e2m2.universite-lyon.fr** Sec. : Sylvie ROBERJOT Bât. Atrium, UCB Lyon 1 Tél : 04.72.44.83.62 secretariat.e2m2@univ-lyon1.fr	**M. Philippe NORMAND** Université Claude Bernard Lyon 1 UMR 5557 Lab. d'Ecologie Microbienne Bâtiment Mendel 43, boulevard du 11 Novembre 1918 69 622 Villeurbanne CEDEX **philippe.normand@univ-lyon1.fr**
EDISS	**INTERDISCIPLINAIRE SCIENCES-SANTÉ** **http://ediss.universite-lyon.fr** Sec. : Sylvie ROBERJOT Bât. Atrium, UCB Lyon 1 Tél : 04.72.44.83.62 secretariat.ediss@univ-lyon1.fr	**Mme Sylvie RICARD-BLUM** Institut de Chimie et Biochimie Moléculaires et Supramoléculaires (ICBMS) - UMR 5246 CNRS - Université Lyon 1 Bâtiment Raulin - 2ème étage Nord 43 Boulevard du 11 novembre 1918 69622 Villeurbanne Cedex Tél : +33(0)4 72 44 82 32 **sylvie.ricard-blum@univ-lyon1.fr**
INFOMATHS	**INFORMATIQUE ET MATHÉMATIQUES** **http://edinfomaths.universite-lyon.fr** Sec. : Renée EL MELHEM Bât. Blaise PASCAL, 3e étage Tél : 04.72.43.80.46 infomaths@univ-lyon1.fr	**M. Hamamache KHEDDOUCI** Université Claude Bernard Lyon 1 Bât. Nautibus 43, Boulevard du 11 novembre 1918 69 622 Villeurbanne Cedex France Tél : 04.72.44.83.69 **hamamache.kheddouci@univ-lyon1.fr**
Matériaux	**MATÉRIAUX DE LYON** **http://ed34.universite-lyon.fr** Sec. : Yann DE ORDENANA Tél : 04.72.18.62.44 yann.de-ordenana@ec-lyon.fr	**M. Stéphane BENAYOUN** Ecole Centrale de Lyon Laboratoire LTDS 36 avenue Guy de Collongue 69134 Ecully CEDEX Tél : 04.72.18.64.37 **stephane.benayoun@ec-lyon.fr**
MEGA	**MÉCANIQUE, ÉNERGÉTIQUE, GÉNIE CIVIL, ACOUSTIQUE** **http://edmega.universite-lyon.fr** Sec. : Stéphanie CAUVIN Tél : 04.72.43.71.70 Bâtiment Direction INSA Lyon mega@insa-lyon.fr	**M. Jocelyn BONJOUR** INSA Lyon Laboratoire CETHIL Bâtiment Sadi-Carnot 9, rue de la Physique 69621 Villeurbanne CEDEX **jocelyn.bonjour@insa-lyon.fr**
ScSo	**ScSo*** **https://edsciencessociales.universite-lyon.fr** Sec. : Mélina FAVETON INSA : J.Y. TOUSSAINT Tél : 04.78.69.77.79 melina.faveton@univ-lyon2.fr	**M. Christian MONTES** Université Lumière Lyon 2 86 Rue Pasteur 69365 Lyon CEDEX 07 **christian.montes@univ-lyon2.fr**

*ScSo : Histoire, Géographie, Aménagement, Urbanisme, Archéologie, Science politique, Sociologie, Anthropologie

Acknowledgements

This work came under three different influences, deep learning, software defined radio, and physical layer modelling. Each bringing different considerations to the table. I wish to thank my three advisors that helped me work at this convergence: Jean-Marie Gorce, Leonardo S. Cardoso and Jakob Hoydis.

- Leonardo for his technical support, mostly on the software defined radio implementations, and his presence in the face of stubborn and mysterious experimental difficulties.
- Jean-Marie for his moral and methodological support, the lengthy discussions, the flow of research ideas.
- Jakob for his technical support, even from afar, mostly on deep learning and modelling, for his careful and constructive reviews of my writings, for his patience when progress seemed to stall.

The work on a PhD does not limit itself to interactions with advisors so I also wish to thank several colleagues and fellow PhD students. Lélio Chetot for all the desk-side discussions about our mutual research issues, shared expertise and general rambling listening. Diane Duchemin for all of the above, but also for the collaboration that lead to a chapter of this thesis. Malcolm Egan for being there at the beginning of the PhD, even though our work quickly diverged. Matthieu Imbert and the rest of the CorteXlab Team for the discussions and the technical support for the experiments. Claire Goursaud, Chantal Mueller, Sébastien Peychet, for the time spent in class. Fayçal Ait Aoudia, Mathieu Goutay, and the rest of the Nokia team for their support on deep learning implementations and their welcome during my visits. And all the others, Dadja, Tarik, Benoit, Régis, Yuki, Yanni, Tanguy, Guillaume, Gautier, ...

I finally wish to thank all those who balanced the stress of research work. Of course my family, for being there in this last straight line of this cone of my life. The theatre troops for all those moments, their friendship, and the grown up it helped me be. The What'Ups for all the time spent together in the last 9 years, and all the trips and paths taken, sometimes foolishly. My flatmates Tanguy and Thomas that made living together enjoyable, even confined inside in these interesting times.

Cyrille Morin, June 2021

Abstract

CURRENT trends point towards an accelerated augmentation of devices with a desire to access the shared radio spectrum, both due to the continued democratisation and capability augmentation of user facing radio devices, such as cellphones, computers, and especially wearables, but also to the deployment of connected objects and sensors. Technology, protocols, and legislation improvements increase the available frequency bands by opening new channels in the GHz range, but the density of devices is nevertheless expected to increase. Multiple access to a shared radio frequency resource leads to situations that are both complex to model, and to tackle with known algorithms, and it is true of detection tasks that arise in the physical layer of a wireless transmission. The class of deep learning algorithms is especially useful in this sort of situation without model, or with non tractable algorithms, as long as a large amount of labelled data is available to train the related neural networks. This thesis aims at adapting the deep learning tool to physical layer detection problems, in successive steps of a decoding chain. First with the problem of detecting the origin of a received packet, starting with hardware fingerprinting of a transmitting device, and extending it to a scenario with multiple active devices at the same time, detecting the set of active devices transmitting an explicit codeword. The next step after origin detection is bit detection, to decode transmitted messages. For that, deep learning is used to learn constellations allowing for an efficient bit detection in a multiple-access scenario, namely the two-user uplink NOMA. Data used to train the networks involved in this thesis are gathered both from simulated models, and from experimental implementations in the FIT/CorteXlab software defined radio test-bed.

Résumé

Les tendances actuelles pointent vers une accélération de l'augmentation du nombre d'objets cherchant à accéder au spectre radio, à la fois par la démocratisation des objets grand public, smartphones, ordinateurs, montres connectées,... et par le déploiement d'objets et capteurs connectés. Des avancées technologiques, protocolaires et législatives augmentent les bandes de fréquence disponibles en ouvrant l'accès à la zone des GHz, mais la densité des objets communicant sur le spectre tend quand même à augmenter. L'accès multiple à une ressource radio partagée mène à des situations qui sont à la fois complexes à modéliser et à aborder avec les algorithmes actuels, et c'est particulièrement vrai pour les taches de type détection présentes au niveau des couches physiques des communications sans fil. Les algorithmes d'apprentissage profond sont particulièrement utiles dans ce type de situation, sans modèle ou avec des algorithmes existant peu pratiques, pour peu qu'une grande quantité de données soit disponible pour entrainer les réseaux de neurones. Cette thèse vise à adapter l'outil de l'apprentissage profond aux problèmes de détection de la couche physique, à différentes étapes de la chaine de décodage. D'abord par le problème de la détection d'origine d'un paquet reçu, commençant par l'identification de caractéristiques matérielles d'un l'objet émetteur, puis étendant ce scénario à un ensemble d'objets actifs simultanément. L'étape suivant la détection de l'origine d'un paquet est la détection des bits, pour décoder les messages transmis. Dans ce cadre, l'apprentissage profond est employé pour apprendre des constellations permettant une détection efficace des bits dans un scénario multi-accès non orthogonal à deux utilisateurs. Les données servant à l'apprentissage des réseaux de neurones impliqués dans cette thèse sont récoltées soit dans des modèles simulés, soit par des expériences implémentées dans l'équipement de radio logicielle FIT/CorteXlab.

Contents

List of Figures

Acronyms

ADSB automatic dependent surveillance-broadcast. 34

AI artificial intelligence. 8

ASK amplitude shift keying. 70

AUD active user detection. 56–58, 60, 61, 65

AWGN additive white Gaussian noise. 58, 71, 73, 78

BER bit error rate. 72, 81, 82

BMD bit-metric decoding. 81

BMI bitwise mutual information. 81, 84

BPSK binary phase-shift keying. 70

BS base station. 56–58

CE cross-entropy. 24, 81, 84

CER codeset error rate. 58, 59, 64

CNN convolutional neural network. 13, 34, 36, 40, 42

CSI channel state information. 77, 80, 83

CSIR channel state information at the receiver. 56–58

DL deep learning. 8, 11, 56, 57, 60, 65, 67

ELU exponential linear unit. 13

EM electromagnetic. 38

FAR false alarm rate. 58, 64

FCS full constellation system. 87–89

FDMA frequency-division multiple access. 74

FIT/CorteXlab Future Internet of Things / Cognitive Radio Testbed [1]. xii, 7, 27, 28, 36–38, 41, 42, 47, 48, 53, 73, 115

GAN generative adversarial network. 9, 19, 73

GPU graphics processing unit. 4, 15, 17

GRU gated recurrent unit. 15

GS geometric shaping. 69, 70

GUI graphical user interface. 114

i.i.d. independent and identically distributed. 58

IoT internet of things. xviii, 2, 32, 54, 56, 74, 107

ISS independant symbols system. 87, 88

KL Kullback-Leibler. 24

LLR log-likelihood ratio. 89, 90

LSTM long short-term memory. 15

LTE long term evolution. 57, 71

MAC multiple-access channel. 70

MAP maximum a posteriori. 59–61, 64

MD minimum distance. 71

MDR misdetection rate. 58, 64

MI mutual information. 72

MIMO multiple-input multiple-output. 1, 28

ML machine learning. 8, 11, 17, 18

MLE maximum likelihood estimate. 59

MSE mean squared error. 23

MTC machine type communication. 56–58

NB-IoT narrowband internet of things (IoT). 56

NN neural network. 21, 23, 34, 36, 37, 48, 57, 60–65, 86

NOMA non orthogonal multiple access. 2, 3, 56, 57, 70, 74, 80, 104

OFDM orthogonal frequency-division multiplexing. 39, 54, 57, 58, 111

OMA orthogonal multiple access. 74, 80

OOK on-off keying. 70

P2P point-to-point. 71, 74, 80, 90

PAM pulse amplitude modulation. 70

PAPR peak-to-average power ratio. 54

PCA principal components analysis. 10

PSK phase shift keying. 70, 87

QAM quadrature amplitude modulation. 70

QoS quality of service. 21, 82

QPSK quadrature phase-shift keying. 39–42, 54, 70

RE ressource element. 79, 80

ReLU rectified linear unit. 12, 15

RF radio frequency. 38

RL reinforcement learning. 73

RNN recurrent neural network. 15

RTN radio transformer network. 22

SCMA sparse code multiple access. 74

SDR software defined radio. 7, 27, 28, 37, 38

SER symbol error rate. 57, 72

SGD stochastic gradient descent. 17

SIC successive interference cancellation. 57, 74

SISO single-input single-output. 75

SNR signal-to-noise ratio. xiii, 10, 58, 62–64, 74

SVD singular value decomposition. 58

SVM support vector machine. 10

TDMA time-division multiple access. 74

UER user error rate. 58, 64, 67

URLLC ultra reliable low latency communications. 2

USRP universal software radio peripheral. 28, 34

w.r.t. with respect to. 58, 59, 61, 64, 65

— 0 —

Synthèse des Contributions

1. Introduction

1.1. Défis des communications actuelles et futures

Les défis que rencontreront les futurs protocoles de télécommunication tels que la 6G ressemblent fortement à ceux qui ont mené aux standards actuels de communication conçus il y a une décennie comme les réseaux cellulaires 4G et 5G, ou les réseaux basse puissance Sigfox ou LoRaWAN. Ils présentent tous des questionnements relatifs à l'augmentation de la vitesse de transmission, la gestion de toujours plus d'objets connectés, la réduction de la latence, l'augmentation de la fiabilité et la gestion de l'énergie. Mais, même si aucun de ces points n'a pu être considéré comme "résolu", et que tous ont été sujet à une augmentation des attentes qui leur sont portés, ils ont tous fortement évolués.

Vitesse de transmission

L'intervalle entre la 3G et la 4G a vu une multiplication par plus de cent de la vitesse de téléchargement des utilisateurs mobiles, de moins de 400 kbit/s à plus 100 Mbit/s, et la 5G promet un décuplement supplémentaire. Mais les dernières avancées ne viennent pas d'une augmentation de l'efficacité de transmission. La 4G est déjà très proche de la limite de Shannon pour les communications point à point donc cet aspect ne permet pas d'envisager de gains futurs. Les derniers développements de celle-ci proviennent de l'introduction des transmission multi-antennes pour une amélioration de la puissance du signal reçu. Et la 5G apporte une augmentation de la largeur des bandes de fréquences et l'ouverture de canaux au delà de 6 GHz. Il est attendu que les futurs standards continuent dans cette voie, augmentant le nombre d'antennes, avec du beamforming plus précis, une meilleure localisation des utilisateurs,... et en passant à de plus hautes bandes de fréquences, allant vers les terahertz.

Objets connectés

Il y a une dizaine d'années, le déploiement de l'internet des objets (IoT) prévoyait d'exploser avec des entreprises comme Cisco prédisant 50 milliards d'objets connectés d'ici 2020 [2], et même une prédiction de mille milliards d'objets en 2015 par IBM [3]. Pourtant, en 2020 moins de 22 milliards d'objets ont été connectés. Même si les attentes étaient largement surestimées, la quantité d'objets à connecter sans fil augmente rapidement, et ces objets ont besoin de bande passante pour communiquer. Certaines solutions à ce problème peuvent ressembler à celles pour augmenter la vitesse de transmission, avec du multiplexage spatial, ou l'exploitation des larges plages de fréquences disponibles dans les GHz et les THz. Mais ces solutions sont très couteuses en terme de puissance de calcul pour les systèmes multi-antennes, et les distances de transmissions diminuent avec l'augmentation de fréquence, et donc elles ne sont pas très compatibles avec les demandes de l'IoT de bas prix et faible demande de maintenance. D'autres approches sont nécessaires pour permettre une grande quantité de transmission dans la zone de spectre actuelle, déjà encombrée. Le challenge du futur est d'introduire des techniques d'accès multiple non orthogonales (NOMA) permettant à plusieurs objets de communiquer simultanément, sur les même ressources.

Qualité de service

Le développement de machines connectées, d'abord à l'intérieur d'usines puis, et de plus en plus, à l'extérieur avec des véhicules connectés, autonomes ou contrôlés à distance, présente d'importants risques pour le grand public. Cette catégorie demande un très haut niveau de service puisque le manque fiabilité peut être au prix de vies humaines. Les communications de véhicules par nature fortement mobiles et dans un environnement urbain très dynamique demande une très faible latence pour que seules des informations à jour soient reçues. Ces deux demandes se combinent dans la classe des communications ultra fiables et faible latence (URLLC). Pour réduire la latence, les flux de gestion des communications jouent un grand rôle puisque la technique courante nécessite au moins trois transmissions successives avant la communication utile. Donc des protocoles spécifiques doivent être développés pour un accès direct, sans autorisation préalable. Il faut aussi utiliser des paquets de petite taille : une latence de 1 ms n'est pas envisageable si le paquet lui-même est plus long que ça. Mais cela rentre en conflit avec les techniques de codages classiques qui se basent sur de long blocs de données pour être fiables. Donc des codes spécifiques doivent être conçus.

Gestion énergétique

Une gestion efficace de l'énergie est de plus en plus importante pour deux types d'acteurs de la chaine de transmission radio. Les grands équipements de type station base sont alimentés directement et donc peuvent se permettre de consommer beaucoup. Mais ils demandent beaucoup d'énergie et cela se traduit dans la plupart des pays par l'utilisation de centrales électriques à carburant fossiles. Donc toute amélioration du rendement énergétique peut avoir un impact important sur l'empreinte carbone de l'industrie des communications. De l'autre coté, beaucoup d'objets ne sont pas branchés, et tirent leur électricité de batteries. Pour des objets grand public comme des téléphones, une augmentation de 10 % d'une batterie durant 24 heures ne mène pas à de grands changements pour l'utilisateur. Mais pour des capteurs IoT déployés dans des zones d'accès difficiles et jusqu'à une dizaine d'années, la

même amélioration entraine d'importantes économies. Pour les deux catégories, des solutions peuvent être trouvées à travers d'algorithmes plus efficients, et une meilleure gestion de la puissance de transmission. Pour augmenter la durée de vie d'objets basse puissance, des systèmes de récolte d'énergie sont en développement, tout comme des protocoles de transmission simultanée d'énergie et d'information.

1.2. Apprentissage machine pour les communications

Dans tous les domaines présentés, avec utilisateurs ou antennes multiples, fréquences élevées,... les concepteurs s'éloignent des modèles établis avec des systèmes de plus en plus complexes et dynamiques. Pour ces cas là, tout comme les cas où la vitesse et le rendement sont important, il faut de nouveaux algorithmes, et c'est pour cette raison que l'apprentissage machine, et surtout l'apprentissage profond, est devenu un champ d'étude important dans les communications. Ces techniques ont été proposées il y a de nombreuses années, avec une enquête datant de 2000 présentant de nombreuses utilisation des réseaux de neurones pour la modélisation de canal, l'égalisation, le décodage, le filtrage,... et prédisant "Les réseaux de neurones seront certainement une des technologies clé pour le domaine des communications au 21ème siècle" [4]. Néanmoins, et comme dans d'autres domaines, l'utilisation de ces techniques est restée à la marge pendant plus d'une décennie. Et il a fallu attendre des amélioration notables dans les vitesses de calcul, les technologies de cartes graphiques, et de remarquables démonstrations dans le domaine du traitement d'images pour que l'utilisation de l'apprentissage profond intéresse la communauté, en commençant vers 2016 avec des travaux influents sur la classification de modulation [5], le décodage canal [6], ou la représentation d'une chaine de transmission comme un autoencodeur [7]. Ce moment correspond aussi à un point où la recherche avait commencée à s'éloigner des modèles bien étudiés pour s'attaquer aux défis du moment, et la communauté avait besoin de nouveaux outils pour gérer ces nouvelles conditions. Depuis, les techniques d'apprentissage ont imprégné la plupart des taches de traitement du signal dans la couche physique des communications, avec le design de constellations, l'allocation de ressources, la gestion des interférences,...[8], [9].

1.3. Plan des contributions

Le travail présenté dans cette thèse vise à améliorer la performance d'algorithmes de détection présents dans la chaine de traitement du signal d'un récepteur dans des conditions où des transmissions de multiples objets sont attendus et ce, en utilisant des techniques d'apprentissage profond. Pour cela, le chapitre 2 (non reproduit dans la version en français) présente des bases en apprentissage profond avec sa place dans l'écosystème de l'apprentissage machine, suivi de la construction d'un réseau de neurones et son entrainement tel qu'utilisé dans cette thèse. Il apporte également des éléments concernant l'obtention de sets de données pour entrainer et évaluer tels réseaux, avec la description de potentiels problèmes qui peuvent survenir dans le processus d'obtention. Enfin il présente aussi les outils utilisés pour les expériences d'obtention de données avec radio logicielle.

La figure 1 présente les différentes taches abordées dans les chapitres suivants de cette thèse remis dans le contexte d'une chaine émission-réception. Du point de vue du récepteur, le premier traitement est réalisé par le matériel, filtrage, amplification, démodulation en bande de base,... avant d'être converti en échantillons numériques. Cette partie du traitement n'est pas triviale mais puisqu'elle est analogique, elle n'est pas compatible avec les algorithmes

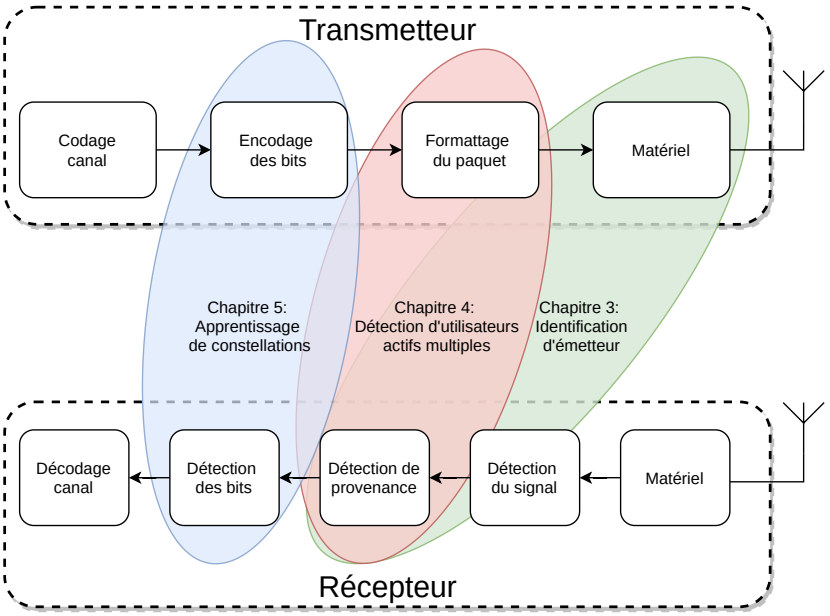

FIGURE 1. : Plan de la thèse

numériques. Après numérisation, la première étape consiste à détecter si un signal est reçu. Tant que le protocole de transmission est connu, cette tâche est relativement simple à implémenter avec des algorithmes classiques donc l'utilisation de l'apprentissage machine n'est pas pertinent. La première tache de détection à aborder est donc la suivante : la détection de la provenance d'un paquet reçu, indispensable pour pouvoir utiliser ses données.

Le chapitre 3 commence à aborder cette tâche dans un scénario où un utilisateur transmet à la fois, parmi un ensemble d'utilisateurs connus. Dans ce chapitre, les imperfections créées par le matériel de l'émetteur sont utilisées pour identifier le transmetteur, à la manière d'une empreinte digitale. Un réseau de neurones de convolution est entrainé sur des données expérimentales provenant d'objets du même modèle, et l'impact du type de données transmises est étudié, tout comme des méthodes de construction d'un set de données pour augmenter la robustesse de la classification envers des modifications des caractéristiques du canal.

Le chapitre 4 étend le scénario du chapitre précédent avec plusieurs utilisateurs actifs en même temps. L'utilisation des imperfections matérielles n'est plus possible à ce moment là, car trop fragile. Chaque émetteur a donc un code spécifique pour l'identification. Ce chapitre cherche à concevoir un décodeur capable de détecter l'ensemble des utilisateurs actifs dans un environnement simulé, et compare sa performance à des algorithmes plus conventionnels.

Après avoir abordé la détection d'origine, le chapitre 5 s'intéresse à l'étape suivante dans la chaine de réception : la détection des bits. Ce chapitre considère un scénario avec deux utilisateurs transmettant simultanément, avec l'objectif de détecter les deux flux de bits au niveau d'un récepteur commun. Ce problème peut être assimilé à celui du chapitre d'avant, où l'activité d'un utilisateur peut être vue comme un bit d'information. Mais ici, l'attention est portée sur la conception d'un set de codes (ou constellation) adapté à ce scénario et permettant une détection précise. Il s'intéresse aussi aux compromis inhérents qui surviennent quand on considère les performances de plus d'un utilisateur à la fois.

Pour finir, le chapitre 6 conclut cette thèse et apporte une discussion sur les considérations futures des problèmes abordés ici.

2. Identification d'émetteur

Comme mentionné dans l'introduction, la première étape dans une chaine de réception qui peut bénéficier d'apprentissage machine est la détection de la provenance d'un signal reçu, pour pouvoir utiliser ses données de manière appropriée. Traditionnellement, l'identification d'un émetteur se base sur une coopération entre transmetteur et récepteur, par le biais d'un code d'identification. Cette méthode peut être problématique puisqu'elle nécessite, d'un coté de transmettre des données non utiles, et de faire confiance à l'émetteur, que celui-ci ne cherche pas à se faire passer pour un autre. Pour des objets connectés, à bas coût et sobres en énergie, la transmission d'un code d'identification consomme des ressources précieuses. Et de manière générale, cette étape du traitement est sujette à des attaques [61].

Pour résoudre ce problème, il faut trouver d'autres moyens d'identifier un transmetteur, idéalement d'une manière qui dépend de caractéristiques physiques de l'émetteur. Dans le monde réel, ces émetteurs sont construits avec des composants qui, même si ils respectent les spécifications, diffèrent légèrement les uns des autres. Cela crée des variations dans le signal radio, faibles mais perceptibles, et ce, même entre objets du même modèle. La combinaison des variations de tous les composants d'un objet peut être vue comme une empreinte radio, qui pourrait être reconnue par le récepteur.

De nombreux travaux visent à utiliser cette empreinte radio pour identifier un émetteur, et ce, en utilisant des simulations d'imperfections matérielles, ou des données recueillies sur du vrai matériel, téléphones, équipements Wifi, objets Zigbee ou de radio logicielle. Une analyse plus précise de l'état de l'art est disponible dans le chapitre 3 et la table 3.1. Il en ressort que la majorité des travaux précédents ne prend pas en compte l'effet du canal sur la précision de l'identification. Certains constatent cet impact, autrement dit que, si rien n'est fait pour y remédier, la précision s'effondre dès d'un émetteur ou un récepteur est déplacé. Enfin, des papiers récents proposent des solutions pour augmenter la fiabilité en ajoutant artificiellement des perturbations au signal émit, augmentant la force de l'empreinte radio. Le travail présenté ici cherche à réduire l'impact négatif du canal, mais sans introduire d'élément artificiel à l'émission, qui pourraient être usurpés par des attaquant, de la même manière que les codes actuels.

2.1. Modélisation

Formulation du problème

Le problème d'identification d'émetteur consiste en un ensemble de N émetteurs (aussi appelés *nœuds*) et de M récepteurs. Le $k$ième nœud envoie la séquence de complexes $x_k(t)$, et le $i$ième récepteur reçoit la séquence complexe $y_i(t)$. Dans un système standard, une partie de $x_k(t)$ est utilisée pour identifier le nœud, en l'occurrence le champ d'identification à l'intérieur de l'en-tête du paquet. Contrairement à ce système, on considère ici que $x_k(t)$ ne contient rien permettant l'identification, x est indépendant de k et on définit $x_k(t) = x(t)\ \forall k$.

La séquence k passe ensuite dans la chaine radio du nœud en question, possédant une fonction de transfert globale g_k inconnue et pouvant être non linéaire. Cette fonction de transfert correspond aux effets combinés de toutes les opérations effectuées au signal, tel que la

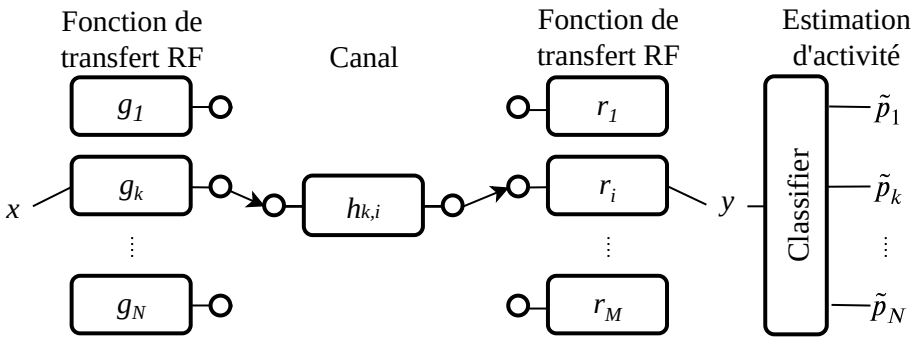

FIGURE 2. : Modèle du système

conversion numérique analogique, le mixage, les filtrages et l'amplification. Le signal résultant est ensuite transmit dans un canal $h_{k,i}$, présent entre le nœud k et le récepteur i, et de réponse inconnue. Le récepteur applique par sa chaine de traitement radio, la fonction de transfert r_i pouvant également être non linéaire. Cela mène à l'observation de la séquence échantillonnée $y(t)$ à partir de laquelle la détection de l'utilisateur actif doit être faite, correspondant à l'équation suivante :

$$y(t) = r_i(h_{k,i} \times g_k(x(t))) \tag{1}$$

Où le nœud actif k est tiré dans $K \sim \mathcal{U}_{\mathbb{N}}(0, N)$ et le récepteur actif est tiré dans $I \sim \mathcal{U}_{\mathbb{N}}(0, M)$, avec $\mathcal{U}_{\mathbb{N}}(a, b)$ la distribution uniforme sur les entiers entre a et b (b exclu). Cette situation ne considère pas de collisions entre paquet émis. Seul un nœud et un récepteur sont actifs à la fois.

La question a traiter est donc la suivante : Y a-t-il suffisamment d'information contenue dans le signal observé $y(t)$ pour trouver k sans utiliser la signature du canal $h_{k,i}$ mais en exploitant uniquement la signature radio de l'émetteur g_k ? L'objectif est donc de concevoir un système capable de calculer les fonctions de classification $\tilde{p}_k = f_k(y(t))$ avec $\tilde{p}_k$ l'estimation de la probabilité d'activation du $k$ième nœud. Pour évaluer la performance d'une telle classification, la fonction de perte utilisée est la cross-entropie catégorique [14] :

$$\mathcal{L} = -\sum_{a=1}^{N} \mathbf{1}_{[a=k]} \cdot \log\left(\tilde{p}_k\right). \tag{2}$$

Classification par apprentissage profond

Un système à même de minimiser 2 permet de répondre à la première moitié de la question posée : Y a-t-il suffisamment d'information dans le signal observé pour trouver l'identité de l'émetteur. La seconde moitié de la réponse doit venir d'un apprentissage sur un ensemble de données bien construit.

Pour cela, l'approche retenue est l'entrainement d'un classificateur par réseau de neurones, fonctionnant sur le signal observé brut y et sortant directement les probabilités estimées $\tilde{p}_k$ pour chacun des nœuds. L'utilisation de l'apprentissage profond se justifie : les caractéristiques utiles de g_k sont inconnues et certainement non linéaire, et il est possible, via une mise en place expérimentale sur la plateforme FIT/CorteXlab, d'obtenir d'une grande quantité de données labellisées pour l'entrainement. Le réseau est entrainé en utilisant la fonction de

perte 2, mais la seule utilisation de celle-ci ne permet pas de discriminer entre les effets de g_k et de $h_{k,i}$ puisqu'ils dépendent tous deux de k. Pour faire cette discrimination, il faut un ensemble de données construit spécifiquement pour réduire l'impact de $h_{k,i}$ à l'apprentissage.

2.2. Implémentation

Réalisation expérimentale

Le travail présenté ici ne se focalise pas sur une source particulière d'imperfection, mais s'intéresse à l'impact du matériel dans son ensemble. Une simulation n'est donc pas envisageable, et les données utilisées doivent provenir d'expériences avec du vrai matériel. Pour cela, la plateforme expérimentale FIT/CorteXlab est utilisée. Elle consiste en une salle isolée du monde extérieur et contenant, entre autre, 22 radio logicielles de type universal software radio peripheral (USRP) du même modèle. Cette salle permet de s'assurer de l'absence d'interférence, et de contrôler l'évolution du canal : tant que rien n'est déplacé, le canal est statique. Un robot de type Turtlebot est disponible dans la salle pour faire ces déplacements de manière automatique.

La programmation des nœuds radio est faite avec l'aide de GNU Radio avec un code pour l'émission, un pour la réception, et un pour organiser les émissions en s'assurant de l'absence de collisions. Chaque paquet émis contient un préambule pour que le récepteur détecte qu'il y a un signal, un en-tête qui contient l'identité de l'émetteur, et une séquence de données. Un intervalle de garde est introduit pour éviter les interférences entre chaque élément . La séquence de données est enregistrée dans le fichier correspondant à son émetteur, en fonction de l'en-tête reçu, de manière à pouvoir fournir un label à l'entrainement. Cet en-tête n'est pas utilisé après cela, et les données enregistrées ne contiennent pas d'informations relatives à l'émetteur.

La séquence de données émise fait 560 échantillons de long, et le récepteur enregistre 600 échantillons de manière à capturer le niveau de bruit et les phases transitoires au début et à la fin des données. Pour un scénario donné, l'ensemble de données (ou dataset) contient 50000 exemples de 600 échantillons pour chaque émetteur.

Paramètres expérimentaux

Chaque expérience pour créer un set de données est définie par un tuple de paramètres créant un scénario : le type de données, le type d'émission, et le nombre de récepteurs.

Données Il y a trois types de données étudiées :

- *Static* : Une séquence fixe, prédéfinie, et commune à tous les nœuds. Cela correspond l'utilisation du préambule d'un paquet pour l'identification
- *Random* : Une séquence aléatoire de bits (changeant à chaque paquet) encodés en QPSK. Cela correspond à utiliser la séquence de données utiles d'un paquet.
- *Noise* : Une séquence aléatoire de bruit uniforme. Cela correspond au pire cas possible de modulation.

TABLE 1. : Caractéristiques du signal

Parameter	**Value**
N	21
M	1 (*MonoRX*), 21 (*MultiRX*)
Bande de fréquence	ISM 433 MHz
Largeur du signal	2.5 MHz
Forme d'ondes	QPSK (filtrage RRC), Uniforme
Échantillons/symbole (QPSK)	2
Échantillons de données émis	560

Type d'émission Comme le canal dans la salle expérimentale est statique, et que les équipements sont fixés, au plafond et séparés les uns des autres, le canal de propagation est assez différent d'un nœud à l'autre. Or l'objectif est d'identifier en se basant sur les caractéristiques du matériel, et pas du canal. Pour remédier à cela, trois types d'émission sont considérés :

- *Plain* : Rien n'est fait pour éviter les biais dus au canal. Ce mode sert de point de comparaison avec les deux autres.
- *Varying* : L'amplitude du signal est variée d'un paquet à l'autre pour émuler une variation de la distance de propagation.
- *Robot* : Un robot est introduit et se déplace aléatoirement dans la salle pour apporter des réflexions dynamiques et changer les caractéristiques multi-chemin du canal. Ce mode inclut aussi les variations d'amplitude du mode précédent.

Nombre de récepteurs Dans le mode de base, appelé *MonoRx* un seul nœud est utilisé en tant que récepteur : $N = 21$ et $M = 1$. Pour augmenter les variations de canal, et réduire la capacité du système à apprendre des propriétés du canal, un deuxième mode, *MultiRx* est introduit. Il combine des observations de plusieurs récepteurs en un set de données. Ce mode est obtenu en collectant un ensemble de 21 expériences en MonoRx avec $N = 20$ et $M = 1$, en changeant le récepteur entre chaque, pour obtenir un dataset combiné avec $N = 21$ et $M = 21$. On peut noter que un paquet émis à un instant donné n'aura qu'un seul récepteur, mais que, à terme, chacun des 21 nœuds aura été écouté par les 20 autres. Le réseau de neurone (présenté après) utilise dans tous les cas les données d'un seul paquet, sans connaitre, ni les paramètres du scénario, ni l'identité du récepteur.

La table 1 résume les caractéristiques de la charge utile utilisée pour l'identification. Et les données collectées sont disponibles en ligne, accompagnées de scripts pour en générer de nouvelles [102].

Système apprenant

Architecture Comme présenté dans la figure 3, le réseau de neurones utilisé est basé sur une architecture de réseau convolutif (CNN) avec cinq couches de convolution 2D avec une couche de max pooling entre chaque, le tout suivi d'une étape d'aplatissement et de six couches denses avant une sortie finale en Softmax. Les échantillons complexes utilisés

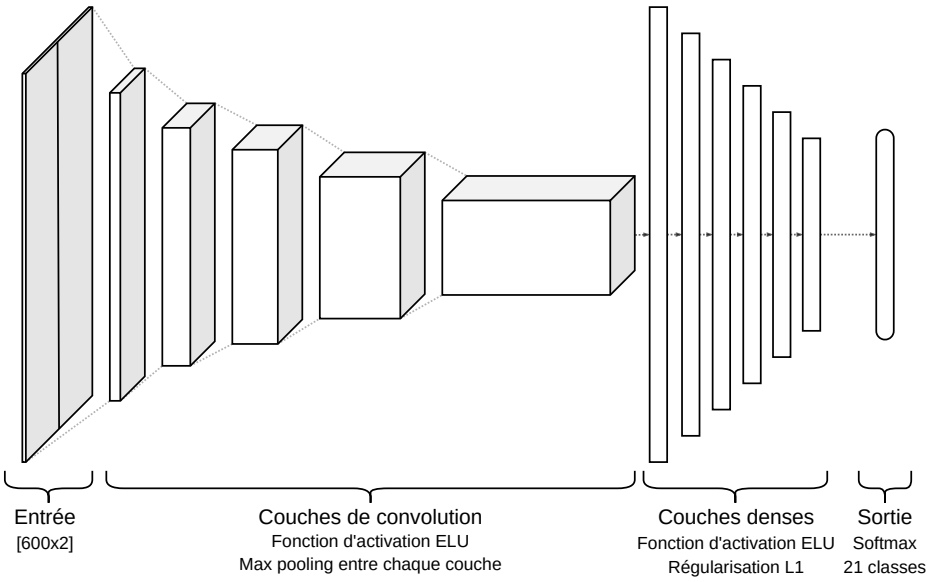

FIGURE 3. : Architecture du réseau de classification

en entrée sont représentés par deux nombres réels indépendants codant leurs coordonnées Cartésiennes. Le vecteur d'entrée de 600 échantillons complexes devient donc une matrice de 600×2 nombres réels utilisés comme images 2D par les couches de convolution. La sortie du réseau consiste en un vecteur de probabilité (ou de certitude) d'activation des 21 transmetteurs possibles dont la somme est égale à 1.

Entrainement Chaque set de données est mélangé aléatoirement avant entrainement, et découpé de manière standard. 70 % des échantillons sont utilisés pour l'entrainement, 10 % pour la validation, et 20 % pour le test. Chaque scénario est utilisé pour entrainer des instances différentes de l'architecture présentée ci-dessus. Les exemples d'entrainement sont présentés par mini-lots de 128 exemples, sur 30 époques dans le cas des scénarios MonoRx, et 100 époques dans le cas du MultiRx. L'optimiseur utilisé est Adam, avec un taux d'apprentissage de 0.001, minimisant la fonction de perte de cross-entropie catégorique, entre les labels et les prédictions. L'affinage des hyper-paramètres du réseau est fait sur un set de données avec un scénario complexe : MultiRx avec variation d'amplitude et séquence aléatoire (*Random*). Des entrainements d'une durée de 10 époques sont fait avec différents paramètres (taille de lot, taux d'entrainement, nombre de couches,...) et les plus performants sont retenus.

2.3. Résultats

La première étape cherchait à savoir si le type de données utilisées impacte la performance de l'identification. Six datasets correspondant aux combinaisons des trois types de données avec les types d'émission *Plain* et *Varying* sont collectés, à la fois pour le mode MonoRx et MultiRx, et les performances obtenues sont comparées. Il en ressort que le mode de données *Static* permet une meilleure précision. Le réseau profite du fait de connaitre à l'avance le signal à recevoir. Ce résultat permet d'envisager d'utiliser le préambule d'un paquet pour faire

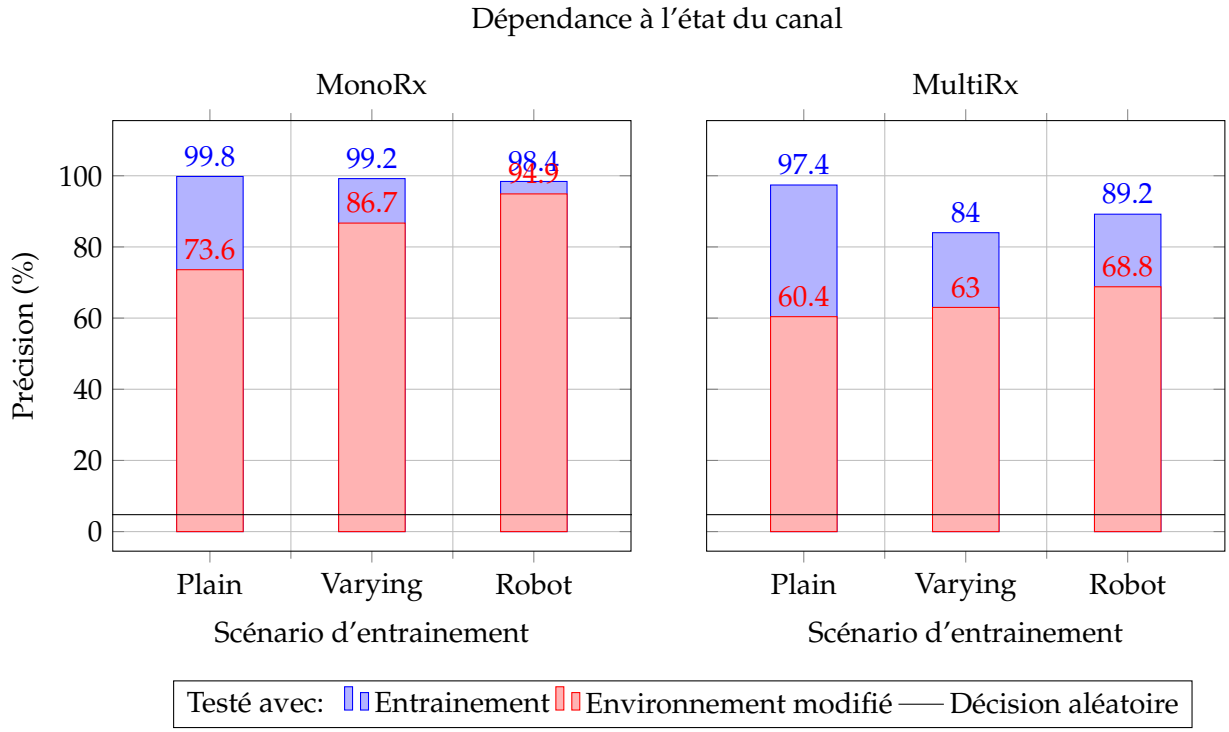

FIGURE 4. : Précision obtenue par des réseaux entrainés sur un scénario avec données *Static*, et testés, d'abord sur les données du dataset d'entrainement, puis sur un autre dataset, du même scénario, mais avec modification de l'environnement, et donc du canal.

l'identification puisque cet élément est émis quoi qu'il arrive, étant nécessaire à la détection du signal, et qu'il est constant.

Pour évaluer la sensibilité des réseaux appris aux modifications du canal, de nouveaux datasets de test sont créés. Dans ceux-ci, l'environnement dans la salle d'expérimentation est modifié en ajoutant un tabouret métallique, de manière à créer de nouvelles réflexions, et donc de nouvelles caractéristiques de propagation. Des réseaux entrainés sur des données avant la modification sont testés sur ces nouvelles conditions. La figure 4 présente les résultats correspondant, la précision sur les données avant modification de l'environnement étant donnée en bleu, et la précision après modification étant donnée en rouge, dans les deux cas de MonoRx et MultiRx. Ces résultats confirment la grande sensibilité des réseaux quand le type d'émission *Plain* est utilisé : une importante réduction de performance est observée. Cela correspond aux observations de la littérature. Ils montrent aussi qu'apprendre sur des données plus complexes (scénario avec robot) permet de réduire la perte de performance de manière significative, notamment dans le cadre du MonoRx. Le MultiRx présentant une difficulté supplémentaire dans le sens où le réseau doit apprendre à gérer, à la fois les variations de canal, mais aussi la diversité des fonctions de transfert RF r_i dans les données à traiter. Ces résultats montrent qu'une construction adaptée de l'ensemble de données d'entrainement permet effectivement de focaliser la classification par un réseau de neurones sur les caractéristiques physiques de l'émetteur, sans s'appuyer sur celles du canal.

2.4. Conclusion

Ce travail a permit de montrer qu'il est possible d'améliorer la robustesse de l'opération d'identification d'émetteur par rapport aux variations de canal, sans avoir besoin de recourir à l'ajout d'imperfections artificielles dans le signal émis. Il en découle les conclusions suivante par rapport au données d'entrainement : les données utilisées pour l'identification devant être déterministes autant que possible (le préambule d'un paquet étant un bon candidat), et le canal doit être rendu aussi variable que possible au moment de l'entrainement de manière à forcer le réseau de neurones à apprendre à gérer ces variations.

Pour pouvoir envisager un déploiement, certains aspects doivent encore être étudiés pour améliorer la fiabilité et la praticité de l'opération. La conception d'une séquence de préambule adaptée à la fois à une détection efficace du signal et à faire ressortir les caractéristiques matérielles de l'émetteur peut être recherchée pour améliorer la précision de l'identification. Une recherche est nécessaire sur la quantité suffisante d'échantillons pour permettre une classification efficace dans un temps raisonnable. Enfin, le système présenté ici se base sur la connaissance préalable du nombre d'utilisateurs, et ne peut pas gérer de nouveaux nœuds. Des architectures capables de détecter des intrus, des erreurs, ou d'ajouter des utilisateurs après déploiement permettraient une implémentation plus versatile et de gérer des situations dynamiques.

3. Détection d'utilisateurs actifs multiples

Cette partie reprend la tache de détection de la provenance d'un signal abordée précédemment et l'étend à une situation où plusieurs utilisateurs peuvent transmettre en même temps. Dans ce cas, l'empreinte radio n'est plus une bonne source d'information, et il faut utiliser des codes dans les paquets transmis pour pouvoir gérer l'interférence. Cette situation est envisagée dans le cadre de réseaux radio dédiés à l'IoT à accès massif. Dans ces conditions, une quantité très importante de capteurs peut s'activer de manière sporadique et est sujette aux contraintes des communications de type machine. Ces réseaux sont amenés à être implémentés principalement avec des objets à bas coût, ayant donc des fonctionnalités radio assez faibles, et peu de ressources en terme de puissance et de capacité de calcul. En plus de cela, ils ont les mêmes demandes en terme de latence et d'efficacité spectrale et énergétique. Les transmissions montantes de ces capteurs simples visent donc à minimiser le nombre d'émissions, ainsi que leur durée, vu la faible quantité de puissance disponible, et de données à envoyer. Pour de tels objets, un surdébit important peut fortement réduire la durée de vie opérationnelle, vu qu'ils passeraient plus de temps, et donc d'énergie, à transmettre des messages protocolaires, plutôt que des données utiles. Contrairement aux procédures d'accès de type 4G actuelles telles que planifiées en NB-IoT, un message pouvant être envoyé sans autorisation préalable et contenant à la fois une demande d'accès, de quoi identifier l'émetteur, et des données, serait idéal.

Pour atteindre cet idéal, le plus grand défi est la détection du sous-ensemble de capteurs actifs par la station base. Pour permettre la transmission des identifiants utilisateurs, même au sein d'un réseau très dense, il faut utiliser une technique dédiée au partage de spectre qui soit compatible avec une charge importante de trafic et un grand nombre d'utilisateurs potentiels. L'accès multiple non-orthogonal (NOMA), et en particulier sa version basée sur des codes, est un bon candidat puisqu'il permet de limiter les collisions de transmissions

simultanées, sans avoir besoin de séquences d'accès extrêmement longues.

Dans ce cas là, toute la complexité de la détection d'utilisateurs actifs (AUD) est déportée sur la station base. Le fait d'éviter la procédure d'établissement de la liaison demande un algorithme efficace de détection pour fonctionner sur un seul message d'accès et avec une information d'état du canal au récepteur (CISR) limitée. Le détecteur optimal est connu, mais est extrêmement complexe et n'est donc pas compatible avec une implémentation temps réel. Une version itérative de ce détecteur optimal a été introduite, amenant une réduction de complexité, et aussi de performance, mais ce n'est toujours pas suffisant. Le défi est donc de proposer un algorithme efficace alliant performance et rapidité d'exécution. Pour cela, l'approche proposée ici est d'utiliser un détecteur basé sur l'apprentissage profond.

3.1. Modèle

Le canal à accès aléatoire est important pour garantir un accès équitable au médium radio dans le cadre des communication massives de type machine. On considère ici que les nœuds capteurs, simplement appelés nœuds par la suite, reçoivent à l'avance un code aléatoire qui est utilisé pour envoyer une requête d'allocation de ressource à la station base qui leur est associée. Dans l'approche standard, si deux nœuds envoient une requête au même moment, cela crée une collision, et au moins l'un des deux messages est perdu (voire les deux). Ici, connaissant les codes distribués aux nœuds, la station base essaie de déterminer l'identité de tous les nœuds impliqués dans une requête. Dans le modèle étudié ici, on considère que la station base transmet un signal permettant aux nœuds de se synchroniser et de contrôler leur puissance d'émission pour que le signal reçu à la station base ait une puissance moyenne stable. L'état du canal instantané n'est pas connu, et aucun pilote n'est utilisé. Le détecteur opère donc en mode de détection non-cohérente.

Dans la suite de cette section, les notations suivantes sont utilisées : $(\mathcal{U}, \Phi)$ est un espace mesurable dans lequel $\mathcal{U}$ dénote l'ensemble des nœuds, avec une cardinalité de $K{=}|\mathcal{U}|$ et $\Phi{=}\mathcal{P}(\mathcal{U})$ est l'ensemble puissance de $\mathcal{U}$. Un sous-ensemble de nœuds est noté $\mathcal{A} \in \Phi$. Pour un créneau d'accès aléatoire donné, l'ensemble des nœuds *actifs* est noté $\underline{\mathcal{A}} \in \Phi$. La fréquence d'activité de ces nœuds est faible (inférieure à 0.5), ce qui implique un ensemble creux. De plus, l'activité des nœuds suit une loi de Poisson de moyenne λ (la probabilité d'activité d'un nœud est donc $\theta = \lambda / K$). Un dictionnaire $\mathcal{C}$ est généré et partagé par tout le monde. Chaque nœud k possède donc son propre code dédié $\mathbf{c}_k$ de longueur M ayant une puissance unitaire.

Comme mentionné précédemment, les messages reçus sont considérés comme synchronisés et avec un contrôle de puissance moyenne permettant que les messages soient reçus avec un rapport signal à bruit (SNR) moyen ρ. La station base comprend N antennes et les nœuds n'en ont qu'une. Les transmissions passent dans un canal à évanouissement plat et par bloc de Rayleigh, modélisé par un vecteur aléatoire $\mathbf{h}_k \sim \mathcal{N}_{\mathbb{C}}(0, \mathbf{I}_N)$ de taille N où $\mathbf{I}_N$ est la matrice identité n et $\mathcal{N}_{\mathbb{C}}(0, \cdot)$ indique une distribution Gaussienne complexe standard. Le récepteur subit un bruit additif blanc Gaussien (AWGN) modélisé par le vecteur aléatoire $\mathbf{z} \sim \mathcal{N}_{\mathbb{C}}(0, \mathbf{I}_{NM})$ de taille NM. Il faut noter que ni la station base, ni les nœuds émetteurs, ne connaissent les réalisations exactes du canal. Mais ils en connaissent les statistiques telles que décrites ici. Pour un nœud actif donné k, les coefficients de canal $h_{m,n}$ sont constants par rapport à m et sont indépendants et identiquement distribués par rapport à n. Cela correspond à des messages envoyés sur un canal à bande étroite tel un sous-canal d'une trame OFDM, comme défini en NB-IoT.

En notant $\mathbf{y} \in \mathbb{C}^{NM}$ le signal reçu, ρ le SNR visé, et $\otimes$ le produit de Kronecker, le signal

reçu est donné par :

$$\mathbf{y} = \sum_{k \in \underline{\mathcal{A}}} \sqrt{\rho}(\mathbf{I}_N \otimes \mathbf{c}_k)\mathbf{h}_k + \mathbf{z}. \tag{3}$$

La station base fait sa détection à partir de $\mathbf{y}$ et connait à l'avance le dictionnaire, la loi de probabilité d'activité, et la statistique de l'état du canal (autrement dit ρ). Notons $\hat{\mathcal{A}}$ l'ensemble des nœuds actifs détectés. Pour pouvoir évaluer la performance des algorithmes étudiés, les métriques suivantes sont utilisées : taux d'erreur d'ensemble des codes (CER), taux d'erreur utilisateur (UER), taux de non détection (MDR) et taux de fausse alarme (FAR) selon les définitions suivantes :

$$\text{MDR :} \quad \bar{\epsilon}_{md} = \mathbb{E}_k \left[\mathbb{P}[k \notin \hat{\mathcal{A}} | k \in \underline{\mathcal{A}}] \right] \tag{4}$$

$$\text{FAR :} \quad \bar{\epsilon}_{fa} = \mathbb{E}_k \left[\mathbb{P}[k \in \hat{\mathcal{A}} | k \notin \underline{\mathcal{A}}] \right] \tag{5}$$

$$\text{UER :} \quad \bar{\epsilon}_s = \bar{\epsilon}_{md} \cdot \theta + \bar{\epsilon}_{fa} \cdot (1 - \theta) \tag{6}$$

$$\text{CER :} \quad \bar{\epsilon}_C = p[\hat{\mathcal{A}} \neq \underline{\mathcal{A}}]. \tag{7}$$

Le MDR correspond aux faux négatifs, le FAR aux faux positifs, le UER combine ces erreurs pour donner un taux d'erreur moyen individuel, et le CER est un taux d'erreur au niveau du système complet qui compte les fois où l'ensemble de codes dans son entier possède au moins une erreur.

3.2. Détecteurs MAP

Le détecteur neural introduit plus bas sera comparé avec des détecteurs de type maximum a posteriori (MAP) pour avoir une comparaison avec des algorithmes traditionnels. Dans le cas où l'objectif est de minimiser le CER, le détecteur optimal est le C-MAP :

Definition 1 (Estimation C-MAP)**.** *L'estimateur C-MAP pour le problème de détection d'ensemble de codes est donné par :*

$$\hat{\mathcal{A}}_C = \operatorname{argmin} \mathcal{A} \in \Phi p[\underline{\mathcal{A}} \neq \mathcal{A} | \mathbf{y}] \tag{8}$$

$$= \operatorname{argmax} \mathcal{A} \in \Phi p(\mathbf{y} | \mathcal{A}) p(\mathcal{A}), \tag{9}$$

Avec $\hat{\mathcal{A}}_C$ le sous-ensemble détecté à partir du signal reçu $\mathbf{y}$ et de la connaissance du livre de codes $\mathcal{C}$, des distributions Gaussiennes des canaux $\mathbf{h}_k$ et du bruit $\mathbf{z}$. Et avec une probabilité à priori donnée par $\mathbb{P}(\mathcal{A}) = \lambda^{|\mathcal{A}|} \cdot (1 - \lambda)^{|\mathcal{U}| - |\mathcal{A}|}$.

Mais si l'objectif est de minimiser le taux d'erreur utilisateur, le risque de Bayes sous-jacent est modifié pour donner l'estimateur optimal suivant :

Definition 2 (Estimation U-MAP)**.** *L'estimateur U-MAP pour le problème de détection d'ensemble de codes est donné par :*

$$\hat{\mathcal{A}}_U = \cup_{k \in \mathcal{U}} \left\{ k | \delta_k(\mathbf{y}) = 1 \right\}, \tag{10}$$

Avec δ la fonction delta et $\delta_k(\mathbf{y}) = 1$ donné par :

$$\delta_k(\mathbf{y}) = \begin{cases} 1 \text{ if } \sum\limits_{\substack{\mathcal{A}\in\Phi;\\ k\in\mathcal{A}}} p(\mathbf{y}|\mathcal{A})p(\mathcal{A}) > \sum\limits_{\substack{\mathcal{A}\in\Phi;\\ k\notin\mathcal{A}}} p(\mathbf{y}|\mathcal{A})p(\mathcal{A}) \\ 0 \text{ else} \end{cases} \tag{11}$$

Le calcul de ces estimateurs nécessite d'évaluer chaque élément de l'ensemble puissance Φ, ce qui rend ce type de détection impraticable en temps raisonnable.

Une alternative à ces solutions est l'utilisation d'un algorithme alternatif. L'It-MAP est construit comme une approximation du C-MAP, sur un principe similaire à l'annulation successive d'interférence (SIC). Il traite le signal reçu de manière itérative et récupère un nouvel utilisateur actif à chaque itération i en se basant sur la précédente estimation $\hat{\mathcal{A}}$ à $i-1$. Le sous-ensemble détecté est construit itérativement, mais la règle de détection est basée sur le critère du MAP donné par eq. (9) pour chaque i, en restreignant la recherche à certains éléments de Φ. Les sous-ensembles à évaluer $\mathcal{A}_i \in \Phi_i$ sont construit à partir du sous-ensemble précédent $\hat{\mathcal{A}}_{i-1}$:

$$\Phi_i = \cup_{k\in\left\{\mathcal{U}\setminus\hat{\mathcal{A}}_{i-1},\emptyset\right\}} \left\{\hat{\mathcal{A}}_{i-1}, k\right\} \tag{12}$$

La détection s'arrête dès que deux itérations successives rendent le même sous-ensemble, c'est à dire $\hat{\mathcal{A}}_{i-1} = \hat{\mathcal{A}}_i$.

3.3. Détecteur par réseau de neurones

Estimateur NN-MAP

L'architecture NN-MAP pour le problème en cours est définie de la manière suivante. Les entrées du détecteur sont $\mathbf{y}$ et ρ (en dB). Il ressort un vecteur $\mathbf{p}$ de longueur K contenant les probabilités estimées que chaque nœud est actif. Cette probabilité est obtenue en utilisant une fonction d'activation sigmoid à la sortie du réseau, et elle est comparée avec le vecteur de labels $\mathbf{t}$ correspondant avec une fonction de perte de cross-entropie binaire pour pouvoir optimiser les paramètres du réseau :

$$\mathcal{L}(\mathbf{t},\mathbf{p}) = -\sum_{k=1}^{K} t_k \cdot \log(p_k) + (1-t_k)\cdot\log(1-p_k) \tag{13}$$

Pendant la phase d'entrainement, le coût moyen sur chaque tuple $(\underline{\mathcal{A}}_i, \mathbf{y}_i, \mathbf{p}_i)$ est, avec I la taille du mini-lot :

$$\bar{\mathcal{L}} = \frac{1}{I}\sum_i \mathcal{L}(\mathbf{t}(\underline{\mathcal{A}}_i), \mathbf{p}_i), \tag{14}$$

Après entrainement, les probabilités sont converties en décisions avec un seuil : un utilisateur k est considéré actif si $p_k > 0.5$. Ce choix est justifié par le théorème suivant, dont la preuve est présentée dans la section 4.3 :

Theorem 1. *Pour un problème d'AUD non cohérent, la solution qui minimise la fonction de coût donnée dans* (14) *converge vers l'estimation U-MAP définie en 2 si il y a suffisamment de données d'entrainement.*

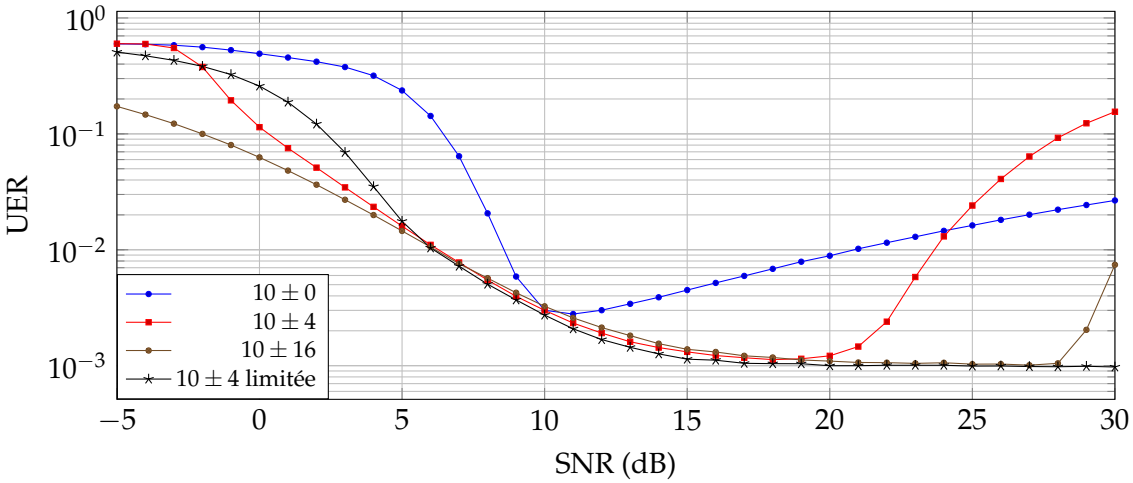

FIGURE 5. : Réseaux entrainés sur des exemples avec des plages variées de SNR. La moyenne est gardée à 10 dB et la largeur de la plage est changée. Pour la ligne limitée, la valeur donnée en entrée au réseau est limitée, au moment du test, aux bornes de l'intervalle d'entrainement.

Paramètres du réseau de neurones

L'architecture du réseau de neurones et les hyper-paramètres sont optimisés de manière empirique de la façon suivante et pour un scénario avec $K = 10$, $M = 8$, $N = 4$, $\lambda = 4$. Un dictionnaire est généré aléatoirement et réutilisé à chaque entrainement pour avoir une comparaison équitable des résultats obtenus.

Type d'architecture : Le vecteur d'observations $\mathbf{y}$ est une combinaison de variables aléatoires Gaussiennes liées au propriétés du dictionnaire, des réalisations du canal, et du bruit. La seule corrélation possible vient du livre de code, qui est défini de manière aléatoire. Donc il y a peu de corrélations dans le vecteur d'entrée, et l'utilisation de couches de convolutions n'est pas utile. De plus, le modèle ne considère pas de corrélations dans les données transmises, ni entre les nœuds, ni temporellement. Les couches récurrentes ne sont donc pas non plus pertinentes. L'architecture choisie est donc celle d'un réseau dense, totalement connecté.

Couches : Un ensemble de réseaux est entrainé en augmentant le nombre de couches denses, en gardant constante la quantité de neurones dans chaque, de 3 à 12 couches. Les réseaux avec plus de 5 couches ne montrent pas d'amélioration significative.

Neurones : Le nombre de neurones par couches est paramétré pour être proportionnel à K, M et N pour qu'il puisse passer à l'échelle suivant la complexité du scénario. Le facteur de proportionnalité est choisi en l'augmentant selon des puissances de deux entre 1 et 128. Les performances s'arrêtent d'augmenter au delà de quatre, donc ce nombre est retenu.

Plage de SNR : Les données d'entrainement sont générées avec un certain ρ, qui est donné en entrée au réseau. Il est donc nécessaire de déterminer les valeurs à utiliser pour

TABLE 2. : Résumé des paramètres du réseau

Paramètre	**Valeur**
Couches	5
Neurones	$4 \times K \times M \times N$
Taux d'apprentissage	1×10^{-3}
Optimiseur	Adam
Taille de lot	4096
Itérations d'entrainement	100000
Plage de SNR	10 dB $\pm$ 16 dB

l'entrainement. Le SNR peut en effet avoir un fort impact sur les performances du réseau : si la valeur est trop faible, le signal peut se retrouver trop bruité pour permettre l'apprentissage , et si elle est trop élevée, le réseau ne doit plus apprendre à gérer la présence de bruit. Dans notre approche, les données d'entrainement sont générées avec des valeurs de SNR distribuées uniformément sur un intervalle spécifique. Pour trouver ce bon intervalle, l'impact de deux paramètres est évalué : la largeur de la plage (figure 5), et sa moyenne. Il apparaît que, pour que le réseau fonctionne efficacement sur un SNR supérieur à sa plage d'apprentissage, il faut limiter la valeur de SNR donnée en entrée à celui-ci, comme montré avec la ligne $-\star-$ où le réseau entrainé sur une plage de 10 ± 4 dB fonctionne bien au dessus de 14 dB. Ce n'est pas le cas en dessous de l'intervalle, où le réseau entrainé sur la plage 10 ± 16 dB bat les autres courbes. Les paramètres importants des réseaux utilisés ici sont résumés dans la table 2.

3.4. Résultats

Comparaison entre les différents algorithmes

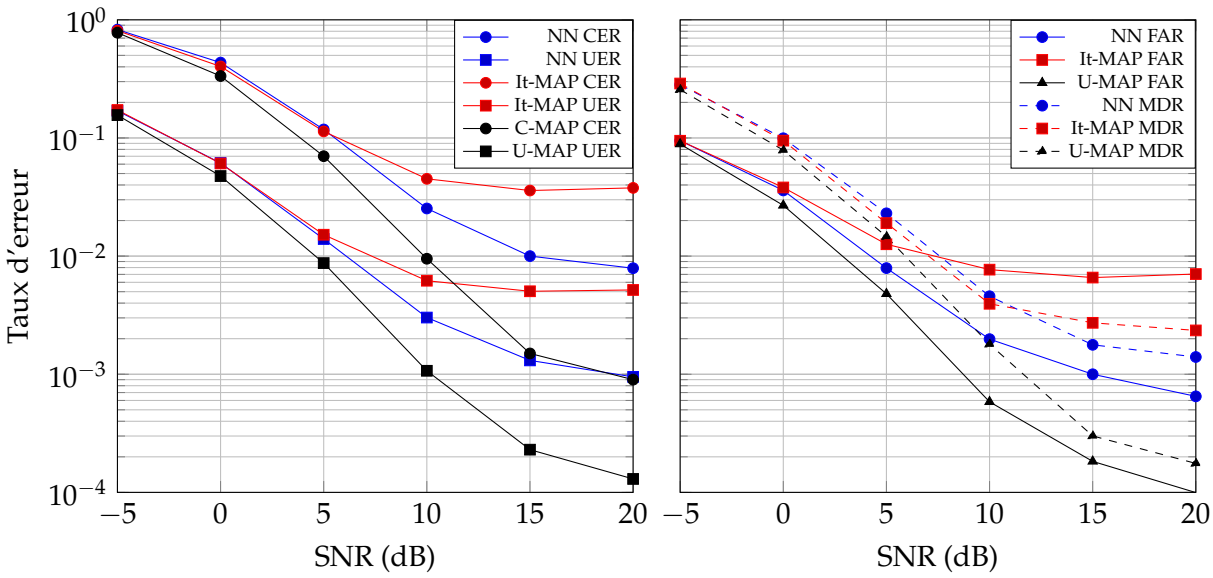

FIGURE 6. : Comparaison de performance entre le C-MAP, le U-MAP, l'It-MAP, et le NN-MAP proposé ici, pour les quatre métriques utilisées

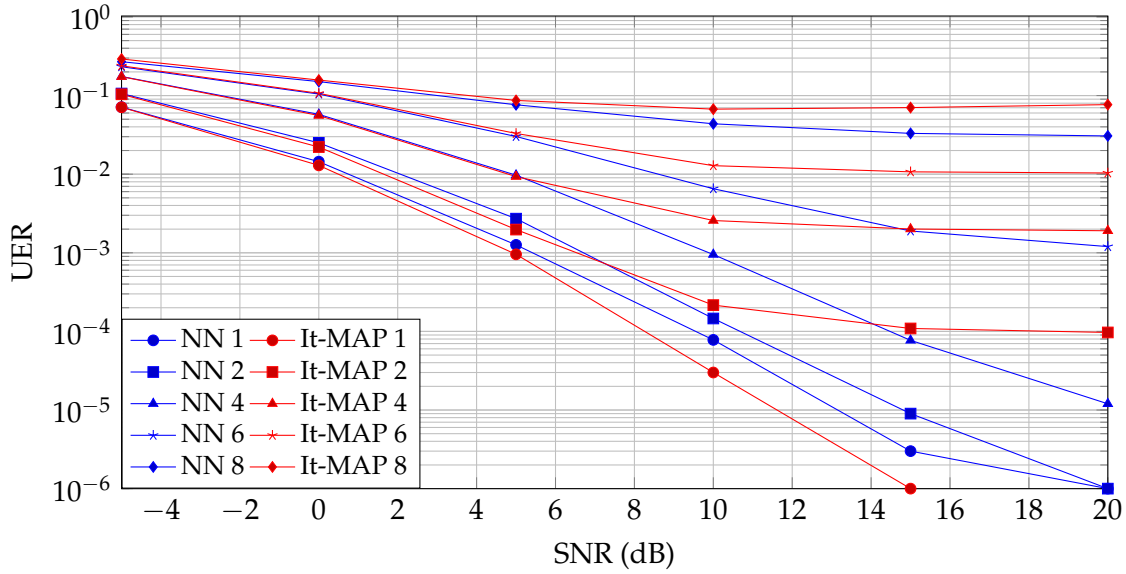

FIGURE 7. : It-MAP et NN-MAP sont paramétrés et entrainés avec $\lambda = 4$. Au moment du test, le nombre d'utilisateurs actifs est fixé et varié de un à huit.

Les différents algorithmes définis précédemment sont comparés pour le scénario $K = 10$, $M = 8$, $N = 4$, $\lambda = 4$. Dans la figure 6, le NN-MAP entrainé est plus performant que le It-MAP, surtout à haut SNR par rapport aux métriques de UER et de CER. La performance de ce NN-MAP est à mi-chemin de l'optimal correspondant aux C-MAP et U-MAP. De plus, le NN-MAP bat le It-MAP à la fois pour le MDR et le FAR. On peut noter que le MDR de l'It-MAP est plus bas que son FAR, tandis que c'est l'inverse pour le NN-MAP. Il serait possible d'ajuster la fonction de perte à l'entrainement du réseau de neurones pour changer l'équilibre entre MDR et FAR, éventuellement au prix de la performance en UER.

Tous les algorithmes testés ici ont la connaissance préalable de λ, soit explicitement, soit au travers de l'apprentissage. La figure 7 montre comment l'It-MAP et le NN-MAP se comportent quand le nombre réel d'utilisateurs actifs diffère de la valeur attendue. L'It-MAP est plus performant que le NN-MAP uniquement pour un seul utilisateur actif. À ce moment là, l'It-MAP n'a qu'une itération a effectuer, et est donc équivalent au MAP non itératif qui est optimal. Il serait possible d'améliorer les performances du réseau de neurones sur cet aspect en l'entrainant sur diverses valeurs de λ, ou en utilisant une distribution uniforme plutôt qu'une loi de Poisson pour l'activité au moment de l'entrainement.

Scénarios plus grands

Dans la figure 8, les résultats de performance de l'It-MAP et du NN-MAP sont donnés avec une augmentation de K. Dans ce scénario, les U-MAP et C-MAP ne sont plus calculables en temps raisonnable. Ici, le NN-MAP est meilleur que l'It-MAP jusqu'à $K = 20$ mais les deux saturent au même niveau au delà.

Le réseau de neurones peut être adapté pour des scénarios encore plus gros, avec $K = 100$ mais là même l'It-MAP devient trop long à exécuter. La longueur des codes devient très importante sur la performance finale du système.

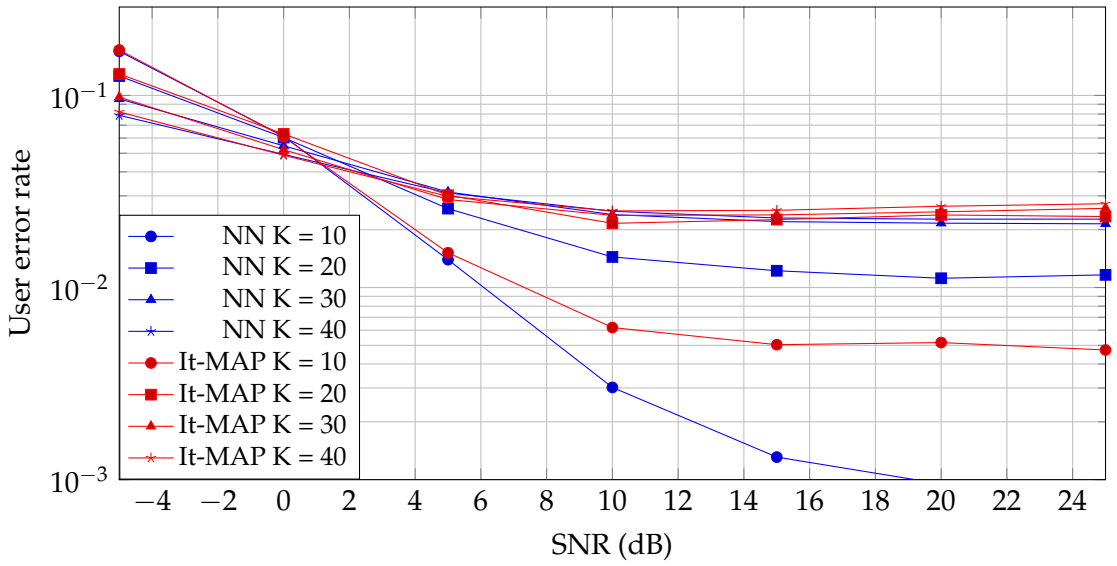

FIGURE 8. : Comparaison d'UER entre NN-MAP et It-MAP en augmentant le nombre d'utilisateurs potentiels. $\lambda = 4, M = 8, N = 4$.

TABLE 3. : Temps de calcul moyen sur un scénario avec $K = 10$, $M = 8$, $N = 4$, $\lambda = 4$.

MAP CPU	It-MAP CPU	NN CPU	NN GPU
159 ms	8.25 ms	3.6 ms	472 ns

Temps de calcul

En plus de ses performances, l'approche par réseau de neurones permet une réduction du temps de calcul. Cette réduction a deux sources : le calcul est plus simple, n'utilisant que des opérations simples d'addition et de multiplication, sans opérations de branchement, ce qui mène à une charge réduite et une exécution plus fluide par le système, et il permet aussi de faire des calculs par lot massivement parallèles de manière efficace. Ce second aspect est le plus visible. L'It-MAP se base sur des opérations séquentielles avec des boucles imbriquées qui peut créer des erreur dans les prédictions de branchement du processeur et qui ne sont pas triviales à convertir vers un calcul parallèle permettant la sous traitance par une carte graphique. Il est donc limité à une exécution sur processeur, tandis que le NN-MAP peut être exécuté sur carte graphique, par lot de 100000.

3.5. Conclusion

Dans cette partie, l'utilisation de l'apprentissage profond a été proposée pour le problème de la détection d'utilisateurs actifs non-cohérent basé sur le NOMA codé. Le détecteur NN-MAP qui minimise la fonction de coût MAP permet d'améliorer, ou au moins d'égaler la performance d'algorithmes itératifs, surtout sur la métrique d'UER. Il permet aussi de réduire la complexité algorithmique. Cela montre l'intérêt de l'apprentissage profond pour ce genre de taches, même si il reste encore à améliorer le passage à l'échelle de l'architecture vers des ensembles de nœuds plus importants.

4. Apprentissage de constellations

Après avoir étudié le problème de détection des utilisateurs actifs, l'étape suivante est de faire en sorte que ces utilisateurs actifs puissent transmettre leurs données efficacement. Cette partie s'intéresse au fait de concevoir des constellations de modulation appropriées à des conditions d'accès multiples pour maximiser le débit transmit.

4.1. Façonnage de constellations

En point à point

Dans les communications numériques, le terme de constellation décrit le mappage de *messages* qui sont des séquences de bits d'une certaine longueur, aussi appelé l'*ordre* de la constellation, en un tuple de N_{real} nombre réels à transmettre. Chaque tuple de nombre réel est appelé un *symbole* et il y a $2^{N_{bpm}}$ symboles dans une constellation d'ordre N_{bpm}. Le fait de concevoir une constellation spécifique peut être appelé façonnage de constellation.

Les constellations traditionnelles opèrent avec $N_{real} = 2$ pour décrire les deux composantes d'un symbole complexe, ce qui peut être représenté sur un plan, menant à la visualisation d'un nuage de points ressemblant une constellation, d'où le nom.

L'objectif principal du façonnage est de réussir à placer les symboles de manière à ce que, après avoir subit les distorsions du canal, les points se retrouvent le plus loin possible les uns des autre pour faciliter le décodage. La principale contrainte à ce placement vient du fait que la puissance d'émission est limitée par des éléments extérieurs (puissance disponible, matériel, protocole, législation). Les constellations sont donc généralement conçues avec une puissance moyenne de un, laissant la gestion de la puissance à d'autres éléments. Cela implique que, si certains symboles prennent des valeurs élevées pour s'éloigner des autres, les autres doivent se rapprocher de zéro pour maintenir la puissance moyenne.

En plus de gérer la disposition des symboles, le concepteur d'une constellation peut aussi avoir à décider de l'affectation des messages à chaque symbole. Dans le cas où $N_{bpm} > 1$, une erreur de décodage de symbole impacte plus d'un bit, et le problème de labellisation des bits devient important. Une labellisation idéale, ou de Gray, fait en sorte que des symboles voisins ne diffèrent que d'un bit. Les constellations d'ordre élevé peuvent rendre cette opération difficile à cause de l'augmentation du nombre de combinaisons possibles.

Il existe de nombreuses constellations conçues pour des communications point à point (P2P), et capable de s'approcher de la capacité du canal. Mais elles sont adaptées à des canaux simples et ne s'adaptent pas de manière continue à l'évolution de la capacité en fonction du niveau de bruit. Pour remédier à cela, un certains nombre de travaux ont cherché à utiliser de l'apprentissage profond pour apprendre des constellations adaptées à un canal, et un niveau de bruit donné, en implémentant un auto-encodeur représentant la chaine de transmission complète, émetteur, canal, récepteur, et en optimisant les deux parties comme un tout.

En accès multiple

Les systèmes P2P auxquels correspondent ces constellations impliquent que transmetteur et récepteur sont seuls dans leur bande de fréquence, sans interférence externe. Comme mentionné dans les parties précédentes, la forte densité d'objets voulant communiquer demande de partager les ressources radio. La méthode classique pour gérer cette condition d'accès multiple consiste en la division de la ressource partagée en créneaux de temps et/ou

de fréquence, et d'en allouer un à chaque utilisateur, pour pouvoir se retrouver dans les conditions du P2P. Ces méthodes sont l'accès multiple par division en fréquence (FDMA), et en temps (TDMA). Le TDMA possède deux variantes : celle à contrainte de puissance maximum, chaque utilisateur émet à son maximum une certaine fraction du temps, et à contrainte de puissance moyenne, où chaque utilisateur maintient une puissance moyenne globale. Dans ce cas, moins l'utilisateur transmet souvent, plus il peut émettre fort et donc augmenter son SNR. Cette deuxième option est moins réaliste puisque la puissance est généralement limitée par soit la capacité des amplificateurs, soit la législation. Le FDMA se comporte comme le TDMA à puissance moyenne. Celle si se concentre quand la bande utilisée se réduit, augmentant également le SNR. Ces approches orthogonales (OMA) permettent de se rapporter aux conditions P2P bien étudiées, mais elles limitent le nombre d'utilisateurs qui peuvent être servis simultanément, et demande beaucoup de messages de contrôle pour allouer les créneaux.

L'approche non-orthogonale (NOMA) alloue un créneau à plusieurs utilisateurs, permettant une augmentation de la capacité théorique. Cela permet aussi de réduire la latence induite par l'attente d'un créneau libre. Ces deux éléments rendent cette approche très intéressante, notamment pour la canal montant. Des constellations adaptées sont donc nécessaires.

Un certain nombre de travaux étudient le comportement de constellations de taille finie dans des conditions de NOMA montant, à la fois pour des canaux Gaussiens et de Rayleigh. Mais aucun ne s'est intéressé à la conception de constellations à base d'apprentissage profond sur un canal de type Rayleigh, et surtout sans s'intéresser à l'étude de la comparaison des performances de l'approche OMA et du NOMA, ni aux compromis de débit introduit par la présence de deux ou plus utilisateurs en compétition pour une même ressource. Ce travail s'intéresse donc à cette comparaison et à ce compromis, en se basant sur des constellations apprises avec de l'apprentissage profond, sur des canaux Gaussiens, et de type Rayleigh dans un cadre à deux utilisateurs.

4.2. Modèle

Le système simulé possède deux utilisateurs, TX 1 et TX 2, à des positions différentes, transmettant simultanément sur un canal dont ils ne connaissent que la statistique, à un récepteur unique RX. Ils sont toujours actifs, et le récepteur le sait. Les deux utilisateurs envoient les signaux complexes x_1 et x_2 avec une puissance moyenne de un. Puisqu'ils ne sont pas au même endroit, ces signaux subissent des conditions différentes, représentées par les puissances reçues P_1 et P_2. Ils sont également affectés par les coefficients de canal complexes h_1 et h_2 de propriétés différentes selon le scénario, mais avec une puissance moyenne à un. ($\mathbb{E}\left[||h||_2\right] = 1$) Ils sont ensuite additionnés, et ajoutés à un bruit Gaussien $n \sim \mathcal{N}_{\mathscr{C}}(0,1)$, créant le signal reçu y :

$$y = (x_1 \times \sqrt{P_1} \times h_1) + (x_2 \times \sqrt{P_2} \times h_2) + n \tag{15}$$

Puisque $\mathbb{E}\left[||n||_2\right] = 1$, les valeurs de SNR correspondent directement aux contraintes de puissance reçue $SNR_1 = P_1$ et $SNR_2 = P_2$.

Deux scénarios sont étudiés, un avec bruit blanc Gaussien ($h_1 = h_2 = 1$) qui permet, entre autre de faire une comparaison avec des travaux précédents, et un où les coefficients de canal sont tirés d'un modèle OFDM et corrélés selon un modèle de Jakes.

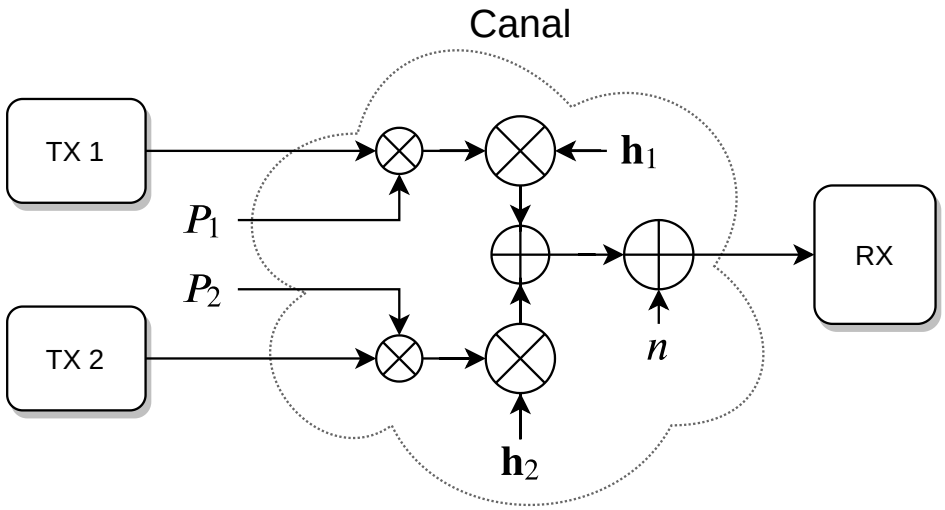

FIGURE 9. : Modèle de transmission. Dans le scénario Gaussien, $\mathbf{h}_1 = \mathbf{h}_2 = 1$ et dans l'autre $\mathbf{h}_1$ et $\mathbf{h}_2$ viennent d'un modèle OFDM Rayleigh corrélé.

4.3. Implémentation

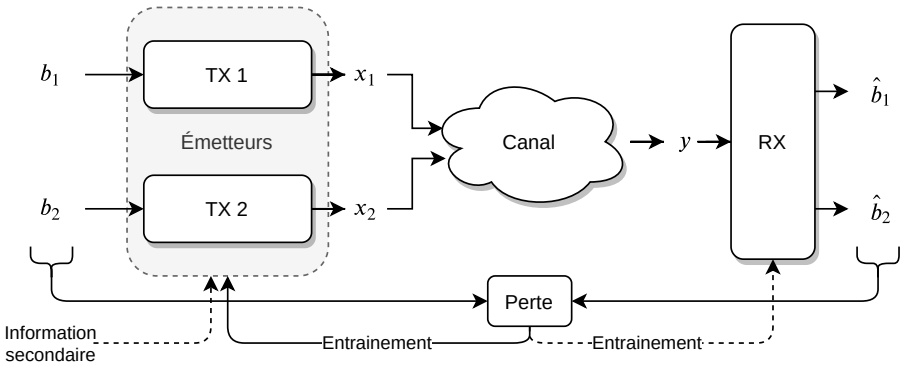

FIGURE 10. : Contexte d'entrainement

Comme décrit dans la figure 10, deux flux de bits b_1 et b_2 sont générés et donnés aux émetteurs. Dans ceux-ci, N_{bpm} bits sont assemblés pour former les flux de messages m_1 et m_2 puis encodés pour donner les signaux à transmettre x_1 et x_2. Il passent ensuite dans le canal et y est récupéré au récepteur. Celui-ci en estime les bits transmis $\hat{b}_1$ et $\hat{b}_2$. Le flux originel de bits est comparé avec l'estimation par la fonction de perte, et la perte qui en résulte est envoyée au transmetteurs (et au récepteur) pour les mettre à jour. La forme et la taille des flux gérés dépend du scénario choisi, ils sont gérés par lot d'examples et ces exemples correspondent, soit à un symbole de constellation (en Gaussien), soit à une trame OFDM.

Métriques

Les métriques classiquement utilisées pour comparer les constellations sont les taux d'erreur binaire (BER) et d'erreur symbole (SER). Dans le cas présent, une métrique plus intéressante se présente sous la forme de l'information mutuelle par bit (BMI) qui se trouve être un débit atteignable pour un système de décodage de bits tel qu'étudié ici [149].

$$R_{BMI} := H(X) - \sum_{i=1}^{N_{bpm}} H(B_i|Y) \tag{16}$$

Il s'avère que $H(B_i|Y)$ peut-être estimé par la fonction de perte standard de cross-entropie binaire. Donc, en l'utilisant au niveau du récepteur, le débit atteignable R_{TX} pour la transmission d'un émetteur peut être mesuré.

$$R_{TX} = \frac{N_{bpm}}{N_{cu}}(1 - \mathcal{L}), \tag{17}$$

Avec $\mathcal{L}$ la perte de cross-entropie moyennée sur un certain nombre d'examples.

Ces métriques de taux d'erreur et de débit sont liées à un unique émetteur. La grande majorité des travaux existant dans le domaine étudié ici utilise une métrique agrégée pour tout combiner un seul nombre, avec un BER moyen, ou un débit total. Cela permet d'avoir des résultats plus pratiques et de n'avoir qu'une dimension à étudier. Mais cela réduit l'information disponible en cachant les compromis possibles entre les performances des deux utilisateurs. Par exemple, pour un débit total donné de 4 bits par utilisation de canal, on ne peut pas distinguer si $R_1 = R_2 = 2\,\text{bit/cu}$, $R_1 = 4\,\text{bit/cu}$ ou $R_2 = 0\,\text{bit/cu}$. L'opérateur peut aussi vouloir satisfaire des aspects de qualité de service, par exemple s'assurer que $R_2 > 0.5\,\text{bit/cu}$, avant de chercher à maximiser la performance globale. C'est pour ça que les métriques présentées ici ne sont pas agrégées mais présentées sous forme de paires de débits sur un plan 2D, traçant une région de débit.

Sur un canal Gaussien, les débits théoriques sont connus, à la fois pour le OMA, et pour le NOMA. Et en OFDM, une borne supérieur est utilisée pour le TDMA sous la forme d'un débit atteignable avec information complète de l'état du canal, accompagné d'une affectation des bits appropriée au transmetteur.

Pour avoir une comparaison équitable avec l'approche orthogonale, celle-ci sera représentée par des constellations apprises et optimisées sur le canal P2P, ce qui permet d'avoir une performance au moins identique aux constellations classiques, tout en s'adaptant finement au niveau de bruit.

Fonction de perte conjointe

La sélection de la cross-entropie binaire permet de s'assurer que les réseaux appris maximisent l'information mutuelle binaire, et donc la métrique de débit qui nous intéresse. L'aspect d'accès multiple demande cependant d'ajouter une réflexion supplémentaire. Les deux trans-

missions parallèles de TX 1 et TX2 mènent à deux valeurs de perte $\mathcal{L}_1$ et $\mathcal{L}_2$.

$$\mathcal{L}_1 = -\sum_{n=1}^{N_{bpm}} b_{1_n} \cdot \log\left(\hat{b}_{1_n}\right) + (1 - b_{1_n}) \cdot \log\left(1 - \hat{b}_{1_n}\right) \tag{18}$$

$$\mathcal{L}_2 = -\sum_{n=1}^{N_{bpm}} b_{2_n} \cdot \log\left(\hat{b}_{2_n}\right) + (1 - b_{2_n}) \cdot \log\left(1 - \hat{b}_{2_n}\right) \tag{19}$$

$$\tag{20}$$

Ces deux valeurs sont indépendantes et doivent être agrégées en une valeur unique pour entrainer le système. Cela permet aussi de cibler une zone spécifique de la région de débit sans rester limité à une situation avec des débits égaux.

Définissons un paramètre $\alpha \in [0, 1]$ pour contrôler le compromis entre les deux émetteurs, avec $\alpha = 1$ correspondant à tout donner à TX 1, $\alpha = 0$ à TX 2, et $\alpha = 0.5$ une situation équilibrée. Ce paramètre est utilisé définir un vecteur $\mathbf{u}$ unitaire et y projeter le vecteur $\mathbf{r}$ correspondant à la paire de débits obtenus.

$$\mathbf{u} = \begin{pmatrix} \frac{1}{\sqrt{1+s^2}} \\ \frac{s}{\sqrt{1+s^2}} \end{pmatrix} \mathbf{r} = \begin{pmatrix} R_1 \\ R_2 \end{pmatrix} \tag{21}$$

Cela donne un débit total projeté qui doit être inversé pour obtenir une valeur à minimiser. Pour donner une information contextuelle à la fonction de perte, celle-ci est comparée à la capacité Gaussienne correspondant aux niveaux de SNR rencontrés.

$$R_{1_{proj}} = \mathbf{u}_x \times \mathbf{u} \cdot \mathbf{r} \tag{22}$$

$$R_{2_{proj}} = \mathbf{u}_y \times \mathbf{u} \cdot \mathbf{r} \tag{23}$$

$$R_{projsum} = R_{1_{proj}} + R_{2_{proj}} \tag{24}$$

Émetteur

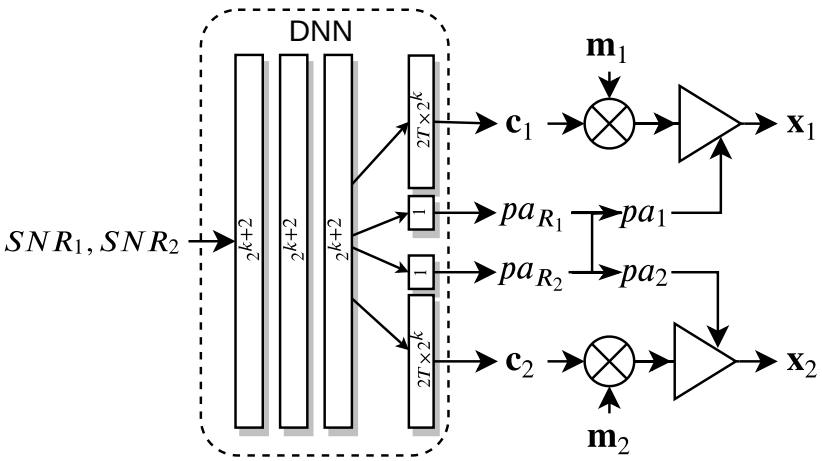

FIGURE 11. : Transmetteur neural. $k = N_{bpm}$ et $T = N_{cu}$.

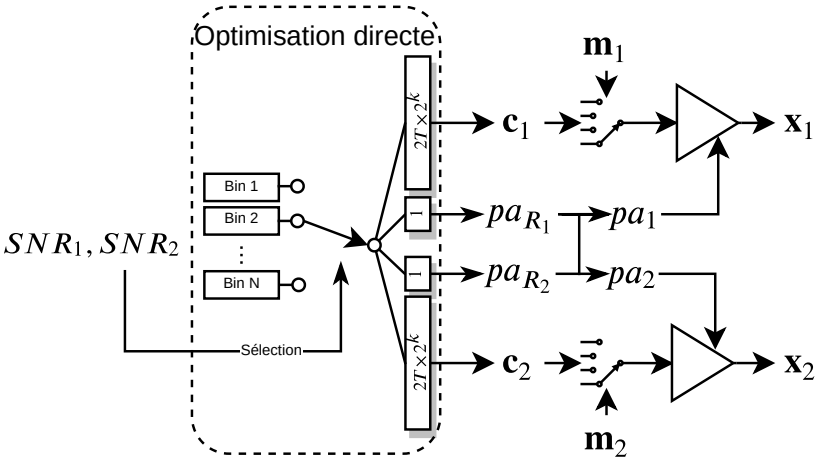

FIGURE 12. : Transmetteur à optimisation directe. $k = N_{bpm}$ et $T = N_{cu}$.

Deux designs de transmetteur sont proposés, un basé sur un réseau de neurones (NN), et l'autre à optimisation directe (DO). Ils consistent en un système joint apprenant les constellations des deux émetteurs au même endroit. Ils utilisent en entrée le SNR des deux utilisateurs et ressortent une constellation, et un nombre de requête d'allocation de puissance $pa_R \in [0, 1]$ pour chaque, utilisé pour encoder les messages à transmettre $\mathbf{m}_1, \mathbf{m}_2 \in \left[0, 2^{N_{bpm}}\right[$. Les vecteurs de constellation font $2 \times 2^{N_{bpm}}$ éléments de long, avec des paires de valeurs réelles pour les composantes réelles et imaginaires des symboles complexes à transmettre. Le fait d'optimiser la constellation complète plutôt que d'avoir un système apprenant un encodage symbole par symbole permet de simplifier l'architecture et de résoudre certains problèmes d'implémentation, comme décrit dans la section 5.4.3.

Le transmetteur NN implémente un réseau de neurones dense avec trois couches pleinement connectées opérant avec le SNR en entrée. Le transmetteur DO, lui, contient un ensemble de matrices de constellations, chacune affectée à une plage de SNR. Pour un SNR donné, la plage correspondante est sélectionnée et la matrice correspondante est optimisée directement, par descente de gradient.

L'approche DO est plus simple avec moins de paramètres à entrainer, mais la quantité de plages est fixée, alors que le NN permet de parcourir l'espace des SNR de manière continue. Réduire la largeur des plages permet de mieux s'adapter à un SNR précis, mais cela augmente le temps d'entrainement et le besoin mémoire.

Les scalaires pa_R également appris permettent à un utilisateur de réduire sa puissance d'émission pour donner plus de place à la constellation de l'autre. Les requêtes pa_{R1} et pa_{R2} sont transformées en allocations de puissance $pa_1, pa_2 \in \left[0, 2^k\right]$ comme suit :

$$pa_1 = \frac{pa_{R1}}{\max pa_{R1}, pa_{R2}}, \quad pa_2 = \frac{pa_{R2}}{\max pa_{R1}, pa_{R2}} \tag{25}$$

Bien que la fonction de perte utilisée et son paramètre α permettent de viser une zone l'espace des débits, la performance obtenue après un apprentissage reste aléatoire. Il y a une certaine dispersion des paires de débits obtenues après l'entrainement de plusieurs transmetteurs, même avec des paramètres identiques. Pour gérer cette limitation, plusieurs transmetteurs sont entrainés pour 21 valeurs de α entre zéro et un. Tous les transmetteurs

résultants sont évalués et ceux se situant sur l'enveloppe de Pareto sont retenus et présentés dans la section résultats.

Récepteur

Ce travail s'intéresse spécifiquement à l'optimisation des constellations au niveau de l'émetteur, mais un système a besoin d'un récepteur pour décoder les signaux transmis et calculer une perte à partir des bits estimés. Pour cela, deux récepteurs sont utilisés. Le premier, appelé *MLRX* est un décodeur classique, non appris, basé sur un maximum de vraisemblance (ML), qui utilise la connaissance de la constellation complète (rendu possible par l'architecture du l'émetteur) pour décoder le signal et sortir un logarithme de rapport de vraisemblance (LLR) pour chaque bit possible. La constellation complète utilisée ici est la combinaison des deux constellations transmises, soit l'ensemble des points pouvant résulter de l'addition de deux symboles transmis. Dans le cas du scénario OFDM corrélé, ce récepteur commence par opérer une estimation de canal avec une minimisation d'erreur des moindres carrés (LMMSE) avant le décodage ML. Cette estimation se base sur la conaissance de la matrice de covariance du canal, et sur des pilotes insérés dans la trame OFDM. Ces pilotes sont placés de manière orthogonale, ce qui double la quantité de créneaux pris par des pilotes par rapport à une trame en TDMA et réduit donc plus la perte de débit. Les LLRs calculés passent dans une fonction d'activation sigmoïde pour obtenir l'estimation des bits $\hat{b}_i = p(b_i = 1|y)$ et calculer la perte.

Le second récepteur, appelé *DLRX*, est uniquement utilisé dans le scénario OFDM et est un réseau résiduel de convolution. Il opère sur une trame entière, donc son entrée est un tenseur de forme $N_T \times N_S \times 2$ (deux canaux pour les composants I et Q), et sa sortie est un tenseur de taille $N_T \times N_S \times 2\frac{N_{bpm}}{N_{cu}}$, la matrice des bits correspondant à chaque élément. Les premières et dernières couches sont des couches de convolution 2D seules, tandis que les couches intermédiaires sont organisées en modules résiduels composés de deux convolutions 2D et deux normalisations par lot alternées. En P2P, ce type d'architecture permet d'opérer sans aucun pilote dans la trame ([127]). Les constellations apprises dans ce contexte contiennent une légère asymétrie permettant l'estimation de canal. Le contexte actuel du multi-accès, crée un entrainement trop complexe pour permettre une convergence efficace sans pilotes.

4.4. Résultats

Scénario Gaussien

Le travail le plus proche du système étudié ici [134] s'intéresse à un canal Gaussien avec deux utilisateurs et un maximum de deux bits par utilisateur. Il propose une constellation unique dans ce cas, cherchant à maximiser le débit moyen. L'approche proposée ici permet d'obtenir, à la fois de meilleurs performances, mais aussi de parcourir toute la région de débit sans être limité à un seul point. On peut par contre noter que, dans les conditions du papier en question, aucune des deux approches ne permet de gagner en performance par rapport à du TDMA avec puissance moyenne.

Ce manque de performance par rapport au TDMA est lié à la symétrie qu'il y a entre les puissances des deux émetteurs. Le suivi de l'évolution des régions de débit en fonction de la différence de puissance montre qu'un faible niveau d'asymétrie ne permet pas aux

constellations apprises de dépasser le TDMA à puissance moyenne, mais qu'à partir d'une différence de 6 dB entre P_1 et P_2, on peut observer des gains de performance.

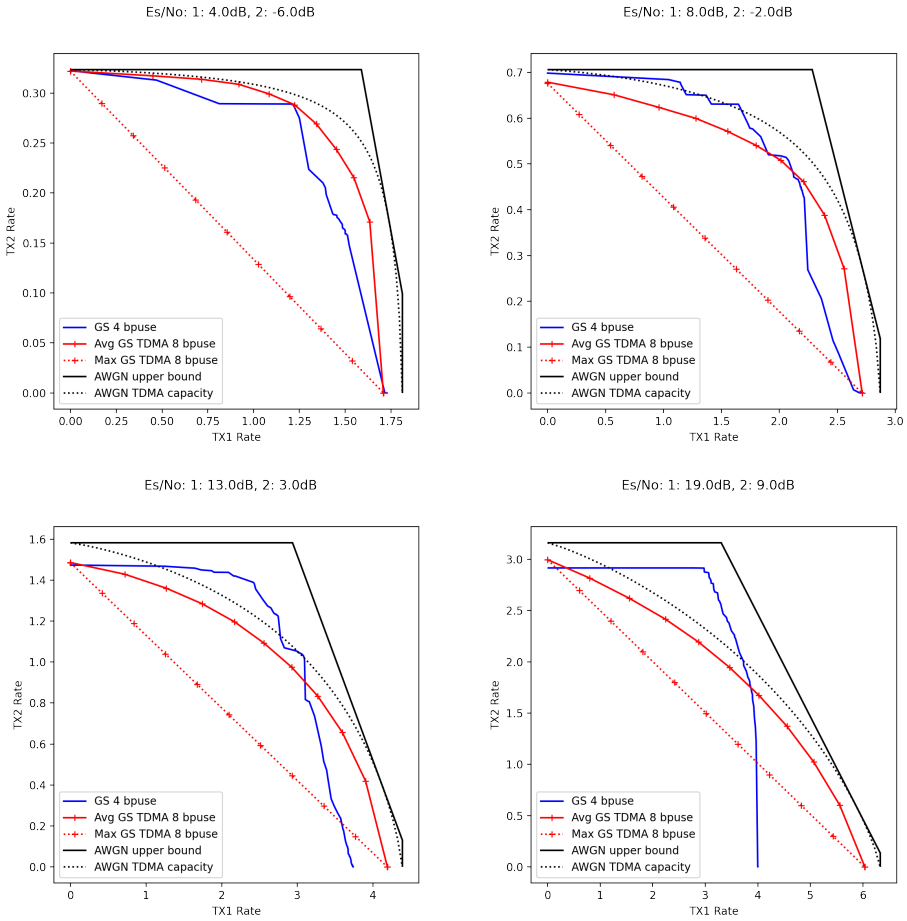

FIGURE 13. : Régions de débit avec $P_1 = P_2 + 10\,\text{dB}$. Scénario Gaussien, transmetteur NN, $N_{bpm} = 4$, $N_{cu} = 1$, comparé avec transmission 8bit TDMA apprise (GS)

Dans la figure 13, une différence de 10 dB est utilisée tandis que le niveau de SNR est augmenté. Pour un fort niveau de bruit pour TX 2, les constellations obtenues s'approchent de la performance du TDMA moyen, mais elles gagnent rapidement en débit et sortent même de la région de capacité théorique du TDMA moyen. A haut SNR, et puisque $N_{bpm} = 4$, un maximum de quatre bit peuvent être envoyé à la fois, ce qui limite le débit possible et fait saturer la courbe. Pour monter au delà, il faudrait augmenter l'ordre de la constellation. On peut aussi noter que, dans tous les cas, les constellations apprises donnent de meilleurs débits que le TDMA à puissance maximale, qui est la plus réaliste des deux variantes.

La figure 14 montre un exemple de constellation obtenue dans le dessus de la région de débit de la figure 13 avec $P_1 = P_2 + 10\,\text{dB}$, $P_2 = 3\,\text{dB}$. La partie supérieure montre les constellations isolées et leur combinaison tandis que la partie inférieure montre les régions de décision du décodeur MLRX pour chaque bit de chaque utilisateur. La couleur rouge est associée à un bit à 0, le bleu à 1, et les blancs (et les couleurs pâles) montrent les zones ambiguës. La figure correspond à la partie supérieure, où plus d'importance est donnée au débit de TX 2.

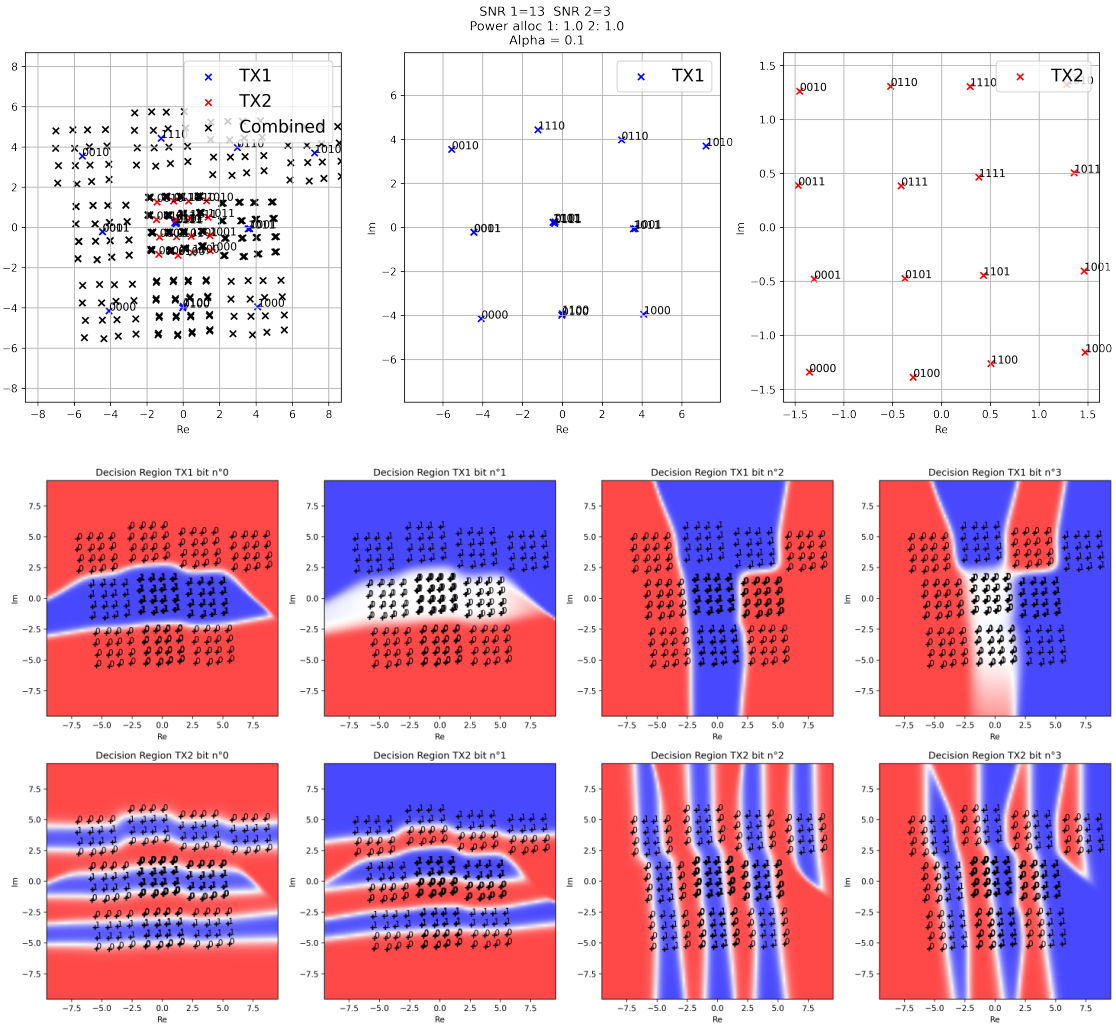

FIGURE 14. : Constellations obtenues avec $P_1 = P_2 + 10\,\text{dB}$, $P_2 = 3\,\text{dB}$ avec les régions de décision qui correspondent. Scénario Gaussien, transmetteur NN, $N_{bpm} = 4$, $N_{cu} = 1$. Débit correspondant : $R_1 = 1.9$, $R_2 = 1.44$

La constellation pour cet utilisateur est très proche d'une 16-QAM, avec tous les symboles également espacés, tandis que TX 1 a une constellation avec des symboles superposés, ce qui mène à de l'ambigüité sur les bits 1 et 4. Cette réduction dans le nombre de symboles différents donne plus d'espace pour décoder les bits de TX 2. L'allocation de puissance est à 1 pour les deux émetteurs, la constellation de TX 1 a ses symboles suffisamment loin les uns des autres pour faire rentrer la constellation de TX 2.

La comparaison des performances des deux types de transmetteurs (NN et DO) proposés montre que le NN a de meilleures performances dans la plupart des cas, surtout à haut SNR et par rapport au débit de TX 2 (celui avec les moins bonnes conditions de canal). Il permet également de parcourir une région de débit de manière plus continue, ce qui permet de gérer plus finement le compromis de débit requis. Le DO peut toujours être utile dans des conditions où l'espace mémoire est très restreint puisqu'il nécessite beaucoup moins de paramètres.

Scénario OFDM corrélé

TABLE 4. : Paramètres OFDM

Paramètre	Valeur
Sous porteuses	72
Espacement des sous-porteuses	15 kHz
Symboles OFDM	14
Fréquence porteuse	2.6 GHz
Étalement des délais	100 ns
Vitesse	1.4 m/s

Pour évaluer les performances sur ce scénario, la même différence de puissance est utilisée, avec le transmetteur NN. La figure 15 montre les résultats obtenus dans ce cas. Une faible quantité de pilotes est utilisée (18 par utilisateur) de manière à tester l'estimation de canal des deux types de récepteurs, tandis que la version orthogonale est poussée à son maximum de débit en opérant sans pilotes. Les débit prennent en compte la perte de débit lié à ces pilotes. Les régions de capacité Gaussiennes sont présentées pour comparaison mais ne sont pas atteignables ici.

Une première observation est que le récepteur DLRX permet une meilleure performance, ayant même une zone meilleure que le TDMA à puissance moyenne, même si les gains sont moindres qu'en Gaussien. Une seconde observation vient en comparant les courbes MLRX et DL/MLRX. Cette dernière est obtenue en prenant les constellations donnant la courbe du DLRX, et en les testant avec le récepteur ML. Les constellations qui donnent les meilleures performances en DLRX ne sont pas les même que celles qui donnent les meilleures performances en MLRX. Cela indique que les bonnes constellations pour le DLRX contiennent des éléments qui aident le récepteur à estimer le canal, au prix d'une réduction de la distance entre les symboles. Et que l'amélioration dans l'estimation de canal est plus utile que la réduction en décodage. Puisque le MLRX n'utilise que les symboles de pilotes pour l'estimation, il ne voit que la réduction de performance de décodage.

La figure 16 est un exemple de constellation dans ce cas (marqueur supérieur). Ici aussi, certains symboles sont superposés, sacrifiant certains bits, pour améliorer la capacité de

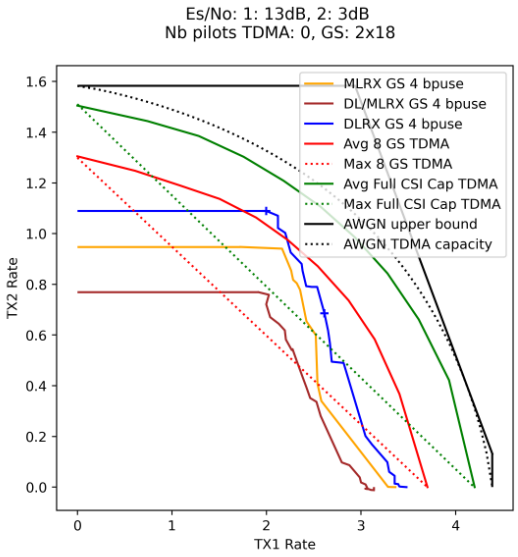

FIGURE 15. : Régions de débit avec $P_1 = P_2 + 10$ dB. Scénario OFDM corrélé, transmetteur NN avec DLRX et MLRX et 18 pilotes par utilisateurs. Comparé avec transmission 8bit TDMA apprise et 0 pilotes.

détection des autres. On peut noter qu'il peut y avoir des bits partiellement sacrifiés, comme le bit numéro 1 de TX 1.

4.5. Conclusion

On a montré qu'un réseau de neurones dense peut apprendre des constellations adaptées à un scénario NOMA montant à deux utilisateurs, avec des gains par rapport, à la fois aux approches TDMA classiques, et à l'état de l'art. Et ce, même par rapport au système optimiste et moins réaliste du TDMA avec puissance moyenne, et en conditions de canal Rayleigh. L'étude de ce système multi-accès demande de considérer des compromis qui ne peuvent pas être capturés efficacement en regardant seulement des métriques agrégées telles que le débit total.

On note que l'entrainement est difficile dans les conditions d'évanouissement de Rayleigh, mais il serait possible de trouver des gains en améliorant le processus d'entrainement, et de réduire le besoin de pilotes, peut-être jusqu'à l'éliminer.

5. Conclusion

Cette thèse a exploré les usages de l'outil d'apprentissage profond dans le cadre de tâches de détection pour la couche physique de communications sans fil à utilisateurs multiples, en travaillant soit dans des conditions faiblement modélisées (Chapitres 3 et 5) ou en améliorant la performance et la vitesse d'algorithmes traditionnels (Chapitre 4). L'apprentissage profond est un outil bien adapté à cette situation, surtout que ces tâches de détection sont réalisées fréquemment, une fois par paquet reçu, ou même par symbole reçu. Cela fait que de grandes quantités de données peuvent être générées et labellisées pour alimenter des algorithmes

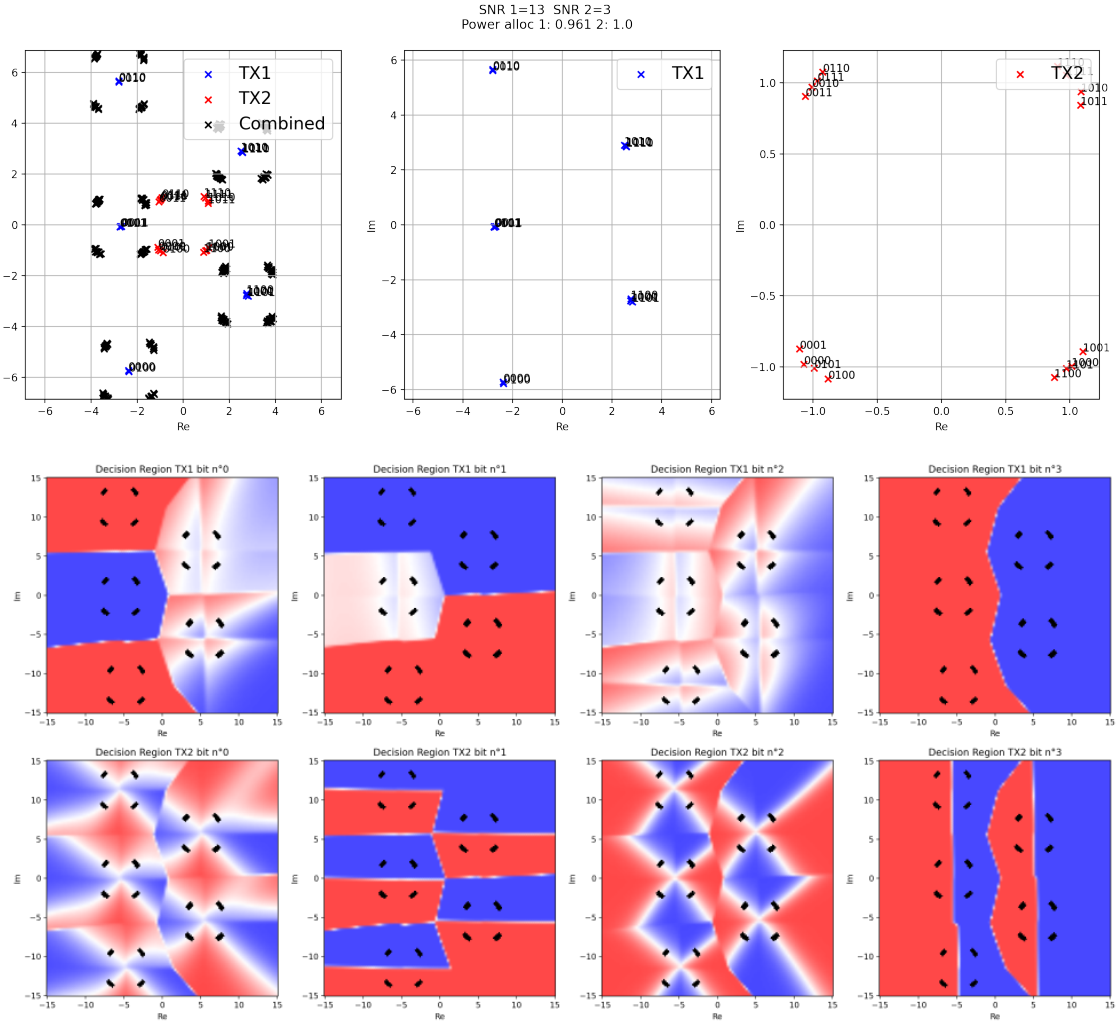

FIGURE 16. : Constellations obtenues avec $P_1 = P_2 + 10\,\text{dB}, P_2 = 3\,\text{dB}$ avec les régions de décision qui correspondent. correlated OFDM scenario, Scénario OFDM corrélé, transmetteur NN avec DLRX. Débit correspondant : $R_1 = 2.07, R_2 = 1.13$, marqueur supérieur de la figure 15.

d'apprentissage. C'est un outil puissant qui peut permettre d'éviter d'avoir à construire des modèles, mais peut-être très complexe et long à implémenter à cause de la difficulté à trouver des architectures neurales performantes et à en ajuster les hyper-paramètres. De plus, les approches par apprentissage développent leur potentiel en travaillant sur des points de données réels. La conception et l'implémentation d'expériences avec des banc d'essais et du véritable matériel est donc indispensable à la validation de ces approches, mais c'est une tâche complexe en soi et qui peut mener à des blocages, comme rencontré dans le chapitre 3.

Les travaux présentés ici, comme la majorité de ceux présents dans la littérature sur l'apprentissage machine pour les communications sont des simulations ou, au mieux, des implémentations sur banc d'essai. Pour pouvoir envisager de les mettre en production, et donc les déployer de manière robuste et efficace à grande échelle, il reste trois défis à relever : une implantation efficace des réseaux, des protocoles d'apprentissage continus, et une garantie de fiabilité.

- Les réseaux présentés dans cette thèse ont été implémentés et entrainés sur un serveur avec processeur et cartes graphiques. Ces unités de calcul sont puissantes, capable de gérer des réseaux massifs par de larges capacités de transfert, des opérations à virgule flottante, et beaucoup d'espace mémoire. Mais les objets grand public, et surtout de l'IoT, sont très contraints en espace, énergie, et en prix. Ils ne possèdent généralement pas de carte graphique dédiée, ni même de processeur puissant. Pour remédier à cela, des unités de calcul dédiées aux tenseurs sont développées pour une inférence plus efficace. Mais pour faire rentrer un réseau sur de telles unités de calcul, il faut porter une attention spécifique aux systèmes d'apprentissage pour réduire la taille des réseaux et pour simplifier les opérations de calcul, notamment en utilisant de la quantification pour passer de paramètres en nombre flottants sur 32 ou même 64 bits, à des entiers de quelques bits. Cette étape de quantification peut impacter la performance finale, mais des mesures existent pour limiter cet effet [53].

- Par leur nature, les systèmes d'apprentissage profond sont très gourmands en données pour leur entrainement. Même si ces données peuvent être générées à grande vitesse quand on utilise des symboles ou des paquets reçus, l'obtention des données et l'entrainement peut quand même prendre plusieurs minutes, ou même heures. Et au niveau de la couche physique, les conditions changent rapidement, surtout en cas de mobilité. Donc on peut s'attendre à devoir ré-entrainer ou affiner fréquemment des systèmes en production. Il faut donc créer des protocoles spécifiques pour gérer le transfert de données d'entrainement, de gradients et/ou de paramètres. Le tout, sans gêner le flux de données principal. Il y a aussi besoin de techniques pour réduire le nombre d'exemples nécessaires à une mise à jour d'un réseau.

- Il est difficile d'expliquer le comportement d'un réseau de neurones. La dimension extrêmement grande des paramètres du réseau fait que la fonction qui en résulte est très complexe et donc qu'il est difficile de la comprendre et d'évaluer l'importance et l'impact de chacune des entrées. Avec des algorithmes classiques, il peut être relativement simple de caractériser les effets de bords, leur impact et où ils se produisent. Mais un réseau de neurones a des "bords" multi-dimensionnels, et cela mène à des zones problématiques très dispersées dans l'espace d'entrée et peut causer des bugs difficiles à trouver et résoudre. Ce problème de fiabilité peut être exploité par des attaques provenant de systèmes apprenant et causer des problèmes de sécurité [105], [106]. La nature même

des traitement effectués en couche physique, écoutant en continu n'importe quelle transmission sans fil, signifie que les systèmes à ce niveau sont directement exposés au agents malicieux. Une forme de protection pourrait venir de l'adversaire traditionnel des communications : le bruit et les effets du canal. En effet, de nombreuses techniques d'attaque se basent sur des signaux malicieux construits très précisément. La distorsion causée par le canal entre l'attaquant et la cible pourrait rendre ces signaux inefficaces. Mais certaines attaques ont été démontrées, même en présence de signaux bruités et hors de simulations [151].

— 1 —

Introduction

1.1. Current and future communication challenges

THE future communication challenges to be met by the next protocols like 6G and beyond look a lot like the challenges that led to the current wireless communication standards, such as the 4G and 5G cellular networks or the Sigfox and LoRaWAN low power networks that were designed more than a decade ago. There are similar concerns towards the increase in transmission speed, handling of ever more connected devices, decrease in latency while improving reliability, and attention towards energy efficiency for a lesser carbon impact of the telecommunications world. But while none of these aspects has been considered "solved", with continuously increased expectations towards each of them, they all have seen evolution, testament to the work done by researchers and industry alike over the years.

Transmission speed

The span between 3G and 4G cellular networks has seen a few hundred fold increase in mobile users download rates from less than 400 kbit/s to more than 100 Mbit/s between the two generations and 5G promises to add another ten-fold increase to that. But the latest increases, and the ones that will come in the future do not come from an increase in transmission efficiency. The 4G standard is already nearing the Shannon limit for point-to-point communications, so no significant gains can be expected on that front. Latest developments in 4G come from the introduction of massive multiple-input multiple-output (MIMO) transmissions allowing for an increase in received signal strength. And 5G brings an increase in frequency band width, opening channels above 6 GHz. Future standards are expected to continue in this trend, improving MIMO techniques by increasing the number of antennas, more accurate beamforming, localisation to more precisely point beams at users,... and by going to even higher frequency bands, towards the terahertz.

Connected devices

10 years ago, deployment of internet of things (IoT) was expected to explode, Cisco predicting 50 billion connected devices by 2020 [2], and even a 2012 prediction of one trillion connected devices in 2015 by IBM [3], yet, as of 2020, less than 22 billion devices have been connected. Despite those over-enthusiastic expectations, the amount of devices to wirelessly connect is growing rapidly, and those devices require bandwidth to communicate. Solutions to this problem can be found in the same toolbox as for increasing transmission speed, reducing interference with spacial multiplexing from beamforming, and exploiting the large spectrum real estate found in higher frequencies. But these solutions are costly in term of processing power for MIMO systems, and transmission distances shrink with raised frequency, so they do not fit with the ecosystem of IoT devices characterised by low price and low maintenance for an ubiquitous deployment. Other approaches are needed to allow a large number of transmissions to occur in the crowded spectrum of the lower frequencies. Protocols such as Sigfox have been proposed that rely on robustness towards packet collisions using a large amount of transmitted redundancy, but the future challenge is to introduce truly non orthogonal multiple access (NOMA) allowing for several devices to simultaneously and successfully transmit signals.

Quality of service

The development of connected machinery, at first inside of factories, but increasingly outside, with connected, remote controlled, and autonomous vehicles presents a high risk to the public. This category of devices requires a very high level of quality of service. Reliability is key in ensuring the safety of human lives. Communication with highly mobile vehicles, especially in a highly dynamic urban environment, also require a very low transmission latency to ensure that only up-to-date information is received. This gave the rise to the category of ultra reliable low latency communications (URLLC). Signalling overhead plays a big role in reducing latency, the classical back and forth handshake relying on at least three separate transmissions. So specific grant-free protocols need to be used. Small packets are also required: a transmission latency of 1 ms can never be met if the packet itself is longer than that. But this clashes with the common coding scheme, relying on long blocks of data to provide reliability. So specific codes need to be designed.

Energy management

Efficient usage of energy is a growing concern for two types of actors in the radio communication chain, with different underlying causes. Big equipments such as base stations are connected to the grid and, as such, can have important power requirements without much impact outside of the exact specification of the power connections. But they require a lot of energy, and in most of the world, this comes from fossil-burning power-plants. So any efficiency increase could have an important impact on the overall carbon footprint of the communications industry. On the other side, many connected devices are not connected and get their energy from batteries. For user facing devices such as phones or wearables, battery life directly affects user experience, but, for a phone lasting 24 hours in one charge, a 10 % increase may not lead to noticeable changes for the user, especially when said user lives in proximity to power outlets. On the other hand, IoT sensors can be deployed in hard

to reach locations and for very long times, so the same 10 % increase in the battery life of a sensor deployed for 10 years can lead to significant cost savings for their operators. For both base-stations and devices situations, solutions can be found in more efficient signal processing algorithms, as well as a better management of the transmission power. To increase low power devices battery life, energy harvesting systems are being developed, as well as simultaneous energy and information transmission protocols.

In all these aspects, with multiple users, multiple antennas, higher frequency bands, ... designers are venturing away from well established and understood models and theories, with more and more complex and dynamic systems. For this, and for the other aspects where speed and efficiency are key, new or more efficient algorithms are required, and this is why machine learning, and more especially deep learning, has become an important field of study for communications.

1.2. Machine learning for communications

This is where machine learning comes into play, with the widespread adoption of its deep learning variant. This approach of learning solutions to complex problems from data has spread throughout the scientific community at large since the beginning of the previous decade, starting in 2012 in the image processing community. It promises great strides in performance compared to classical algorithmic approaches, to the point where it is seen by some as a one size fits all solution, being applied too any and all problems without regards to the pertinence of such application. While it can indeed be very powerful, it can not be applied everywhere effectively or practically. Fig. 1.1 presents a basic, rule of thumb, flowchart to find if a machine learning approach such as deep learning can be useful for a particular problem. The main selling point of machine learning is its ability to learn from data. The corollary to this is the need to have access to a large quantity of high quality data points to train the system. If it is not available, it is expected to perform poorly. Fortunately, the main activity of the digital communications field is to manage and transmit large quantities of information from machine to machine, in an automated manner. This leads to a high amount of available data. Machine learning algorithms have the capability to directly learn from data, without requiring an explicit model of the physical processes underlying said data, contrary to traditional approaches that require a deep comprehension of the model before designing an algorithm. It means that they are particularly interesting in cases where no good model exist, such as with NOMA conditions, or considering highly non linear hardware effects. Finally, even if a good model exists, if no fast and efficient traditional algorithm has been designed, machine learning, and especially deep learning systems can offer quicker, or more easily parallelised implementations.

Machine learning and neural networks have been proposed a long time ago, with a survey dating back to 2000 presenting multiple uses of neural networks for channel modelisation, equalisation, decoding, filtering,... and predicting that "Neural networks will certainly be one of the key technologies for the communication domain in the 21st century" [4]. Nevertheless, and in a similar fashion as in other fields of study, the use of these techniques remained marginal for more than a decade. And it was after significant development in processing speeds, graphics processing unit (GPU) technology, and the remarkable demonstrations of performance from the image processing field, that the use of machine learning, and more specifically deep learning, for communications became of interest to the community beginning

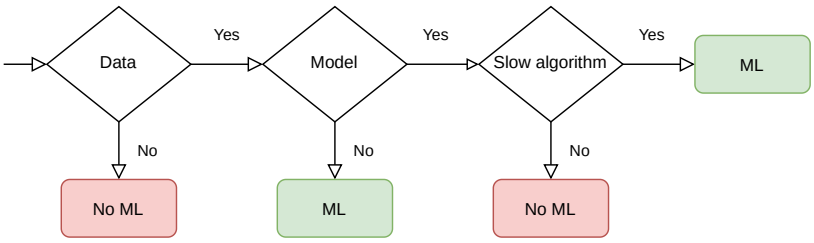

Figure 1.1.: Flowchart of ML algorithm pertinence

around 2016 with influential works on modulation classification [5], channel code decoding [6], or viewing the end-to-end transmission chain as an autoencoder [7]. It also corresponds to a point where research had started to move away from the well understood scenarios to tackle the challenges presented above, and the community needed to use new tools to handle the new, model-less conditions. Since then, machine and deep learning has permeated most signal processing tasks in the physical layer for communications, with geometric shaping, resource allocation, interference management, hardware modelisation,... [8], [9].

1.3. Outline and contributions

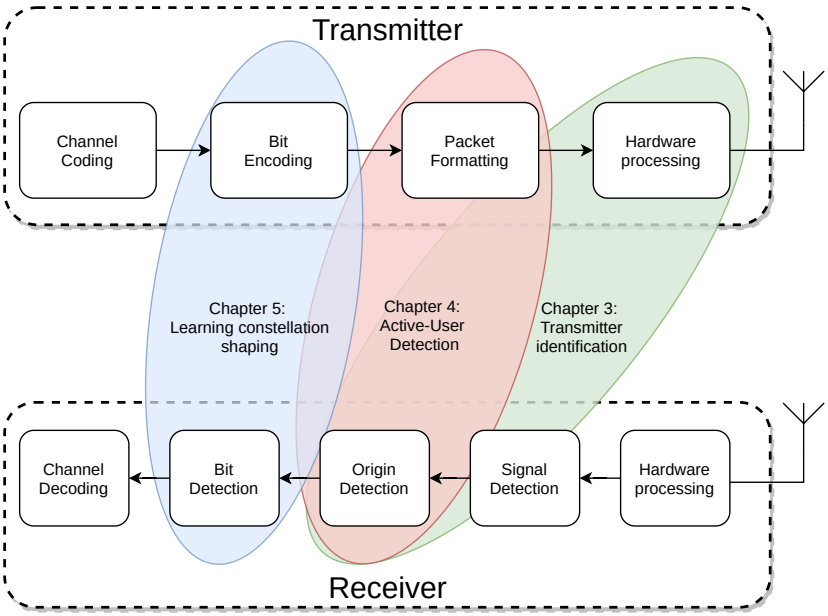

Figure 1.2.: Outline of the thesis

The work presented in this thesis aims at improving the performance of detection algorithms used in the signal processing chain of a wireless receiver by employing deep learning techniques in scenarios where multiple devices are expected to transmit. To this end, Chapter 2 presents a background in deep learning, with its place in the machine learning ecosystem. It is then followed by the construction of a neural network and its training as used in the

following chapters. Elements regarding the gathering of datasets to train and evaluate neural networks are also discussed, as well as potential issues that can arise in the process with matters of labelling, or the introduction of biases. Finally, the tools for the software defined radio experiments for data gathering throughout this thesis are presented.

Fig.1.2 presents the tasks tackled in the various chapters of this thesis, in the context of a transmission-reception chain. From the point of view of the receiver, the first processing is done in hardware, with filtering to isolate the frequency channel of interest from noise and interference, amplification, conversion to baseband,... before being converted from an analogue waveform to digital samples. This part of processing is not trivial to execute, with many scientific and industrial implementation challenges, but since it is analogue, it is not compatible with the digital machine learning algorithms. After digitisation, the first step is to detect if, and when, a signal has been received. As long as the transmission protocol is known in advance, this task is relatively simple to implement with well performing classical algorithms available, such as correlation with Zadoff-Chu sequence preambles. In this case, the use of a machine learning algorithm is not relevant. When aspects of the protocol are not known in advance, spectrum sensing techniques are required such as modulation classification, and in this case, deep learning has been shown to be relevant, but it is not in the scope of this thesis. The first detection task where the use of deep learning is relevant is then origin detection as the transmitter of a received packet needs to be identified for the contained data to be useful.

Chapter 3 first tackles this task in a scenario where one user transmits at a time, in a pool of multiple known devices. In this chapter, the imperfections of the hardware processing at the transmitter are leveraged to identify the transmitting device in a form of fingerprinting. A convolutional neural network is trained on experimental data from devices of the same models, and the impact of the type of transmitted data on identification accuracy is studied, as well as dataset construction methods to increase the robustness of the classification towards changes in the channel characteristics.

Chapter 4 extends the scenario of the previous chapter to one where multiple devices can be active at the same time. The use of fingerprinting with subtle hardware imperfections is not possible anymore, and a specific code is introduced in the transmitted packets of each transmitter. This chapter is concerned by designing a neural network decoder able to identify the set of active users in a simulated environment and comparing its performance to more traditional algorithms, but not by the design of the set of identification codes.

After tackling origin detection, chapter 5 concerns the next step in the reception chain, bit detection. In this chapter, a scenario with two simultaneously transmitting users is considered, with the objective of detecting the transmitted bits of both devices at the receiver. It is a similar problem as the previous chapter, where the activity of a user could be seen as the transmission of one bit of information, but here, the focus is on the design of performant codebooks tailored to this specific scenario and allowing for an accurate detection at the receiver. The chapter also focuses on the inherent performance tradeoffs that arise when considering more than one user at a time.

Finally, Chapter 6 concludes this thesis with a summary of this work, as well as look towards future prospects on the problems touched here.

1.4. Publications

Journal papers

[10] C. Morin, L. Cardoso, J. Hoydis, *et al.*, "Channel resilience inducing dataset construction for physical layer device fingerprinting", *IEEE Transactions on Cognitive Communications and Networking*, 2021, to be submitted.

Conference articles

[11] C. Morin, L. Cardoso, J. Hoydis, *et al.*, "Transmitter Classification With Supervised Deep Learning", in *Crowncom 2019 - 14th EAI International conference on Cognitive Radio Oriented Wireless Networks*, Poznan, Poland, Jun. 2019, pp. 1–14
[12] C. Morin, D. Duchemin, J.-M. S. Gorce, *et al.*, "Active user blind detection through deep learning", in *Crowncom 2020 - 15th EAI International Conference on Cognitive Radio Oriented Wireless Networks*, Rome, Italy, Nov. 2020, pp. 1–14
[13] C. Morin, L. Cardoso, J. Hoydis, *et al.*, "Neural Constellation shaping for the two-user uplink NOMA gaussian channel", in *IEEE Global Communications Conference*, Madrid, Spain, Dec. 2021, to be submitted.

Patents

Presentations

GdR ISIS 2018: Transmitter identification with deep learning.
French GNURadio days 2018: Learning to fingerprint : physical layer identification.
ISP-IoT Winter School 2018: Learning to fingerprint : physical layer identification (An experimental approach). (Poster)
GNU Radio Conference 2019: Your radios are unique. Don't believe us? Come gather more data. (Poster)

Awards

6th IRACON training school on Machine and Deep learning techniques for (beyond) 5G Wireless communication systems. 1st place in Machine Learning Challenge.

— 2 —

Background

THIS chapter presents background knowledge on the concepts used throughout this thesis, starting with a look at machine learning, and more specifically its deep learning variant, followed by considerations of data gathering, to cover the learning aspects of the work presented here. The second part involves the presentation of the signal processing tools used for experimental works, focused on the Future Internet of Things / Cognitive Radio Testbed [1] (FIT/CorteXlab) experimentation room, and the GNU Radio software defined radio (SDR) framework.

2.1. Machine learning and Deep learning

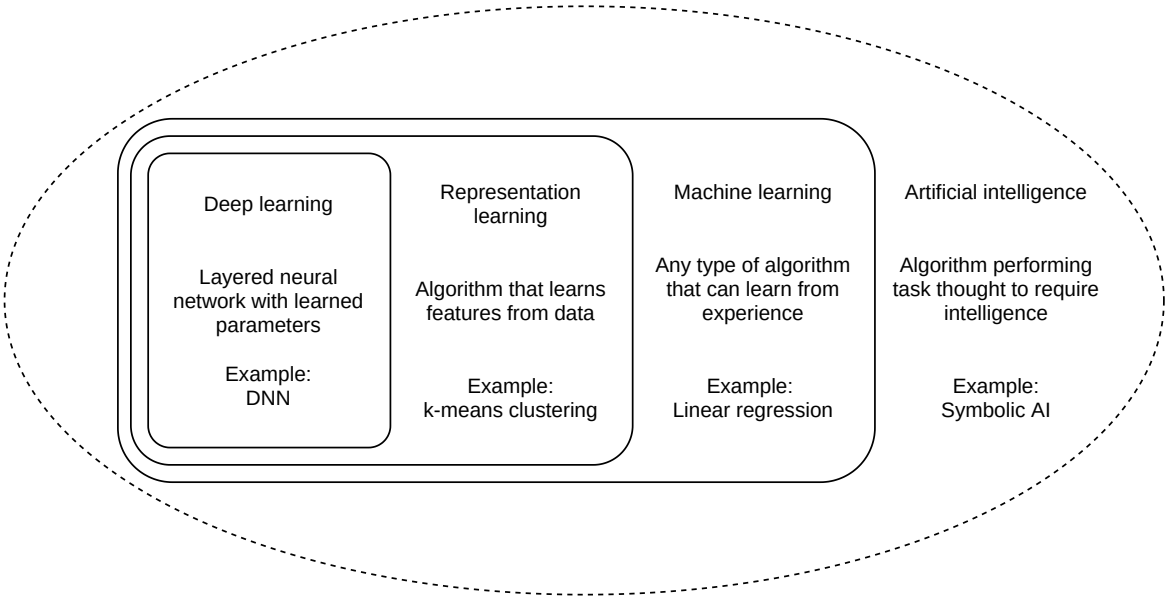

Figure 2.1.: Venn diagram showing the relations between the various concepts of artificial intelligence relative to deep learning.

This part briefly presents what are machine learning (ML) and deep learning (DL) and the aspects that will be of interest in this thesis. A more detailed description can be found in [14]. Many related but distinct terms exist around this topic with their own acronyms, from AI, to ML, to DL. Fig. 2.1 provides a short description of the relation between some of that will be explored in this chapter.

2.1.1. Machine learning

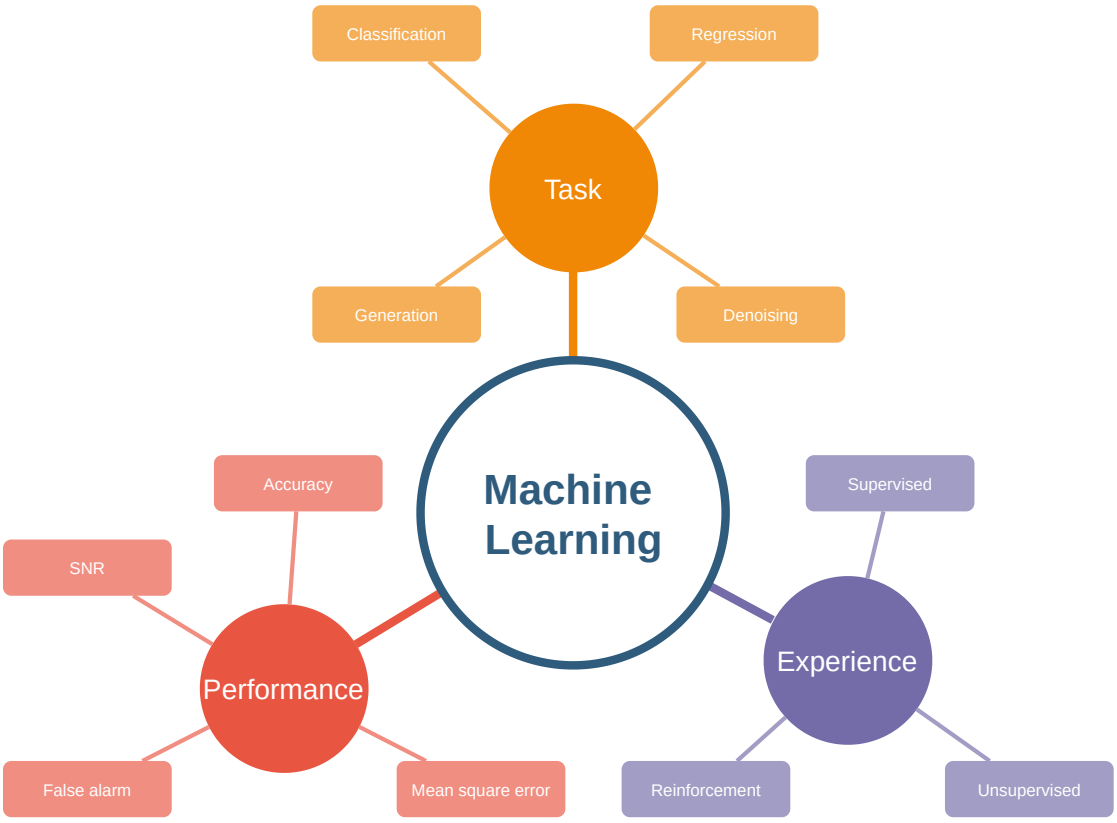

Figure 2.2.: Map of the three components of a machine learning algorithm, with some examples for each

Machine learning is a specific, albeit widespread, kind of artificial intelligence (AI), that describes a computer algorithm able to use data to adapt and improve its performance, in a learning process. Mitchell [15] proposes this definition: "A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E. The definition of experience E, task T, and performance P is very broad and can encompass a wide variety of scenarios The following presents some common examples, starting with tasks, performance, and finally experience. Tasks that will be explored or mentioned in this thesis and often seen in communication scenarios include:

- **Classification**: This is the task tackled by most of the proposed algorithms in this thesis. The algorithm is given a list of k classes, and it has to decide, for a certain input y,

which of the k classes it belongs to, leading to a set of active classes $\underline{\mathcal{A}}$ for y. Several sub categories exist, such as the binary classification, where $k = 1$ and the system has to decide if the class is, or isn't active. Multi-class classification describes tasks with $k > 1$ and only one is active at a time, $|\underline{\mathcal{A}}| = 1$. The Transmitter identification problem explored in Chapter 3 is an example of this type, such as the more famous handwritten digit recognition task of the MNIST dataset [16]. The generalisation of the multi-class problem, is the multi-label classification, with no restriction on the cardinality of the active class set $\underline{\mathcal{A}}$. Chapter 4 shows and example of this task, as well as the receivers of Chapter 5. The multi-label and multi-class problems can also be reverted to a set of binary classifications by either slitting it into a set of k *one-against-all* tasks, or a set of $\frac{k(k-1)}{2}$ *one-vs.-one* tasks, but step may need to be taken to preserve the $|\underline{\mathcal{A}}| = 1$ property of the multi-class problem.

- **Anomaly detection**: In this type of task, that could be distilled into a binary classification problem, the system has to decide if y is anomalous or not. The authorisation type works presented in Chapter 3 that focus on deciding if the input if from a known transmitter of not is a good example of it. This task can be distinguished from its parent classification by the fact that, by their nature, anomalies are rare and unpredictable, so it is often not feasible to include all anomalous states in the experience E of the machine learning algorithm training. Surveys of this type of task can be seen in [17], [18].

- **Generation**: The previous categories take a complex input and refine it down to a few underlying features. This type of task can be seen as an opposite, with the output being more complex than the input. A good example of this task can be seen in the generator part of generative adversarial network (GAN) [19] where an algorithm is tasked with synthesising signals to fool a detector in distinguishing them from real world signals. The encoding element in Chapter 5, creating constellations with only one noise level value as input, is also an example of this type of task.

- **Denoising**: In between the Classification and the Generation type, the Denoiser task involves the same dimension in input and output. Here, the goal for the machine learning algorithm is to clean the input corrupted example from noise. Many communication tasks are closely linked to noise reduction, as are all the algorithms used in a receiver presented in this thesis.

Some other commonly encountered task include:

- **Regression**: In this type of task, the algorithm has to predict a value, typically a scalar but not limited to it, given y. In contrast to the classification, the output is not limited to binary values, or probabilities between 0 and 1, but can span $\mathbb{R}$, or even $\mathbb{C}$. An example of regression task could be the prediction of the area burned by a forest fire given meteorological data [20], [21].

- **Translation**: This type involves the translation of a sequence of symbols from one language to another, without changing its semantics. It is typically used in text translation, such as in the Google Translate tool [22], but could also be considered for translation between types of data, such as text to speech, or optical character recognition.

The performance measure P is the quantitative measurement of the performance of the algorithm on the selected task T, sometimes also referred as the *metric*. There can be more than

one evaluated performance measure for a given task, to more completely capture the multiple dimensions of performance. For instance, a classification task often involves measuring the accuracy of the algorithm, the frequency at which the system predicts the correct active set. But in this same task, one can also be interested in error rates, of even more precisely, false alarm rates, where a class is said active whereas it is in reality not, or misdetection rates, where the class is wrongly set as inactive. A denoising task could look at the signal-to-noise ratio (SNR) at the output, or a measure of distortion of the original signal, while a regression problem could use a measure of distance between the predicted value, and the correct one. Note that many of these performance measures rely on the existence of a "correct" or *ground truth* desired output for computation. This drawback is also present when describing the experience, and will be discussed more deeply there. For reasons that will be discussed with the overfitting problem, the performance measure evaluates performance of the algorithm on data that is different from the data used for training.

The experience E available for training allows to distinguish between three main categories, relative to the information available in the data. When the experience only include the input y, the learning is said to be *unsupervised*. The algorithms will then try in infer underlying features in the set of experiences, such as the probability distribution that may have generated the experiences, or separate the experiences in clusters. When a *label* is associated with the input for each data point, it is called *supervised* learning. This label contains the ground truth value for this data point. So, if the learning algorithm observes the input y seemingly random from its point of view, having an associated ground truth value x, the unsupervised learning restricts the system to learn the distribution $p(y)$, or features of this distribution, while the supervised learning, with the help of the label providing information on the ground truth, allows to learn the prediction of x from y, or an estimation of $p(x|y)$. There is no hard frontier between the two categories. The approach in-between is called semi-supervised learning, and involves labels on only some data points, while the rest is unlabelled.

The two types described above rely on a fixed set of data points, or at least on independence between the operation of the learning algorithm and the experience. In *reinforcement* learning, the algorithm chooses actions that interact with an environment from which the experience is taken. This means that there is a feedback between the system's output, and the experience for the next training iteration. More information on the techniques adapted to this situation can be found in [23].

Many types of machine learning algorithms exist, each more adapted to a specific set of task and experience type. For unsupervised classification, clustering techniques can be used, such as k-means clustering [24], that attempts to divide the space of y into k clusters. It is often confused with the k-nearest neighbours technique that is designed for a supervised approach. The principal components analysis (PCA) algorithm [25] is a commonly used for learning a lower dimension representation of the input data, and is often used as a first step to reduce the complexity of inputs to another machine learning algorithm.

On the supervised learning side, a simple algorithm that serves as basis for many more advanced approaches is the linear regression that, as its name implies, is tailored to solve regression tasks. It can be adapted to classification and becomes logistic regression. These two will be detailed in the next section as they are the basis for neural networks. Support vector machine (SVM) [26] are another derivative of the linear regression for binary classification. It use a base linear regression to slice the input space into two zones, one for each class, but can also replace its internal dot-product by non-linear functions to handle non-linear classification. Another important family of classification algorithms for supervised learning is the decision

tree, that defines a tree of decision nodes to slice the input space into decision regions.

Since machine learning is part of the field of computer science, the experience has to be presented as sets of numbers encoding for real world elements, to be usable by the algorithms executed on binary logic processing units. Images have to be captured and converted into a matrix of pixels with light or colour values, speech digitised into sequences of amplitudes. But the real world is not discrete, and all of its complexity cannot be captured and used at once by an algorithm and, even more so, shouldn't since most of the information available wouldn't be relevant to the task at hand. So the researchers have to decide on a *representation*, a set of *features* that accurately and efficiently describes the necessary information to present to as experience to the learning algorithm in order to solve the desired task. But discovering an appropriate set of features, *feature engineering*, can be extremely complex, as well as designing good algorithms to extract them. For instance, in recognising the presence of a bicycle in a picture, the presence of pedals and wheels can be very useful features, but these are difficult to describe in a pixels, especially since they can be seen from many different angles, in various light conditions, materials, colours,... One solution is to have the learning algorithm tackle a secondary task, of *representation learning*, where it has to discover its own useful features, from a higher dimensional and more generic representation, such as an entire image.

2.1.2. Deep learning

DL is a specific class of ML algorithm, so it inherits the task, performance measure, and experience elements of it. This thesis solely implements DL algorithms, so this chapter focuses on it. But many takeaway points, especially regarding the experience, can be applied to all ML approaches. DL corresponds to the training of a neural network to perform the selected task. This neural network, as its name implies, is constituted of a network of linked simple computation *units*, also called neurons, arranged in a way to allow the computation of complex functions. It has been shown that a sufficiently large neural network of the proper architecture can approximate any function [27]. The layered structured can be seen as performing representation learning, each layer providing a new, hopefully refined, set of features to the next, until the last layer that does the desired task on a simpler representation. This characteristic renders it able to operate on generic representations, providing enough training, and a sufficient number of layers to refine it.

The neuron

As mentioned above, neurons are the base constituents, building blocks of the neural networks and operate a simple mathematical function from input, based on internal parameters, using a linear unit as a basis. This linear unit operates on a vector of inputs of length n $\mathbf{x} \in \mathbb{R}^n$, using an internal set of parameters θ comprising a weight vector $\mathbf{w} \in \mathbb{R}^n$ and a bias scalar $b \in \mathbb{R}$, and outputs the value $y = l(\mathbf{x}, \theta) \in \mathbb{R}$ as an affine function:

$$l(\mathbf{x}; \theta) = \mathbf{w}^\mathsf{T}\mathbf{x} + b \tag{2.1}$$

The neuron unit expands the use of the linear unit by adding an *activation function* $\phi(.)$ whose primary purpose is to introduce non linearities to increase the expressiveness of the resulting network.

$$f(\mathbf{x}; \theta) = \phi(l(\mathbf{x}; \theta)) = \phi(\mathbf{w}^\mathsf{T}\mathbf{x} + b) \tag{2.2}$$

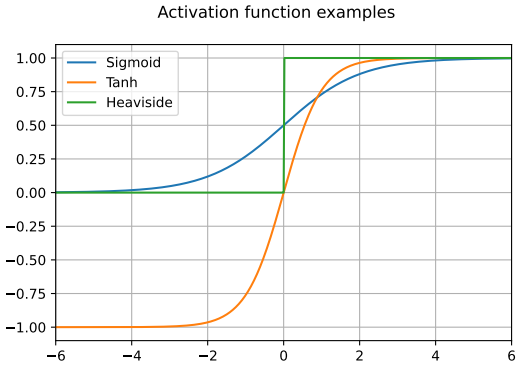

Figure 2.3.: Bounded activation functions

Many activation functions exist, in two broad categories: the ones that introduce bounds, $\phi : \mathbb{R} \to (a, b)$ with a, b not infinite, generally -1, 0, or 1, and the ones that don't, $\phi : \mathbb{R} \to (a, \infty)$. or $\phi : \mathbb{R} \to (-\infty, \infty)$. In the first category and illustrated in Fig.2.3, one can find the sigmoid, or logistic, function, generally used for binary classification:

$$\text{sigmoid}(x) = \frac{1}{1 + e^{-x}}, \tag{2.3}$$

the *tanh* function, and the Heaviside or step function, defined as:

$$H(x) = \begin{cases} 0 & \text{if } x < 0 \\ 1 & \text{if } x > 0 \end{cases} \tag{2.4}$$

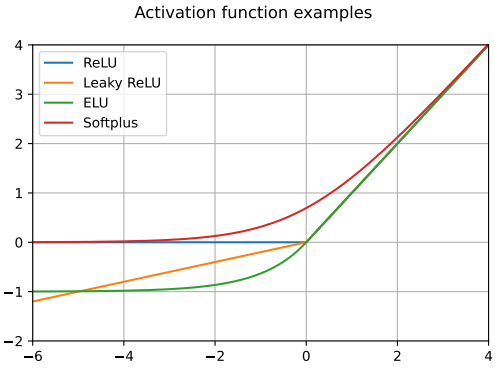

Figure 2.4.: Unbounded activation functions

In the unbounded category, there is the Softplus [28], defined as $f(x) = ln(1 + e^x)$, and the rectified linear unit (ReLU) [29], defined as:

$$\text{ReLU}(x) = \begin{cases} 0 & \text{if } x < 0 \\ x & \text{if } x \geq 0 \end{cases} \tag{2.5}$$

With the standard gradient based training algorithms, described in 2.1.2, the continuity, derivability, and the derivative itself of the activation can have an impact on the convergence of the training. On this regard, ReLU, while being the most commonly used activation function in deep learning applications, shows several drawbacks: its gradient is not continuous and is equal to zero when the input is negative, leading to a possible permanent deactivation of a unit. For this reason, several variants of the ReLU function have been proposed, such as the Leaky Relu [30], where the gradient is never zero:

$$Leaky\,\mathrm{ReLU}(x) = \begin{cases} 0.01 & \text{if } x < 0 \\ x & \text{if } x \geq 0 \end{cases} \tag{2.6}$$

or also exponential linear unit (ELU) [31], that additionally provides a smooth gradient:

$$\mathrm{ELU}(x) = \begin{cases} e^x - 1 & \text{if } x < 0 \\ x & \text{if } x \geq 0 \end{cases} \tag{2.7}$$

Many other variants and functions exist, some with additional parameters, that can sometimes even be trained with the rest of the network.

The network

To form a network in its base form, called multi-layer perceptron (MLP), feed-forward neural network, or dense neural network (DNN), several units are grouped in a layer, and several layers are linked, each one using as input the output of the previous one, and without loops. The *depth* of the network relates to the number of layers, leading to deep or shallow networks, and the *width* relates to the number of units in a layer. This width is more complex to evaluate since layers in a network do not always contain the same number of neurons. The first layer, the one operating on the input fed to the network is called the *input layer*, the last layer is the *output layer*, and the ones in-between, that are not "seen" from the outside are called the *hidden layers*. For ease of implementation, units in a layer are usually stacked in one matrix operation, combining the weight vectors $\mathbf{w}$ into a weight matrix $\mathbf{W}$ and the bias scalars b into a bias vector $\mathbf{b}$. So the computation of the i^th layer goes as follows:

$$\mathbf{x}_i = f(\mathbf{x}_{i-1}; \theta_i) = \phi_i(\mathbf{W}_i^\mathsf{T} \mathbf{x}_{i-1} + \mathbf{b}_i) \tag{2.8}$$

Several additional layer and architecture types have been proposed over the years, such as the convolutional layer, constitutive of a convolutional neural network (CNN). It is able to operate on inputs in the form of matrices, and not only vectors. The main difference of this layer is that it uses a discrete cross-correlation operation on its input instead of a full matrix multiplication like the dense layers. Here, the neuron weights describe a filter, called a *kernel*, that is generally small with regards to the size of the input. When operating in two dimensions such as on an image, with an $\mathbf{X}$ input matrix and a weights kernel $\mathbf{W}$ of height G and width H,the neuron or *filter* outputs a matrix $\mathbf{Y}$ whose m^{th} and n^{th} elements are computed as follows:

$$\mathbf{Y}_{m,n} = f(\mathbf{X}, m, n; \theta) = \phi(b + \sum_{g}^{G} \sum_{h}^{H} \mathbf{X}_{m+g,n+h} \mathbf{W}_{g,h}) \tag{2.9}$$

This operation is repeated by increasing m and n to cover the entire input matrix, and

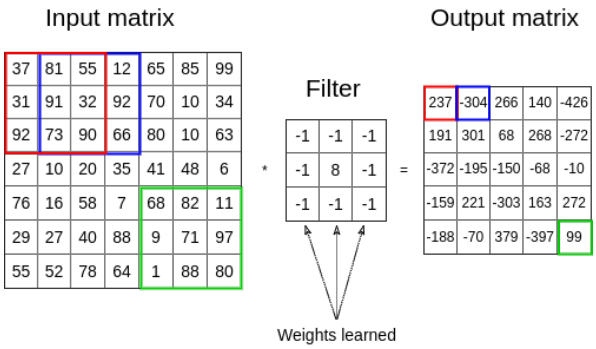

Figure 2.5.: Diagram of a convolutional layer

multiple filters form a layer. The output of this layer is then a tensor with its third dimension being the number of filters. This type of operation is not limited to 2D inputs, but can operate similarly on an arbitrary number of dimensions (usually from 1D to 3D) by adapting the kernel shape, and the number of summations. To truly be a convolution operation, and not a cross-correlation, the filter should do the following:

$$\mathbf{Y}_{m,n} = f(\mathbf{X}, m, n; \theta) = \phi(b + \sum_{g}^{G}\sum_{h}^{H} \mathbf{X}_{m-g,n-h}\mathbf{W}_{g,h}) \tag{2.10}$$

but to optimise implementation, most deep learning frameworks do not include this kernel flip and still call it a convolution. Since the kernel weights will be trained, the system is able to learn an inverted kernel by itself. An additional parameter to this operation is the stride that allows to cover the input matrix with and output matrix of a smaller size. If s is the stride parameter, the layer operation becomes:

$$\mathbf{Y}_{m,n} = f(\mathbf{X}, m, n; \theta) = \phi(b + \sum_{g}^{G}\sum_{h}^{H} \mathbf{X}_{s\times m+g,s\times n+h}\mathbf{W}_{g,h}) \tag{2.11}$$

This stride parameter can be different for all the input dimensions. Similar to the stride, the dilation parameter [32] allows for a filter with disjointed elements, increasing its reach without increasing the number of parameter in the kernel. With a dilation parameter d, that could also have different dimensions, the layer operation becomes:

$$\mathbf{Y}_{m,n} = f(\mathbf{X}, m, n; \theta) = \phi(b + \sum_{g}^{G}\sum_{h}^{H} \mathbf{X}_{m+d\times g,d+s\times h}\mathbf{W}_{g,h}) \tag{2.12}$$

The output matrix of these operations is smaller than the input matrix. To allow for simpler implementation, and easier stacking of multiple layers in complex architectures, a *padding* can be used, adding zeroes around the input matrix until the output has the desired shape. Convolution layers are generally associated to *pooling* layers. These are non learned version where kernel either takes the maximum value, the MaxPool, or does a non-weighted average, the AvgPool. They are used to downsample the data to reduce the computation of the following layers.

Recurrent layers forming a recurrent neural network (RNN) are another commonly used form of layer. As their name implies, their function rest on recursive operation and are designed to work on sequences, or 1D inputs. Here, the same unit, i.e. with the same set of weights, operate sequentially on the input, one sample at a time. It keeps a hidden variable h from one recursion to the next and updates it if necessary, depending on the current input, as a sort of memory of previous iterations. With $\mathbf{x}$ the input vector, and operating on the t^th element of $\mathbf{x}$, this leads to:

$$h^{(t)} = f(h^{(t-1)}, \mathbf{x}^{(t)}; \theta) \tag{2.13}$$

And the output of the unit $\mathbf{y}$ depends on this hidden state:

$$\mathbf{y}^{(t)} = g(h^{(t)}, \mathbf{x}^{(t)}; \theta) \tag{2.14}$$

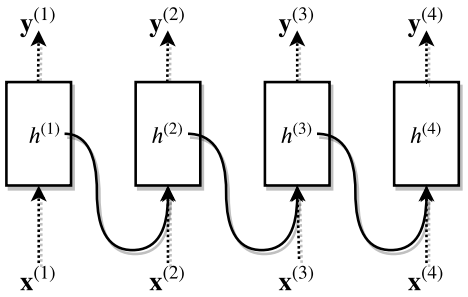

Figure 2.6.: Diagram of an unfolded recurrent layer

Several types of functions to update the hidden states and output a value exist, such as in the long short-term memory (LSTM) [33] or the gated recurrent unit (GRU) [34]. As it is not a type of layer used in this thesis, these will not be detailed extensively. Since the input sequence is finite, the recursive operation can be unfolded with $h^{(t)}$ becoming a function of all previous input samples, as illustrated in Fig.2.6. Bi-directional variants also exist that use the sequence in both orders at the same time:

$$h^{(t)} = f(h^{(t-1)}, h^{(t+1)}, \mathbf{x}^{(t)}; \theta) \tag{2.15}$$

All these layer types can be used together in simple or complex architectures, and a sequential order of layers is not needed. One could imagine a complex system, with a first image being input to a few convolutional layers, the output of which could be flattened into a vector to be treated by a dense layer, with a second input sequence being added to it before input to a recurrent layer. A copy of the convolutional layers output could even skip the dense layer to add to the output. For context, AlexNet [35] that won the ImageNet image recognition contest in 2012 and spurred the widespread use of CNN and GPU is a multi-class convolutional network operating on colour images 224 wide and tall, with five convolution layers interleaved with max pooling, followed by three dense layers, with 1000 output classes and ReLU activation functions. Another popular architecture is ResNet [36], that is based on residual modules of two successive convolutional layers, with the input of the $(i+1)^th$ module connected to the output of both the $(i)^th$ and $(i-1)^th$ modules. This allowed very deep networks, with hundred, or even thousands of layers, and is the basis of receiver network in Chapter 5. The choice of an architecture for a specific task will be

discussed in Sec. 2.3.1

Training

A neural network is a complex succession of numerous matrix multiplications and has the potential to approximate any function, but what makes is useful in practice, and transforms it into deep learning, is the usage of a learning algorithm to train the weights of the neurons to perform well on the desired task. Since the neural networks used in this thesis, as well as in the majority of literature works are in supervised learning scenarios, the training algorithm presented here will allow the use of labelled data. In this section, the network will be considered as an overall function operating on the input vector $\mathbf{x}$ according to the parameter vector $\boldsymbol{\theta}$ combining the weights and biases of all the layers as $f(\mathbf{x};\boldsymbol{\theta})$

Before the algorithm can start, a performance measure P, called *loss function* in the literature, needs to be selected. In this supervised learning setup, the function compares the output of the network with the ground truth, label value $\mathbf{y}$ as $\mathcal{L}(f(\mathbf{x};\boldsymbol{\theta}),\mathbf{y})$. The choice of this loss function is important to the performance of the trained network, this choice will be discussed in Sec. 2.3.2.

The weights could technically be set by hand, but it quickly becomes intractable and needs to be automatised as soon as a few neurons are part of the network. One way to learn the weights is to use a genetic algorithm, creating N copies of the network with random weight initialisation, evaluating them all and keeping the M best performing in the first generation. The next generation of N networks can then be created by slightly modifying the weights of the M networks of the previous generation, and repeating the process until the performance increases.

But the main training algorithm used to learn the weights of a network is based on gradient descent. The goal of this approach is to compute the gradient of the loss function with regards to every component i of the parameter vector as $\frac{\partial}{\partial\boldsymbol{\theta}_i}(f(\mathbf{x};\boldsymbol{\theta}),\mathbf{y})\forall i$. To do this, the algorithm first evaluates $f(\mathbf{x};\boldsymbol{\theta})$, this is the *forward pass*. The loss function value can then be computed. Assuming the fact that all operations in the neural network, in units and the activation functions are derivable, the gradients for the last layer can be computed. It is then propagated to the beginning of the network, layer by layer and operation by operation, in the *backward pass* using the chain rule that states that if $y = g(x)$ and $z = f(g(x)) = f(y)$,

$$\frac{\partial z}{\partial x} = \frac{\partial z}{\partial y}\frac{\partial y}{\partial x}. \tag{2.16}$$

Explicitly computing this backward pass by hand can quickly become tedious in architectures with many layers and complex connections. For this reason, the main feature of the modern deep learning frameworks such as Tensorflow [37] or PyTorch [38] is their automatic differentiation engine to automate the backward propagation process and free the researcher to develop complex architectures.

Once the gradients have been computed, the gradient descent can occur. A *learning rate* a is set, that will define the strength with which the gradient is followed, and the parameter can be updated as follows

$$\boldsymbol{\theta}_i^{(t+1)} = \boldsymbol{\theta}_i^{(t)} - a\frac{\partial}{\partial\boldsymbol{\theta}_i}\mathcal{L}(f(\mathbf{x};\boldsymbol{\theta}),\mathbf{y}) \tag{2.17}$$

This process is iterative and repeated several time until convergence.

To improve the stability of the convergence, and also decrease training time by using parallel computing (e.g. in a GPU), a *batch* of size N inputs can be presented to the network, the result of which is aggregated before parameter update:

$$\boldsymbol{\theta}_i^{(t+1)} = \boldsymbol{\theta}_i^{(t)} - a\frac{1}{N}\sum_n^N \frac{\partial}{\partial \boldsymbol{\theta}_i}\mathcal{L}(f(\mathbf{x}_n;\boldsymbol{\theta}),\mathbf{y}_n) \tag{2.18}$$

Using all the available inputs in the same batch leads to the deterministic gradient descent algorithm, but it is rarely used in practice because it can involve an impractical amount of inputs. When $N = 1$, and by extension when fewer than all the possible inputs are in a batch, the approach is called stochastic gradient descent (SGD) because it involves randomly sampling the inputs to present to the network.

To reduce the chances of the training converging towards a local minimum and improve training stability, several variants of SGD have been proposed, that revolve around the addition of a momentum element [39] with the parameter γ:

$$\boldsymbol{\theta}_i^{(t+1)} = \boldsymbol{\theta}_i^{(t)} - m_i^{(t)} \tag{2.19}$$

$$m_i^{(t)} = \gamma m_i^{(t-1)} + a\frac{\partial}{\partial \boldsymbol{\theta}_i}\mathcal{L}(f(\mathbf{x};\boldsymbol{\theta}),\mathbf{y}) \tag{2.20}$$

This reduces the fluctuations in the gradient introduced by the stochastic approach. The most used algorithm variant, and the one used in this thesis, is Adam [40]. It requires two additional parameters β_1 and β_2 to compute moving average over the gradients and their squares and an ϵ value for computation stability:

$$m_i^{(t)} = \beta_1 m_i^{(t-1)} + (1-\beta_1)\frac{\partial}{\partial \boldsymbol{\theta}_i}\mathcal{L}(f(\mathbf{x};\boldsymbol{\theta}),\mathbf{y}) \tag{2.21}$$

$$v_i^{(t)} = \beta_1 v_i^{(t-1)} + (1-\beta_1)\left[\frac{\partial}{\partial \boldsymbol{\theta}_i}\mathcal{L}(f(\mathbf{x};\boldsymbol{\theta}),\mathbf{y})\right]^2 \tag{2.22}$$

$$\hat{m}_i^{(t)} = \frac{m_i^{(t)}}{1-\beta_1^t} \tag{2.23}$$

$$\hat{v}_i^{(t)} = \frac{v_i^{(t)}}{1-\beta_2^t} \tag{2.24}$$

$$\boldsymbol{\theta}_i^{(t+1)} = \boldsymbol{\theta}_i^{(t)} - \frac{a\hat{m}_i^{(t)}}{\sqrt{\hat{v}_i^{(t)}}+\epsilon} \tag{2.25}$$

2.2. Designing the appropriate dataset for supervised learning

The defining aspect of a ML algorithm is the fact that it learns how to operate based on data that is presented to it, and not based on the knowledge of the designer or the operator. This means that the properties of data used for training have a great impact on the performance ot the trained system. Some well built datasets, such as the modified modified National Institute of Standards and Technology (MNIST) database, that provide labelled black and white images of handwritten digits, have become standard benchmarks for machine learning

algorithms. This section presents generalities on how to select and prepare data for training.

2.2.1. The dataset and its partitioning

It is common practice in the literature to call *dataset*, the set of data points (of *examples*) that is available for training and testing as experience for a machine learning system over a particular task. In cases this dataset is experimental, it is finite, and usually, smaller than what the designer would like. In this case, the temptation would be to use all of it for training, improving the performance of the learned system by maximising the number of shown examples, but then, how could the system be tested? If the examples presented at test time are part of the set shown at training time, the ML system has trained on them and the test does not represent its ability to *generalise*, to perform on new data points. So a portion of the dataset needs to be reserved for testing, the *test set*, and kept separate from the *training set*. When many trainings are expected over the same dataset, for example to iteratively select a good performing neural network architecture, an additional aspect needs to be considered: The repeated training with a random initialisation could happen to create a system that perform specially well on the examples of the test set but not on new data points. To cover for this situation, the common practice is to isolate a third set, the *validation set* that is used in this situation to compare the performance of the various trainings, keeping the test set for final performance measurement. All these sets need to have the same distribution over the phenomenon of interest to be relevant for each other, so they are generally randomly sampled from the overall dataset. Finally, the splitting of examples between the sets need to be considered: A larger training set can increase the performance of the learned system, but reduces the confidence one can have in the test results. The common practice to tackle this trade-off is to use a split with 70% of examples for the training set, 10% for the validation set, and 20% for the test set. These considerations will be relevant for Chapter 3.

In Chapters 4 and 5, the dataset comes from a simulation environment, so this explicit dataset split is not necessary. New examples are generated in real time from the same distribution, as needed, for training validation, and testing in a process sometimes referred ar *online* learning. With offline training, the same training set is generally presented several times to give the network sufficient time to train. Going over the entire training set once is called and *epoch*. This term can also be used in online learning as a more manageable count of training iterations, the epoch length then needs to be defined, e.g. to 100000 iterations, and it does not represent a repetition of training examples.

2.2.2. Dataset construction

A important theme of this thesis, that will be declined in the following chapters, is the importance of using datasets that correctly represent the distribution of wanted features and expected measurement conditions, and avoid unwanted bias in the data. Fortunately, in most communication tasks, especially at the physical layer, issues with the dataset building would only lead to reduced algorithmic performances, with reduced robustness or increased error rates, without much direct impact on human lives. Construction issues are much more prominent and impactful in the field of image recognition, or more generally settings that directly involve people, so some examples in this section will come from these types of fields. These issues are beginning to be noticed, the authors of [41] providing a survey of datasets concerning issues in this field and ways to combat them.

Size

As a general rule with machine learning training, more data is always better. A bigger dataset allows for a better sampling of the true underlying distribution of the phenomenon of interest, as long as the other considerations of this section are not impacted, especially label balance and representativeness. And a better sampling of the underlying distribution allows the neural network to better train. Unfortunately, there is no equation nor guideline for finding the necessary minimum number of examples to use for training on a particular task. A dataset too small can lead to low generalisation capabilities, of even convergence issues, and when encountering this type of symptom (detailed later, related to overfitting), a first approach it to increase the dataset size. In physical layer communications, dataset gathering can often be automated, so gathering more data can be as simple as running a script. But when it is not the case, data augmentation [42] can be used to artificially create more examples. In images, this can involve adding cropped, translated or rotated copies of existing examples. A GAN can also be used to generate new synthetic examples mimicking the existing dataset. This simulated addition is not guaranteed to truly match what real new examples would look like and can lead to representativeness issues.

Labelling

For supervised learning, labels provide the only means to compute the performance measure. As with gathering new data, the physical layer communication field is fortunate in that the labelling process can often be automated, as in Chapter 3 and even more in the simulated environments of the later chapters. When that is not the case, this process can be time consuming and expensive because it needs to be done manually by humans, for potentially millions of examples and so often needs to be either outsourced or crowdsourced. Some CAPTCHAs can be used to help with this process [43], using the unwilling work of internet users. This handmade process can produce labelling errors. It can be seen in the ImageNet dataset [44], with the authors of [45], finding unimageable (corresponding to visible characteristics) labels, and even offensive or erroneous ones, by only studying the portion of the dataset related to people.Even automated processes can introduce errors due to noise or coding errors. A small percentage of erroneous labels may not negatively impact the trained network's performance but is can be a hidden cause of convergence issues.

Label balance

A dataset is balanced if there is the same amount of example for each class, or label. The effect of an unbalance in that regard is more easily apprehended on a classification task, but it can also appear in other types. On a toy example task of a binary classification, if 20 % of the training examples are of class 0, and 80 % are of class 1, a learning algorithm can converge to always predict class 1, and still obtain 80 % accuracy, even if the low number of examples does represent all the possible realisations of that class. So one has to ensure that the appearance frequency of all the classes is similar to reduce the possibility the rarest are ignored. If the dataset is unbalanced, but more data cannot be gathered to balance it, two options are available. Examples of the most common classes can be removed, but this had the drawback of reducing the overall dataset size with a risk of it becoming too small. Or the loss function can be adapted to weigh more heavily the rarest examples, to ensure that the

network learns proportionately more from them.

Representativeness

A neural network system may be expected to preform in a range of operating conditions of noise, interferences, signal source, ... To ensure this, the training dataset needs to have a number sufficiently large of examples over all these expected operating conditions. If this is not the case, the trained network may not be able to adapt to its new conditions. A good example of that can be found in the context of Chapter 3, where, when a network is trained to operate on signals coming from one receiver, a great loss in performance can be seen when changing that receiver.

As will be seen in Chapter 4 and 5, in some cases, training over all possible conditions is not necessary, or even harmful, especially for noise. in that case, training with a noise level too high can prevent the network from learning anything, and a level too low, does not force it to learn robust features. Introducing examples with conditions outside of the optimal range can dilute the useful examples and reduce trained performance.

The wireless communication community often works in simulated environment that model real world systems, without the need for time consuming and expensive experimental setups. The datasets used to train for tasks in this community follow this trend, with relatively few public experimental datasets such as [46].This certainly allows for a faster exploration of the capabilities of deep learning approach in this field, but the datasets are limited by the completeness of the models they rely on, and neural networks trained on these will need to be validated in the field before they can be accepted in production. The need to validate algorithms experimentally has always been present, but, as the synchronisation section of Chapter 3 shows, deep learning algorithms are able to get stuck on unexpected, and seemingly small hardware effects so the translation towards experimental may not be a smooth as with conventional approaches.

Bias avoidance

A neural network is lazy during training: it will learn to use the most distinguishing features available in the dataset, even if it is not actually a distinguishing feature in the real world or something that the designer want, or expect. A compelling example of the potential source, effect and gravity of such bias could be of a system learning to tell innocent from guilty people from their picture. If all the pictures of guilty persons are taken in prison, with jail uniforms of the same colour, and innocents' pictures are taken in the street, wearing everyday clothes, the system will certainly learn to discriminate based on the colour of the clothes, not on any hypothetical feature of the people themselves. Yet, the performance on the test set, taken from the same dataset, will be very high. It has been shown that, in ImageNet, the images corresponding to labelled people do not correspond to a balanced distribution of genders [45], and an automated tool is proposed to provide subsets with a restored balance. In this paper, the authors also note that a balanced distribution may not always be the goal, but more a distribution matching the real world, taking age groups as example. Another instance of this issue happened to Amazon's human resources team [47]. Their newly created recruiting tool, trained on past accepted resumes developed a bias in favour of male candidates, penalising resumes containing the word "women's". This was in a big part because the company had mostly accepted male employees in the past, and so the distribution of accepted resumes was

skewed in that direction. A possible biased situation in the communication field could occur for instance in a service handling various levels of quality of service (QoS) subscriptions at a cellular base station. Since high QoS is more expensive, thus subscribed by wealthier users, a large proportion of high-end, prestige devices, would be linked to this subscription level. So a biased system could learn to automatically affect better QoS to certain brands or models.

If a bias is known in advance, measures can be taken to reduce its impact on the final performance. Either the effect can be removed, or reduced to a unique realisation common to all examples in the dataset (this causes the risk of running into representativeness issues), or it needs to be purposely amplified, increasing its variation so much that the network cannot use it any more.

There isn't a unique method ensure representativeness and bias reduction, it needs to be tailored to the scenario, reflecting on what are the characteristics of the wanted signal, and what undesired elements could be present in the data. Chapter 3 focuses more on bias avoidance while the other two present more interesting representativeness examples.

Ethics considerations

When dealing with datasets involving humans or animals, ethics consideration quickly become important. The gathering and sharing of mass quantities of potentially private or sensitive data, especially through web crawling leads to many privacy questions. It can also question research ethics, a good example being found in the works of [48] that look at the high occurrence of offensive and non safe for work images in the widely used ImageNet set. This dataset has been frequently shown to be limited, as can attest the number of times it has been mentioned in this section, but this is in part because it is open to scrutiny by researchers. The authors mention that many dataset exist without this possibility, and so could be even more problematic, even more when considering the amount of money involved in the gathering, maintaining and exploitation of these sets. Regarding the avoidance of biases, some of them are actually representative of the biases and inequalities of the society. So should the datasets accurately represent the current state of affairs, or should they represent what things should be? And then, what would that be?

Fortunately, the physical layer communications field is quite far from direct involvement with humans, so these difficult questions should not impact the research very much. But it is still a possibility, for instance, device localisation, and even the transmitter fingerprinting of Chapter 3 could quickly impact user privacy.

2.3. Deep learning parametrisation

Another recurring question in this thesis, especially in the last two chapters, is: How to design and parametrise a neural network (NN) to tailor it to the task at hand and allow for good performance? No hard rule exist to predict what type of deep learning system performs the best on a given task and how to tune it. But some guidelines do exist, and knowledge of the task and the data the network operate on can be used to guide the designer. This section discusses the various parameters a deep learning designer has to tune.

2.3.1. Architecture selection

The classical version of the universal approximation theorem states [49] that a single layer perceptron (dense layer) can approximate any continuous function. Using a non-polynomial continuous activation function $\sigma : \mathbb{R} \to \mathbb{R}$, with input dimension n and output dimension N, for any $f : \mathbb{R}^n \to \mathbb{R}^N$ continuous, any compact subset K in $\mathbb{R}^n$ and $\epsilon > 0$ arbitrarily small, there exist a continuous function $\hat{f} : \mathbb{R}^n \to \mathbb{R}^N$ of the form:

$$\hat{f}(\mathbf{x}) = \sigma(\mathbf{W}^\mathsf{T}\mathbf{x} + \mathbf{b}), \mathbf{W} \in \mathbb{R}^{n \times N}, \mathbf{b} \ \in \mathbb{R}^N \tag{2.26}$$

such that can approximate f arbitrarily closely:

$$\sup_{\mathbf{x} \in K} \left|\left| f(\mathbf{x}) - \hat{f}(\mathbf{x}) \right|\right| < \epsilon \tag{2.27}$$

And this has also been shown for other types of activation functions, and network topologies, culminating in [27]. So, even the simplest architecture, when big enough, has the potential to perform well on a task. But there is no guarantee that, for such an architecture, the number of neurons needed is manageable, or that it will train properly on the available data or converge at a reasonable speed.

Knowledge of the task and of underlying properties of the data, sometimes also referred as *expert knowledge*, can be used to design architectures that perform well. The convolutional layers introduce translation invariance in the way the filters operate, and greatly reduce the number of weights to train by reusing them at each application of a kernel over the input tensor. The recurrent layers are effective on sequential data, with patterns that spread over a large distance, due to their memorisation capabilities, in contrast to the convolutional layers, that are limited to the scope of the kernels. They also can extend to input sequences of different lengths and generalise on them, while dense layers would need additional neurons, parameters, and thus training to operate on a changed input dimension.

A good example of an architecture guided by expert knowledge can be found in the radio transformer network (RTN) [50] where a first part of the network, composed of convolutional followed by dense layers are used to estimate a few parameters. These parameters are then input into a non-learned transformation layer where they are used as parameters to phase, frequency and sample timing offset correction operation on the original input before feeding it to the main decoding network section. This forces the network to learn the parameter estimation functions, but frees it from the actual correction functions, thus reducing the complexity of the task to learn, and speeding up the training and increasing performance.

Most of the tasks studied in this thesis, namely the transmitter identification of Chapter 3, the active user detection of Chapter 4, and the receiver side of the transmissions of Chapter 5 have something in common: they are instances of non-binary classification tasks. There are two main strategies available to perform such a task, a direct multi-class or multi-label classification, or the conversion into multiple binary classification operations. And when using a binary classifier, an interesting architecture variant is available, relative to the way knowledge about the possible classes is stored for later identification, a comparison:

- **Comparison**: The identifying network has 2 inputs: the signal to identify, and a known one. It then outputs a score, rating how close the two signals are to each other. This is repeated with all the known possible signals to find the closest match. In that case, knowledge about the possible cases is stored inside the reference signals, not in the

comparing system itself. If characteristics tend to change over time, the reference signals can be updated regularly to keep up with the changes. This approach makes it simpler to work in a dynamic environment, simply by storing new reference signals. But this might require a large memory space, directly scaling with the number of signals to be identified.

- **Direct classification**: The identifying system has only one input: the signal to identify. It then outputs a score rating how much each possible candidate matches to the received signal. In this case, knowledge about the possible cases is stored by the identification system itself (e.g. as weights in a NN), so updates can be harder to accomplish to accommodate for variations of characteristics or new users. On the other hand, identification is done in one step instead of computing a score for every reference signal, hence shortening the processing time and amount of memory required.

In terms of high level operation, the two approaches operate multi-class classifications, the comparison architecture contains an explicit storage system, while the storage in the direct classification is embedded, and merged with the rest of the neural network's operations. The architecture type selected in this thesis are direct classifications, because the scenarios encountered do not require a dynamic number of classes, and they are expected to be used online, inside or along the decoding chain, and as such require a latency as low as possible.

2.3.2. Loss function choice

In the overall machine learning model defined at the beginning of this chapter, the performance measure P is closely linked to the task to optimise for, but this performance measure is a high level metric, such as accuracy, to evaluate overall system performance. The gradient descent algorithm requires a loss function $\mathcal{L}$ to minimise for its internal operations. The loss function and the metrics may be different and several metrics may be used to evaluate different elements of the system performance, while a unique loss function is used in the optimisation algorithm.

Several default loss functions exist, adapted to certain types of tasks. For instance mean squared error (MSE) is used for regression operations, defined using the previous notations and $f(\mathbf{x};\theta), \mathbf{y} \in \mathbb{R}^N$ as $\mathcal{L}_{MSE} : \mathbb{R}^N \to \mathbb{R}$:

$$\mathcal{L}_{MSE}(f(\mathbf{x};\boldsymbol{\theta}), \mathbf{y}) = \frac{1}{N} \sum_{n=1}^{N} (\mathbf{y}_n - f(\mathbf{x};\theta)_n)^2 \tag{2.28}$$

Its cousin, the mean absolute error, can also be used in this type of situation, replacing the square operation by an absolute value. These two functions have similar properties, being always positive, and accepting any real label value, but they differ in that, for $(\mathbf{y}_n - f(\mathbf{x};\theta)_n) > 1$, the MSE will lead to a higher loss, and so a higher gradient and learning rate, while when $(\mathbf{y}_n - f(\mathbf{x};\theta)_n) < 1$, the MAE will have the higher losses. So the MSE should perform better in conditions of high absolute error, and the MAE better when the label and predicted values are close. For very high errors, the squaring operation of the MSE may lead to gradients that are too high for a smooth convergence.

For classification tasks, the default option is to use of a cross-entropy loss function. In binary tasks, only one class is present, so the label $y \in 0, 1$ and $f(\mathbf{x};\theta) \in [0, 1]$. The assumption

on the output format of the network is generally met by having a sigmoid activation function on the last layer.

$$\mathcal{L}_{CE_{bin}}(f(\mathbf{x};\boldsymbol{\theta}),y) = -(y\log(f(\mathbf{x};\theta)) + (1-y)\log(1-f(\mathbf{x};\theta))) \tag{2.29}$$

In the classification task, the maximum likelihood estimator can be seen as minimising the Kullback-Leibler (KL) divergence, or the dissimilarity between the distribution of the data labels and the distribution of the network output

$$D_{KL}(p(y)||p(f(\mathbf{x};\boldsymbol{\theta})) = \mathbb{E}_y\left[\log(p(y)) - \log(p(f(\mathbf{x};\boldsymbol{\theta}))\right] \tag{2.30}$$

A training batch realises a Monte-Carlo sampling evaluation of the cross-entropy $\mathbb{E}_y\left[-\log(p(f(\mathbf{x};\boldsymbol{\theta}))\right]$, so minimising it is equivalent to minimising the entire KL divergence, since the left term is not dependant on the parameters. So, a network using a cross-entropy (CE) loss function converges towards a maximum likelihood detector.

The multi-label version can be seen as a succession of independent binary classifications, leading to an extension of the binary cross-entropy with the label $\mathbf{y} \in 0,1^N$ and $f(\mathbf{x};\theta) \in [0,1]^N$:

$$\mathcal{L}_{CE_{bin}}(f(\mathbf{x};\boldsymbol{\theta}),\mathbf{y}) = -\frac{1}{N}\sum_{n=1}^{N}\mathbf{y}_n\log(f(\mathbf{x};\theta)_n) + (1-\mathbf{y}_n)\log(1-f(\mathbf{x};\theta)_n) \tag{2.31}$$

The multi-class version is slightly different to enforce the assumption that only one class is active at a time. For this, a special activation function is used on the last layer of the network, the softmax : $\mathbb{R}^N \to [0,1]^N$ defined with $\mathbf{p} \in \mathbb{R}^N$ and for the n^{th} element as [51]:

$$\text{softmax}(\mathbf{p})_n = \frac{e^{\mathbf{p}_n}}{\sum_{m=1}^{N} e^{\mathbf{p}_m}} \tag{2.32}$$

This softmax function gives the output of the network the property of a probability vector, i.e. $\sum_n^N \text{softmax}(\mathbf{p})_n = 1$. The *categorical* cross-entropy variant can then be used as a loss function, with the labels in the form of a *one-hot* vector $\mathbf{y} \in 0,1^N, \sum_{n=1}^{N}\mathbf{y}_n = 1$. All the vector elements are zeros, except for a one in the index of the active class.

$$\mathcal{L}_{CE_{cat}}(f(\mathbf{x};\boldsymbol{\theta}),\mathbf{y}) = -\sum_{n=1}^{N}\mathbf{y}_n\log(f(\mathbf{x};\theta)_n) \tag{2.33}$$

The loss function is often also used to guide the network towards having certain properties. For instance, to reduce the network memory footprint, increase its execution speed and/or operate on specific hardware, the weights of the network can be quantised, reducing the number of bits each parameter is stored on. But the quantisation step changes the parameter values, affecting the network operation. To reduce this quantisation error, the authors in [52] propose the addition of an additional quantisation error term to the loss function. For the i^{th} parameter θ_i, whose closest quantised value is θ_i^q, the overall loss function would be as such:

$$\mathcal{L}_Q(f(\mathbf{x};\boldsymbol{\theta}),\mathbf{y},i) = \mathcal{L}(f(\mathbf{x};\boldsymbol{\theta}),\mathbf{y}) + \alpha\left|\left|\theta_i - \theta_i^q\right|\right|_2 \tag{2.34}$$

This require a parameter α to set the strength with which the added term is followed. More

complex and efficient methods have been proposed, such as in [53] to tackle this problem, but this technique is a simple example of additional terms in the loss function.

Such approach can be used to coerce the network towards a certain type of output, or even properties of intermediate hidden layers. It can be noted that it does not guarantee that the desired properties will be obtained at the end of the training, depending on the capabilities of the network, or the strength parameter. So, a strict enforcement may still be needed after training. The most common use of additional terms is for regularisation, that will be discussed in the next section.

2.3.3. Hyperparameter tuning

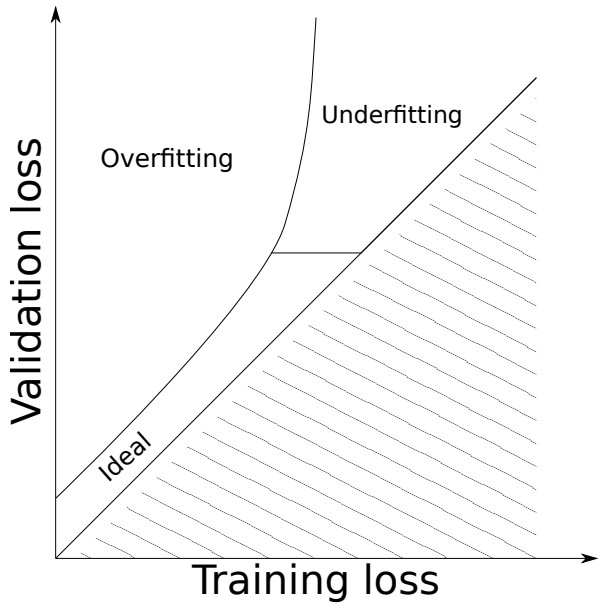

Figure 2.7.: Overfitting and underfitting zones. The zebras correspond to a case that should not appear: high training loss but low validation loss.

The parameters of a neural network are the weights and biases, everything that is automatically trained with gradient descent. But there are many *hyper*-parameters that need to be set, such as, the architecture and size of the network, the activation function used at each layer, learning rate and other optimiser parameters, the loss function and the parameters that set the relative weights of their various terms,... The overall architecture generally comes from expert knowledge of the task to solve, but the actual tuning, the search for a well performing set of hyperparameters is an empirical and iterative process. For this, examination of the evolution of the loss function during training on both the training set and the validation set can reveal symptoms and direct some changes to make to the hyperparameters. Comparing the performance on training and validation sets allows to evaluate the generalisation capabilities of the network and several cases can occur as shown in Fig.2.7:

- **Underfitting**: The network performs poorly on both the training and validation sets. The evaluation of what "poorly" represents depends on the specific task and may involve comparison with previous results, or comparison with human performance

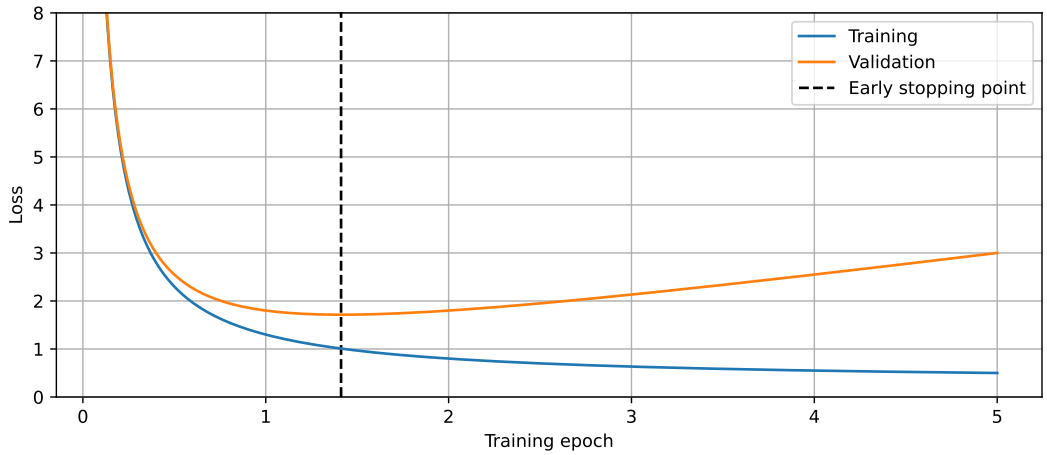

Figure 2.8.: Typical example of a training with overfitting

on the dataset. A possible cause is that the network is not expressive enough, it is not able to represent a complex enough function. Increasing the network size (width and/or depth) or, if applicable, reducing the weight of the regularisation terms in the loss function, could increase this expressivity.

- **Overfitting**: The network performs well on the training set, but not on the validation set. When looking at the loss evolution during training, the training and validation curves diverge after some time, as in Fig. 2.8, and the validation line either stops decreasing or even start increasing. This is the sign that the network is overfitting to the training set, it as learned the distribution of training examples too closely, and is not able to generalise to new examples. To combat this, *regularisation* can be introduced.

Regularisation In general, regularisation is defined by [14] as "any modification that we make to a learning algorithm that is intended to reduce its generalisation error but not its training error". Many strategies exist for that, and some of the most common are presented here. Parameter norm regularisation adds a regularisation term to the loss function that penalises the size of the parameters. Typically, this is done using the L1 and L2 norms of the parameter vector.

$$\mathcal{L}_{L^1}(f(\mathbf{x};\boldsymbol{\theta}),\mathbf{y},i) = \mathcal{L}(f(\mathbf{x};\boldsymbol{\theta}),\mathbf{y}) + \alpha|\theta|_1 \tag{2.35}$$

$$\mathcal{L}_{L^2}(f(\mathbf{x};\boldsymbol{\theta}),\mathbf{y},i) = \mathcal{L}(f(\mathbf{x};\boldsymbol{\theta}),\mathbf{y}) + \alpha\,||\theta||_2^2 \tag{2.36}$$

These terms can be added independently for each layer. This method relies on limiting the expressivity of the network to prevent it from fitting exactly to the training set. Another option to reduce the network expressivity is to introduce *dropout* [54]. During training, a layer with dropout sees some units being randomly deactivated. This can be seen as creating a set of networks, with every possible active unit combination and training them in parallel. At inference, everything is activated, and so all the virtual sub networks vote towards the final prediction, smoothing the output. It can also be seen as a way to force the units to operate well in many conditions. A simple technique not limiting the network expressivity

is to perform *early stopping*. The training is then, as the name implies, stopped early, when the validation error stops improving for a number of iterations. Finally, as mentioned when discussing dataset size in Sec. 2.2,*data augmentation* can be used to increase the dataset size, either by applying small transformations, or adding noise to original examples.

The high amount of hyperparameters lead to a very high dimensional search space. Several approaches can be attempted to automatise the search process [55], such as grid or random search, genetic algorithms, reinforcement learning,... in a process sometimes referred to as *meta-learning*. But these technique require a large number of trainings to give results, leading to a very large consumption of resources, in time and processing power.

2.3.4. Limits to Machine learning's usefulness

Machine learning, and especially deep learning approaches have inundated the research field and industry in the last few years, being applied to many problems with great success. But while the universal approximation theorem guarantees that a neural network should be able to perform well on most tasks, it does not guarantee that it will converge to a better solution to other approaches, or, even, that it will be possible to implement it. By design, machine learning requires data to train, and it requires a lot of it, usually with a time consuming and expensive labelling process. So, when such a dataset is not available, the use of such algorithm becomes impossible, or at least highly impractical. In the same vein, the interest of learning approaches comes from the fact that it requires little knowledge of the underlying process of a given dataset to perform well on it. It does not require any prior model, but learns it from the data. So if a good model is already available, it can become simpler and quicker to design and implement a traditional expert algorithm for the task, than to go through the lengthy process of designing and tuning a machine learning algorithm. It is also the case when a good understanding of a decision process is needed, as the inner workings of most machine learning systems, and especially neural networks, are extremely difficult to understand, despite ongoing works towards explainability [56], [57].

2.4. Tools for Software defined radio implementation

The works presented in Chapters 3 and 5 involve the gathering of experimental datasets (for Chapter 3), and the setup of an experimental testbed for online learning outside of simulation. These required the implementation of software defined radio signal processing chains to generate, receive and process data. This section describes the two main tools used in this instance: the FIT/CorteXlab experimentation room, and the GNU Radio software defined radio, free and open source framework.

2.4.1. FIT/CorteXlab

Conducting radio experiments require hardware, software to control that hardware, and official authorisations to operate on specific frequencies. Elements that take both time and money to obtain. The FIT/CorteXlab platform aims at helping researchers in that regard by providing a free shielded experimentation room of about 180 m^2 filled with SDR devices and computers to control them, usable via remote internet access. The room shielding allows more than 80 dB attenuation from the outside world, and thus, no authorisation is needed to

experiment, on any frequency band reachable by the hardware. Some radio wave absorption is achieved by foam on the ceiling and walls to prevent unending refections on the isolating material, but it is not anechoic. The floor, maintenance equipment, hardware and railings provide raw metallic surfaces for reflections and multi-path propagation conditions. It is host to 22 National Instruments universal software radio peripheral (USRP) 2932 SDR devices equipped with wide band antennas capable of transmission frequencies between 400 MHz and 4.4 GHz, as well as 16 *picoSDR* devices for 2×2 and 4×4 MIMO operations up to 3.8 GHz. Each device is controlled by its dedicated Neousys Nuvo 3000 computer to process the raw samples. And all of these computers, also called nodes, are linked by Ethernet cables into the same network, so reliable back channelling is possible and used in Chapters 3 and 5.

Access to the testbed is done online, connecting to the FIT/CorteXlab gateway server. The code to be executed on the nodes is uploaded to this gateway, along with a scenario file describing what program is to be executed, on which node. To begin an experiment, reservation of the platform on the provided service is necessary, to ensure the absence of interference, and after this, a task is created and the provided software handles deployment of the code on the desired nodes, execution of the programs, and gathering of the results for consultation on the gateway, and clean-up of the nodes for subsequent experiments, all according to the scenario file. This operation allows the researchers to focus on the experiment design without worrying about the inner workings of the platform.

The dedicated room, with online access eliminates human presence in the room outside of maintenance activities, reducing movements of any kind to a minimum. This, combined with the isolation from outside interference, provides extremely stable propagation conditions for experiments with respect to the fading scenario as a whole (path-loss, shadowing and small-scale fading), as shown in [58], leading to reproducible experiments.

2.4.2. GNU Radio

The software defined radio paradigm involves offsetting as much signal processing from the radio devices as possible, to replace it in software. The use of this paradigm in this thesis allows the implementation of custom physical layers with specific signal processing, and new, dynamic modulation schemes, something that would not be tractable with conventional radio implementations. The radio hardware is provided by the FIT/CorteXlab platform described above, but the software signal processing element needs to be able to interface with the hardware, and efficiently operate on the gathered samples. To help with that, GNU Radio is used. It is a free and open source software providing a framework tailored to SDR processing. It operates *flowgraphs*, computation chains made of a graph of computation block, each one doing its own operation (filtering, modulation,...), written in either Python, or C++ when processing speed is required. The framework handles the transmission of samples from one block to the other, so block development can be done in isolation, agnostic to the constitution of a particular flowgraph. This block-based processing allows for modularity, reusing blocks from one application to another, and even allows reconfiguration at runtime. On top of this framework simplifying signal processing programming, GNU Radio comes with a library of pre-made, commonly used blocks, allowing for the creation of simple flowgraphs without programming, some of them providing graphical interface elements to monitor the data flow, such as time or frequency scopes, and widgets to interact with parameters of blocks at runtime. It is also accompanied by the GNU Radio Companion tool, a graphical user interface for building flowgraphs by dragging, connecting, and parametrising blocks for code

free flowgraph definition. Finally, an important community exists around this tool, with a large amount of *modules* (sets of blocks sharing a common theme) being developed and shared online for specific applications, such as interfacing with existing protocols (LoRa, Wifi,GSM,...).

— 3 —

Transmitter Identification

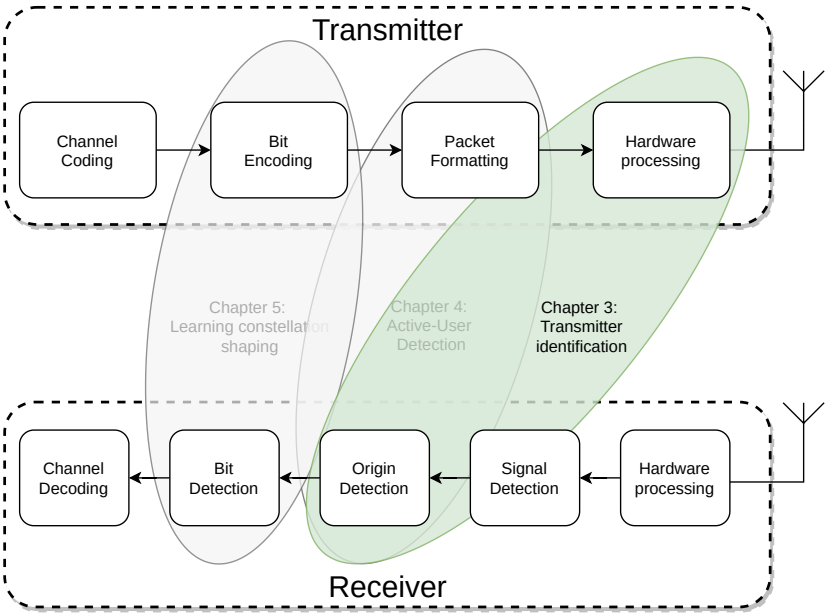

Figure 3.1.: Position of the chapter in the thesis outline

As mentioned in the introduction, the first detection step in a receiver processing chain that is relevant to the application of machine learning is the detection of the origin of a signal as it is necessary to properly use the decoded signal, or even to use the correct decoding scheme. This chapter tackles this problem from the viewpoint of transmitter identification, or transmitter fingerprinting. The detection of the identity of the transmitter that is active amongst a pool of known devices. The work presented in this chapter has lead to the publication of a conference article [11] and the future submission of a

journal paper [10].

3.1. Introduction

Transmitter identification has been a crucial topic since the dawn of radio communications. Unlike wired communications, where communication integrity is ensured by the physical medium, the broadcast nature of electromagnetic waves requires to securely identify the transmitter of a signal. Since World War II, radio identification is mainly based on cooperation with the transmitter, thanks to an identification code. This method of identifying transmitters is inherently prone to problems since it depends on decoding a portion of the signal itself and also on *trusting* that the actual source of the signal is not trying to impersonate a trusted transmitter.

The problem of identifying a transmitter has become even more important nowadays due to two factors. First, the widespread availability of low cost, programmable radios, such as software defined radios (SDR), has opened the radio spectrum to a large public. While still not very widespread, security attacks such as man-in-the-middle (MITM) are utterly possible with SDR technology [59], or simply by altering medium access control (MAC) layers of popular radio devices [60]. Second, the sheer number of radio devices accessing the spectrum is enormous and tends to become even greater with the undergoing massive deployment of internet of things (IoT) devices. To identify all these devices with unique identification codes means that very long codes must be used. Such codes will take up valuable resources and, as already stated, are prone to attacks by malicious transmitters, see for instance [61].

To solve these problems, other means of identifying transmitters must be sought. Preferably, these identification techniques should not depend solely on transmitted identification codes but rather rely on physical characteristics of the transmitting hardware. This is indeed possible since transmitters are built with discrete components which are produced to a certain tolerance around their nominal values, creating a slight but perceivable variation of the radio signal, even between devices of the same brand and model. The compound effect of these and other factors can be seen as a fingerprint of the transmitter, that could be effectively recognised by a receiver.

The literature on physical layer transmitter identification covers the comparison and classification strategies discussed in the previous chapter, as can be seen in Table 3.1. On the comparison side, most approaches directly use the channel effects for identification [63]–[68], while [69] does not include them in their simulation. Most interesting in the scope of the present paper, the authors of [70] introduce a new feature: Amplitude of Quotient (AoQ) that looks at the variation of a Wifi preamble signal over two consecutive frames. This allows to greatly reduce the channel dependency since, from one channel to the other, the variation between two concurrent frames stays the same. Unfortunately, this approach is difficult to implement in an IoT scenario where devices can transmit isolated packets with potentially large time gaps between them. For this case, a one-shot identification system is needed.

Most of the works on the subject focus rather on classifiers, with many experimental setups, over a wide range of hardware platforms (phones, USRPs, Wifi, IoT and Zigbee devices), using a variety of preprocessing techniques and mostly using the steady state features of transmitted packets. Although the range of demonstrated applications seems very promising for real-life usage cases, the vast majority of studies do not consider the effects of a non-static channel on the identification performance. The effects of channel variations can be first seen

Table 3.1.: Comparison of literature papers

Paper	Approach	Channel	Main feature	Preprocessing	Devices	Algorithm	Type	Dataset
[62]	Analytical	No	Transient	No	28	N/A	VHF	Exp
[63]–[67]	Comparison	Used	CSI	N/A	2	Expert	N/A	Simu
[68]				Chan features	?	GMM	USRP	Exp
[69]		No	IQ imbalance	No	5	CNN	N/A	Simu
[70]		Yes	Steady state	AoQ	15	DNN	Wifi	Exp
[71]	Classification	No	Transient	Hilbert	8	PNN	Wifi	Exp
[72]				Bispectrum	3	SVM	USRP	Both
[73]			Many	Feat. extraction	4	NN,SVM		Exp
[74]			Amplifier	FFT	7	NN		Both
[75]			Steady State	Hilbert	5	CNN	N/A	Simu
[76]				Feat. extraction	10000	DNN	N/A	
[77]				VMD	20	LSVM	Phones	Exp
[78]				Handcrafted	6	Expert	Custom	
[79]				STFT	4	SVM	USRP	
[80]				Minkowski	9	SVM,k-NN		
[81]				Recurrence plot	11	CNN		
[82],[83]				No	5,6	ML		
[84]					8	k-NN		
[85]					6	NN	Zigbee	
[86]					8	NN	SDR	
[87]				Perm. entropy	8	k-NN,SVM	Custom	
[88]				Error signal	7	CNN	Zigbee	
[89]				Denoiser	27			
[90]				Multisampling	52			
[91]		Seen		CTF	54	NN		
[92]				FFT	20	CNN	USRP	
[93], [94]		Yes		No	16			
[95]					20			
[96]					19	Complex CNN	Wifi,ADS-B	Simu
[97]					10,50,250,5000	CNN	Wifi	Both
[Ours]					21		USRP	Exp
[98]	Authorisation	No	Steady state	Partial EQ	1049	CNN	Wifi,ADS-B	Exp
[99]		Seen	Preamble	No	163		Wifi	

in [91], where, even though the authors show the stability of learned features (involving 54 transmitters) by testing their network with new data 18 month after it was trained, they encounter a drop in performance when changing the location of the transmitters, due to a drastic change of the channel. The effect of it is also noticed in [92], with segments of the large proposed dataset gathered over several days in a open environment, and performance reduction noticed by the authors when trained on it.

The authors of [93], [94] created a dataset using 16 USRP devices in a realistic room used to train a CNN with good results. Unfortunately, any change in the environment (position of chairs or people moving) prevents the generalisation of the neural network to a new dataset. To counter this effect, additional artificial impairments are embedded in the transmitted signals to improve performance and a protocol is devised to choose those impairments, in a similar manner as for cryptographic keys. To the best of our knowledge, this paper is the only one that provides the dataset they use in their experiments. To the same end, the authors in [95] introduce a FIR filter in each transmitter and use a feedback link from the receiver to optimise their parameters to increase identification accuracy at the receiving CNN. The latter two approaches effectively increase robustness to channel changes, but they reduce the strength of their security claims: as with the current approach, a malicious user could tune its own system as easily as a registered one. Recent works aim at tackling the channel problem with data augmentation [96], [97], with dataset signals being injected through simulated channel models of various parameters before classification. The authors of [96] stay in a simulated environment, but they introduce the use of complex valued convolutional neural networks for this classification task, while the others test their data augmentation pipeline on an external experimental dataset [97], leading to performance accuracy gains.

The works presented above aim at identifying individual transmitting devices. A related task is to only distinguish two categories: authorised, and not authorised, leading to a simpler definition of the classification task, and opening the way to an higher number of devices, at the cost of removing the possible overhead reduction claims, to the profit of the security aspects. The authors of [98] use their own dataset with a large number of devices to train real valued convolutional neural networks, operating on Wifi and automatic dependent surveillance-broadcast (ADSB) transmissions, and show good scaling capabilities by having up to 500 device in the authorised category alone. The work in [99] evaluates the use of various modified classifiers to tackle the authorisation task with openset methods, also using CNN architectures on a Wifi dataset with up to 50 known authorised devices. The authors also note the negative impact of a changed channel on performance.

Two papers that stand out from Table 3.1 are [62] which is a study of possible characteristic features of transient signal in VHF radio, and [100], where the authors, instead of focusing on getting the best possible accuracy on their dataset, attempt to predict the size needed for it. The precise numbers reached in that paper are specific to their neural network design and scenario but they provide a rule-of-thumb for choosing dataset sizes.

The approach used in this chapter is to train a CNN classifier using raw IQ samples without preprocessing and from experimental datasets to ensure that all possible features in the signal can be used by the neural network. The network is made to cope with the channel variation effects natively, without having to introduce any form of additional information at the transmitter, by purposely introducing channel variations in the training dataset. The approach used here can be seen as a data augmentation technique, similar to [96] and [97], but with the augmentation of channel parameters being from physical changes in the experimentation room, not through a simulated channel model. The remainder of this Chapter

is divided as follows: Section 3.2 specifies the problem herein addressed, then Section 3.3 describes the implementation chosen to tackle it, from the experimental dataset gathering, to the NN architecture used for identification. Section 3.4 shows the results obtained using this implementation, and Section 3.5 analyses some specific effects of synchronisation between transmitter and receiver. Finally, Section 3.6 concludes this chapter and elaborates on possible future contributions.

3.2. Problem Formulation and classifier selection

3.2.1. Problem statement

The transmitter identification problem consists of a set of N transmitter *nodes*[1], and a set of M *identification receivers*[2]. The k^{th} node sends a complex discrete sequence, $x_k(t)$ and the i^{th} identification receiver receives a complex discrete sequence $y_i(t)$. Throughout this work, we refer to "MonoRX" in the case where $M = 1$ and "MultiRX" in the case where $M = N$. In regular systems, a part of $x_k(t)$, namely the identification field in the packet header, is used to identify the node. Unlike those systems, here we consider that $x_k(t)$ does not contain any information enabling the identification of the node, so x can be considered independent of k and we define $x_k(t) = x(t)\ \forall k$.

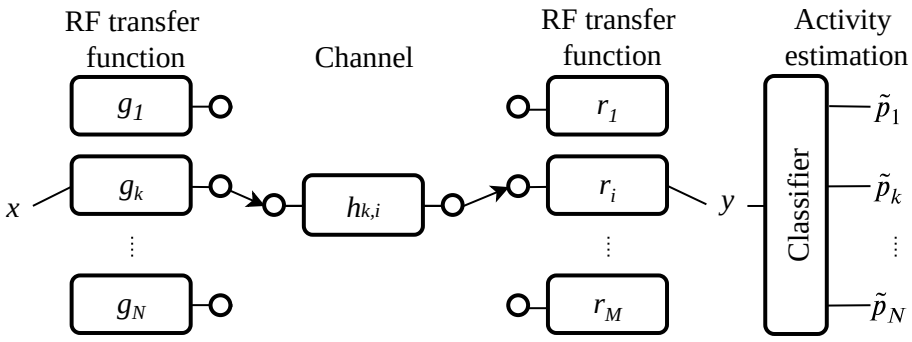

Figure 3.2.: System model

The sequence x is then processed by the radio frequency (RF) chain of the k^{th} radio device, whose overall RF transfer function g_k is unknown and can be non-linear. This RF transfer function corresponds to the compound effect of all signal formatting such as conversion from digital to analog domain, mixing, filtering, and amplification.

The produced signal is transmitted through a channel $h_{k,i}$ between node k and the identification receiver i, all channels having unknown input responses. The identification receiver then applies an RF transfer function r_i which, as for the node, can be non-linear. This leads to the observation of a sampled baseband signal $y(t)$, from which decision of the active node Id has to be taken. The overall model can be seen in Fig. 3.2. Note that in this setup, possible collisions are not considered and during each slot only one node and one identification receiver are active, chosen randomly, leading to the following system equation:

[1]Henceforth all transmitters are called *nodes*.
[2]Henceforth the receivers are called *identification receiver*.

$$y(t) = r_i(h_{k,i} \times g_k(x(t))) \tag{3.1}$$

With k from $K \sim \mathcal{U}_{\mathbb{N}}(0, N)$ the active node and i from $I \sim \mathcal{U}_{\mathbb{N}}(0, M)$ the active receiver, with $\mathcal{U}_{\mathbb{N}}(a, b)$ the uniform distribution over the integers between a and b (b not included).

Now the question addressed in this chapter can be stated: Is there enough information in the observed signal $y(t)$ to blindly find k without using the channel signature, but exploiting only the signature of the RF transfer functions g_k ?

The objective is then to design a system able to compute classification functions $\tilde{p}_k = f_k(y(t))$ where $\tilde{p}_k$ the estimated probability that the k^{th} node is active. To evaluate the performance of such a multi-class classification system, it is standard practice to use the categorical crossentropy loss function [14]:

$$\mathcal{L} = -\sum_{a=1}^{N} \mathbf{1}_{[a=k]} \cdot \log\left(\tilde{p}_k\right). \tag{3.2}$$

3.2.2. DL based classifier

A system able to minimise (3.2) can answer to the first half of the question at hand: Is there enough information in the observed signal y to blindly learn about the ID of the active node? The other half comes from a careful construction of dataset, as described later.

To do that, the approach chosen is to train a NN classifier operating on the raw observed signal y and directly outputting estimated activity probabilities $\tilde{p}_k$ for all possible nodes. The use of deep learning is fully justified: first the features of the function g_k we want to rely on are unknown and suspected to be non linear, and an experimental setup using the FIT/CorteXlab, as described in the next section, allows to get a large database of real signals for training, which is the first of its kind at the best of our knowledge.

No preprocessing is done on the raw data to avoid filtering any unexpected, but useful feature that may be present in the dataset.

The selection of a network architecture type relates to the assumptions made on the characteristics of the input signal. In this case, with the same reasoning as with preprocessing, the only assumption is that the signal is correlated in time, leading to the selection of a 1D CNN type network. This is because small scale memory effects can be expected, e.g. in the amplification stages. To better handle non correlated elements, the last layers of the network are general purpose dense layers.

The network is trained using the loss function (3.2), but the use of that loss function alone does not allow to discriminate between the effects of g_k and $h_{k,i}$ since they both depend on k. To permit such a discrimination, a training dataset will be specifically built in the next section to reduce the impact of $h_{k,i}$ in the learning process.

3.3. Implementation

To properly train a neural network for the identification task, a large dataset that is properly unbiased and correctly labelled is required. In this section, the data collection methodology used to achieve a suitable sample set for our identification needs is described: first, the experimentation room is presented, then the overall system architecture controlling the operation of the nodes and receivers. This is followed by the description of the scenarios for

gathering data, allowing to study the impact of different parameters on the identification performance. Finally, the exact NN architecture and training on the obtained datasets is presented.

3.3.1. Experimentation setup

Experimentation room

The complete system, comprised of a full set of nodes and the identification receiver, is tested within the FIT/CorteXlab testbed described in 2.4.1. Using FIT/CorteXlab allows to control the propagation environment as well as the interference profile, which in turn enables full control of the generated datasets. The nodes, as well as the identification receiver, are implemented on the radios of the NI USRP-2932 type. An optional synchronisation can be achieved over all SDR, effectively synchronising all sampling clocks of all SDR to evaluate the impact of synchronisation errors as detailed in Section 3.5. Since the propagation environment of the FIT/CorteXlab experimentation room is rather static, a metallic covered Turtlebot robot, as seen in Figure 3.3, has been programmed for this work to move according to a random-walk model. This allows to create time varying alternative propagation paths leading to channel diversity inside the room.

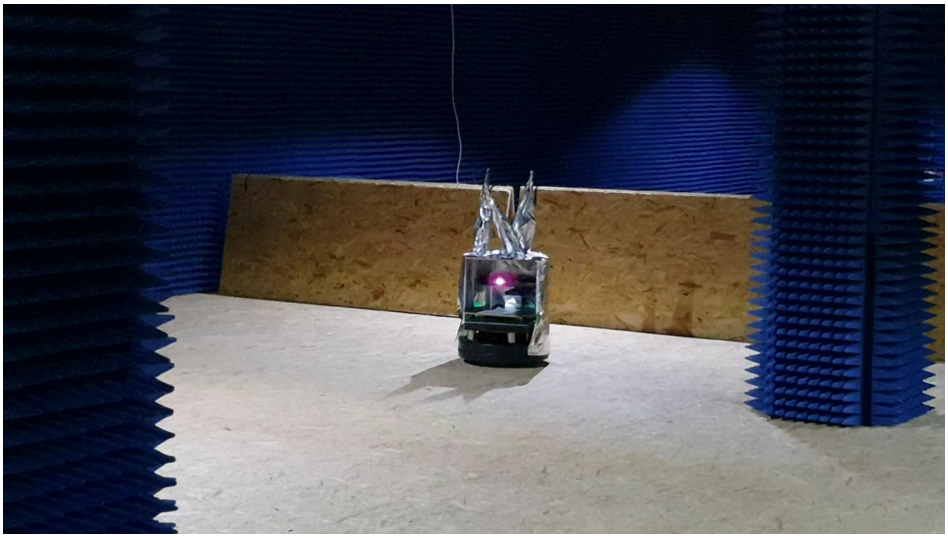

Figure 3.3.: The Turtlebot robot with the metallic sheets, inside of FIT/CorteXlab

The nodes are distributed as in Figure 3.4, where the position of all USRP nodes' antennas are indicated by the crosses with the node numbers, and the footprint of the FIT/CorteXlab experimentation room is delimited by the dotted line. The nodes' antennas are distributed onto a grid with a step size of 1.8 m^2. Among all active USRPs in the experimentation room, one takes the role of the identification receiver while the remaining behave as nodes. The choice of which node takes which role can be defined in the experimentation scenario description file.

Previous works characterised the channels in FIT/CorteXlab, such as [58] and [101]. In [101], the authors provide a measurement of the power-delay profiles observed in the room. Measurements made at a sampling frequency of 10 MHz (low resolution measurements for

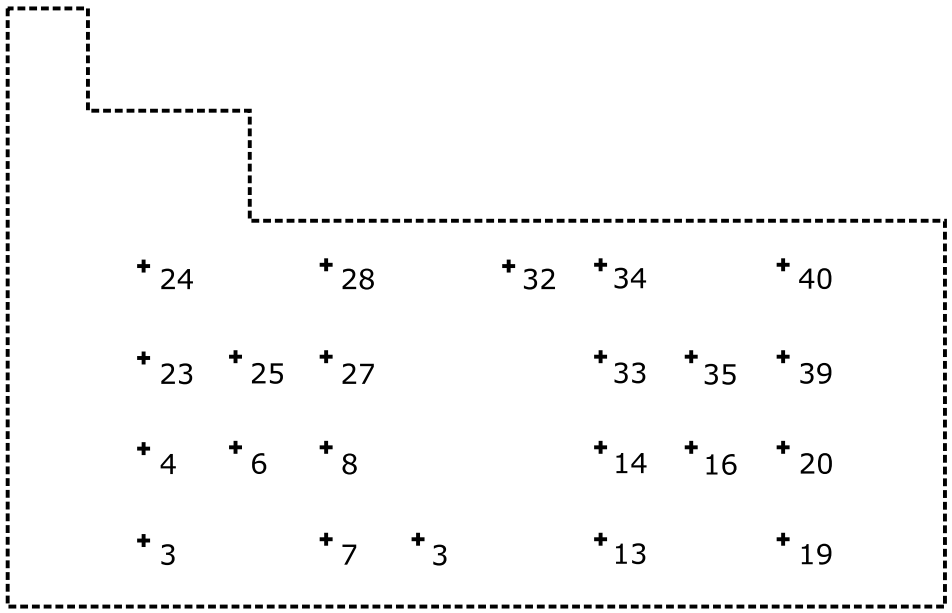

Figure 3.4.: FIT/CorteXlab experimentation room plan and node locations

the dimensions of the room) showed that the channels are essentially flat fading, with very little delay spread. However, even if the FIT/CorteXlab experimentation room is partially covered with electromagnetic (EM) absorbing foam (roof and walls), reflections off the floor and metallic structures occur, creating multipath but with limited delay.

System architecture

In each data collection experiment, 22 or 23 devices are involved: one *scheduler*, one *identification receiver* and 20 or 21 *nodes*, as depicted in Figure 3.5.

The scheduler, as the name implies, orchestrates the transmission of packets over all transmitting nodes, guaranteeing an interference free scenario. It sends a trigger signal to a specific node that initiates a packet transmission. While the scheduler is assigned to a SDR node, it does not need a radio transmitter. Wired Ethernet connections mentioned in the previous chapter are used to trigger the transmitting nodes. The scheduler sends a trigger signal every millisecond to a randomly chosen node via UDP, which ensures that transmissions are made temporally close to each other. This is essential for dynamic channels settings. As such, packets from a single node are spread over the duration of the experiment, and one specific realisation of the channel sees transmissions from more than one node.

The nodes' USRPs are set to "burst mode", to largely reduce the possibility of the oscillator leakage noticed in [74]. This means that the amplifier of an USRP is turned off when not transmitting. Consequently, the radio frequency (RF) circuitry needs time to wake up and stabilise before transmission. Hence, a frame is prepended with 3000 zeros, corresponding to a delay of 0.6 ms.

On the identification receiver side, the USRP remains in listening mode for the duration of the experiment. In order to properly separate the packets from noise and accurately label the received packets' transmitter, a robust detection mechanism is required. This detection mechanism is based on a time synchronisation scheme coupled with an identification header

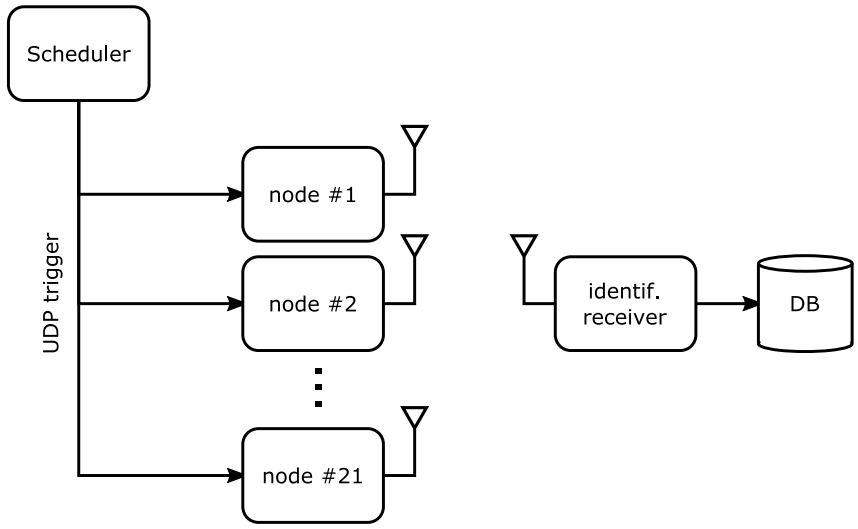

Figure 3.5.: Overall data collection topology (MonoRX)

implemented as described next.

At the nodes, the packets to be transmitted are encapsulated into a *carrier frame* with the following elements:

- A Zadoff-Chu sequence preamble for frame detection and time synchronisation;
- An orthogonal frequency-division multiplexing (OFDM) frame header created using the standard GNU Radio OFDM blocks and containing the node index for data labelling;
- A user-defined payload made from a known quadrature phase-shift keying (QPSK) modulated sequence, random modulated bits or uniform noise, as detailed in Subsection 3.3.2. This part of the overall frame never contains transmitter specific information.

A guard sequence longer than the delay spread of the channel is added between these parts to prevent interference. The overall transmitter GNU Radio scheme is presented in Figure 3.6.

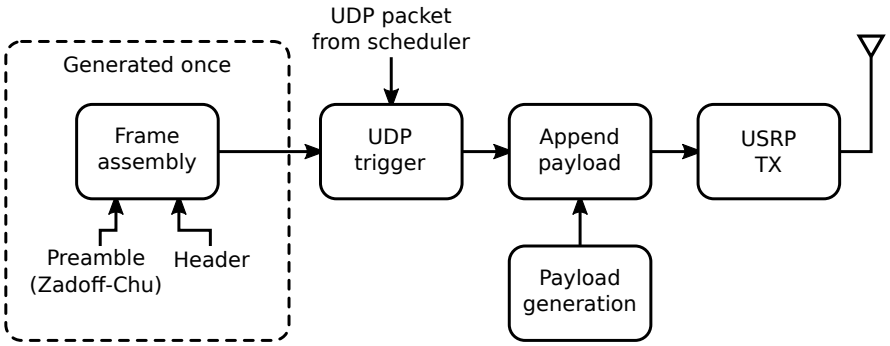

Figure 3.6.: Simplified transmitter flowgraph

At the receiver side, frame detection and time synchronisation are done with a correlator that sweeps the signal looking for the Zadoff-Chu sequence. The next 1040 samples, corresponding to the header, payload, and guard intervals are then forwarded to the header and payload extraction blocks. A standard OFDM receiver then decodes the header and forwards it to a block that uses the transmitter index to send the payload to the corresponding file. The overall receiver chain is presented in Figure 3.7. Finally, the recorded signal, henceforth denoted *example*, is 600 complex samples long, larger than the payload, whose size is 560 complex samples. This oversized cut is exploited to record a small amount of background noise before and after the payload, to ensure recording the start and end of the payload. The authors in [100] suggest, as a rule-of-thumb, to have a number of examples at least 10000 to 30000 times the number of transmitters, in a simple scenario without trying to reduce channel effects. So a standard size dataset is comprised of about 50000 recorded examples for each of the 21 nodes.

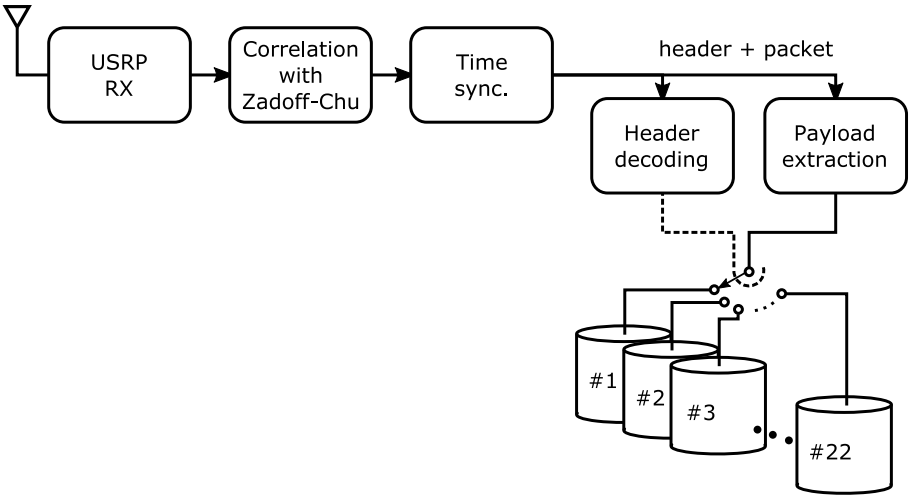

Figure 3.7.: Reception flowgraph.

The receiver is also designed to be able to send received packets directly to a server via UDP to allow for inference in real time. This feature has been implemented and tested successfully but not used extensively in this thesis.

3.3.2. Dataset scenarios

Each experiment is characterised by a tuple of parameters: the payload type, the transmission setup and the number of identification receivers. Such a tuple is referred as a *Scenario* in the rest of this chapter.

Payload

Three types of payload ($x(t)$ in the system model) are studied:

- *Static*: A predefined sequence of bits common to all nodes, modulated in QPSK as defined as the 802.15.4 protocol preamble. A fixed sequence reduces the variability in the examples the CNN trains on, thus reducing the difficulty of its task, while remaining

a realistic setup: in such a case, the frame preamble would be used, as it needs to be transmitted to allow signal detection, regardless of the presence or absence of a header.

- *Random*: Bits from a random source modulated in QPSK. This represents the case where the identification is made using the frame section that contains user data.
- *Noise*: A complex random uniform sample sequence as a worst case scenario. In this case, no pattern coming from a modulation scheme can be used as reference to help measure impairments. It is used as a lower bound on what is achievable by the network architecture regardless of the actual type of modulation used.

Transmission setup

As explained previously, the propagation channel inside FIT/CorteXlab is static and the various devices are fixed on the ceiling and distant from each other, so the channel is quite different from one node to the next. Yet the goal is to be able to identify nodes based on their hardware characteristics and not the channel effects. To remedy this, three transmission setups are defined:

- *Plain*: The basic case where nothing is done to mitigate the channel biases. Everything is static, the experimentation room and transmission parameters, thus the amount of variability in the system is reduced and the identification task is made easier. This mode serves mostly as a benchmark for the two other modes and measures the tradeoff between scenario complexity and learning ability.
- *Varying*: In this case, the amplitude of the payloads to be transmitted by the nodes is scaled by a factor that changes over time before emission by the USRP. This method is preferred over changing gain values for the amplifier in the USRP because it eliminates the need to wait for amplifier stabilisation. It allows the emulation of path loss variations for every node without having to physically move anything in the experimentation room.
- *Robot*: The robot described previously is introduced and set to randomly move around (limited to the right half of the room due to temporary issues). For reference, the robot is moving at a maximum speed of 1 m/s, at the sample rate of 5 Ms/s, the transmission of the 600 samples long payload occurs in 120 µs, with a maximum robot movement of 120 µm, less than 0.018 % of the wavelength, so the channel is static in this time frame. But over the 45 min dataset gathering experiment, the robot is able to travel hundreds of meters. On top of that, the robot is not a cylinder, with many surface features, so its rotation also greatly impact the channel. This mode also includes amplitude variation and is the most complex and realistic of the scenarios proposed but is also the most time demanding to run, and so is only used with the static payload.

Transmission power was measured with a cabled spectrum analyser to be at -9.5 dBm for the plain scenario, and between -9.5 dBm and -42.5 dBm for the other two. The highest transmission power leads to a SNR of 28.7 dB for the node closest to the receiver, down to 6 dB fo the furthest away, equally for all transmission scenarios. The lowest power leads to SNR reaching 0 for all distances. This measurement is taken using the recorded examples in the datasets. Since the examples comprise 600 recorded samples, and the true payloads

are 560 sample, part of the recorded samples are noise, so the SNR level is measured by comparing the power of this part of the example, to the power of the payload portion. This experimental method means that SNR levels below 0 cannot be accurately measured.

Number of receivers

In order to increase even more the channel variations and to reduce the possibility for the receiver to learn from the channel properties, the *MultiRx* setup is proposed where we merge the signals observed from several devices acting alternatively as identification receivers. This *MultiRx* setup, in contrast to the *MonoRx* setup, allows the collection of payloads from 20 different receivers for each transmitting node. It is done by successively collecting a set of 21 MonoRx experiments with $N = 20$ and $M = 1$, each with a different single receiver, and combining all the recorded signals into one dataset with $N = 21$ and $M = 21$. That means that at a given time, one node has only one receiver, but over time, all 21 devices will have been listened to by all the other 20. The neural network defined in the next subsection always uses only one payload, from one receiver whose Id is not given, and it does not know what mode was used.

Practically speaking, another difference between the two modes is the fact that the node number 3 is not used in MultiRx. FIT/CorteXlab contains 22 USRPs devices, so, in MonoRx, 21 act as transmitting nodes while one preselected USRP acts as the receiver. But in MultiRx, the roles are permuted and so the 22 devices act alternatively as transmitters or receiver. However, to facilitate the comparisons between the two modes, the same number of transmitters is used, namely 21.

Table 3.2.: Considered payload characteristics

Parameter	**Value**
N	21
M	1 (*MonoRX*), 21 (*MultiRX*)
Frequency band	ISM 433 MHz
Signal bandwidth	2.5 MHz
Waveforms	QPSK (RRC filtering), Uniform
Samples/symbols (QPSK)	2
Total samples	560

The datasets collected and used in this paper are available online along with the scripts and GNU Radio flowgraphs to generate them [102] in the hope that they can be reused by the community.

3.3.3. Learning architecture

Network architecture

As shown in Fig. 3.9, the neural network used is based on a CNN architecture: five 2D convolution layers with a max pooling layer between each of them, a flattening step and six fully connected layers before the final softmax output. The chosen method for handling the complex-valued input samples is to represent them by their Cartesian coordinates, treating the real and imaginary parts as independent input dimensions. The vector of 600 complex

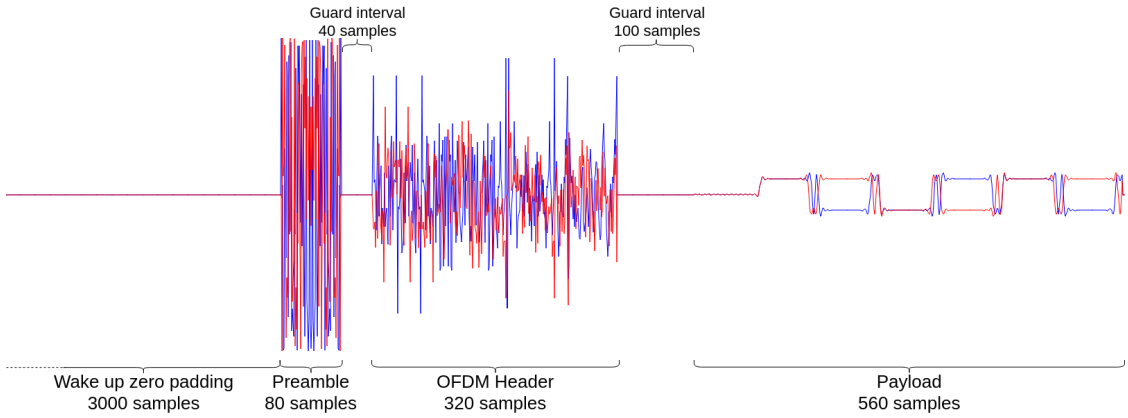

Figure 3.8.: Frame samples sent to USRP for emission, with zeroes for amplifier wake up, preamble, header, and payload with guard intervals.

values becomes a matrix of 600×2 real numbers used as a 2D image by the convolution layers. The network outputs a probability vector over the 21 possible transmitters.

Training phase

Each dataset is randomly shuffled before training and split in a standard 70/10/20 distribution for training, validation and testing. These will be called *Training slice*, *Validation slice*, and *Test slice* in this chapter instead of the usual term *set* to reduce confusion with dataset and set of datasets. Each scenario is used to train a different instance of the architecture presented above. Training examples are presented in mini-batches of 128 examples over 30 epochs for the MonoRx scenarios and 100 epochs for the MultiRx ones. The Adam optimiser is used with a learning rate of 0.001, tasked with minimising a categorical crossentropy loss between labels and predictions.

Hyperparameter tuning was done using a hard to learn dataset: MultiRx setup with amplitude variation and random payload. For this, a ten epoch long training was done for various parameters (learning rate, batch-sizes, layers) and the best performing hyperparameter set was retained.

3.4. Results

3.4.1. Payload type

In this section, the impact of the payload type on the classification performance is assessed experimentally. We perform a series of tests corresponding to different scenarios as described in Section 3.3. The three kind of payloads (static, random, noise) are tested, combined with the two channel conditions (plain or variable) and with the MonoRx or MultiRx configuration. The results are presented in Fig. 3.10. These graphs provide the identification accuracy over the test slice of the training dataset for each combination (payload,channel,receiver). Clearly the learning is more accurate when the payload is static (left figure). This result indicates that such identification task on ambient signals,would be more efficient if performed on a fix

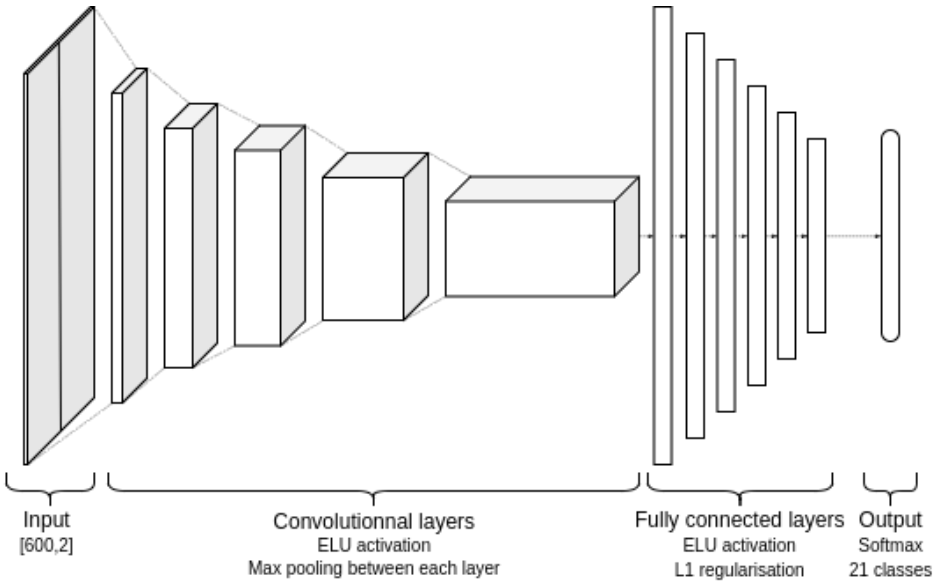

Figure 3.9.: Neural network architecture

preamble or header, rather than on the data payload.

When the channel variability is increased by using varying channel gains (var) and MultiRx, a significant accuracy loss is observed (last column in left figure), but the accuracy remains high at 86%.

With the two other payloads (middle and right figures), the learning capability reduces drastically when channel varying and MultiRx setups are used (last column, with 38%). Clearly, the high accuracy observed with plain and MonoRx setup with these payloads may be achieved thanks to the channel signature rather than the radio node itself, explaining why when MultiRx and variable channel are used, the accuracy collapses (it still remains an order of magnitude above random guess). It is worth mentioning that the scenarios with the Random and Noise payload need a dataset twice as large to avoid overfitting on the training slice. This behaviour is not surprising since these scenarios combine the most variability in payload, channel and also receiver impairments.

For all these reasons, in the rest of this work, a Static payload will be used.

3.4.2. Impact of the transmission setup

In this section we assess the impact of the transmission setup on the identification capability.

According to the results of the former section, the payload is static in these experiments. The identification performance over three scenarios, corresponding to three channel conditions referred as Plain, Varying or Robot, are evaluated. Note that in the scenario labelled Robot, the Varying transmission setup is also used. Three learning dataset are thus built, one for each of the three scenarios, and used to train three networks. Then, the three networks are used independently on new datasets and the identification accuracy is presented in Figure 3.11.

The left Figure presents the results for the MonoRx scenario. The blue bars correspond to the accuracy obtained on the test set Plain, where the networks learned respectively on the three learning sets: Plain, Varying and Robot. The identification accuracy is high in all cases.

Table 3.3.: Detailed network layers description

Layer	Size	Kernel	Activation
Conv_0	8	[6, 2]	Elu
MaxPool_0		[2, 1]	
Conv_1	16	[4, 1]	Elu
MaxPool_1		[2, 1]	
Conv_2	32	[4, 1]	Elu
MaxPool_2		[2, 1]	
Conv_3	64	[4, 1]	Elu
MaxPool_3		[2, 1]	
Conv_4	128	[4, 1]	Elu
MaxPool_4		[2, 1]	
Flatten	1920		
Dense_0	512		Elu
Dense_1	256		Elu
Dense_2	128		Elu
Dense_3	64		Elu
Dense_4	32		Elu
Dense_5	16		Elu
Output	21		Softmax

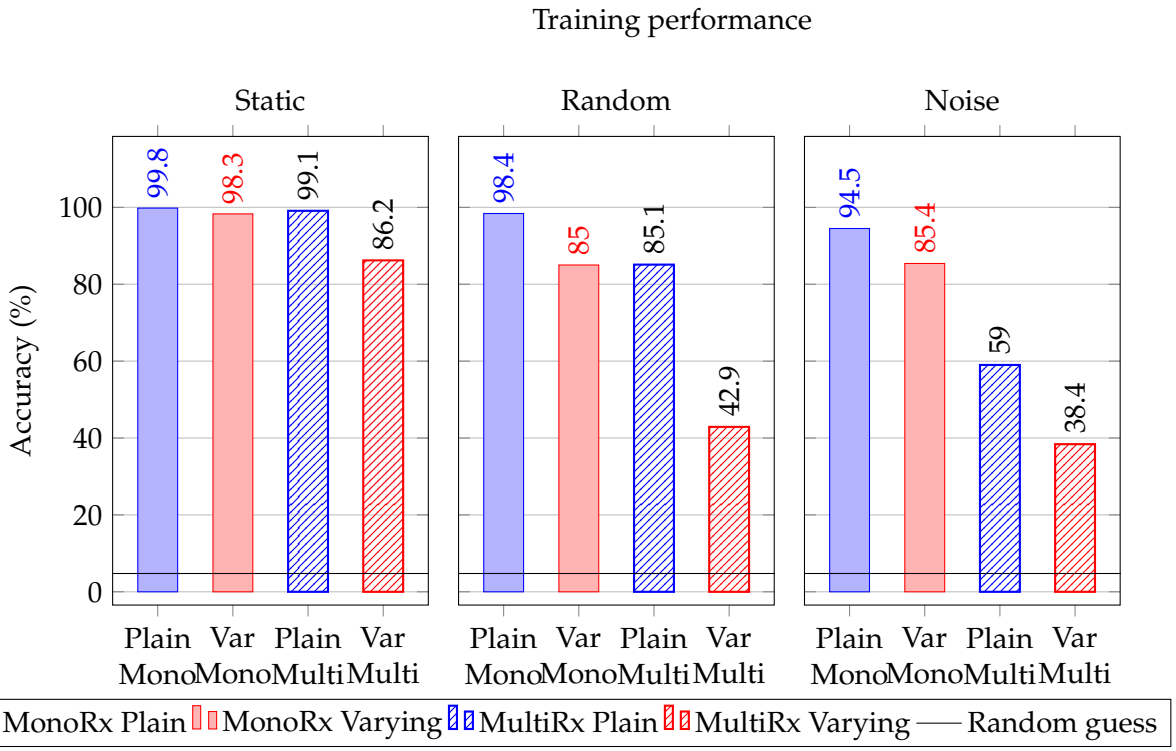

Figure 3.10.: Accuracy reached by networks trained on plain or varying amplitude scenarios and the 3 signal types. Accuracy is measured on the test set from the training dataset.

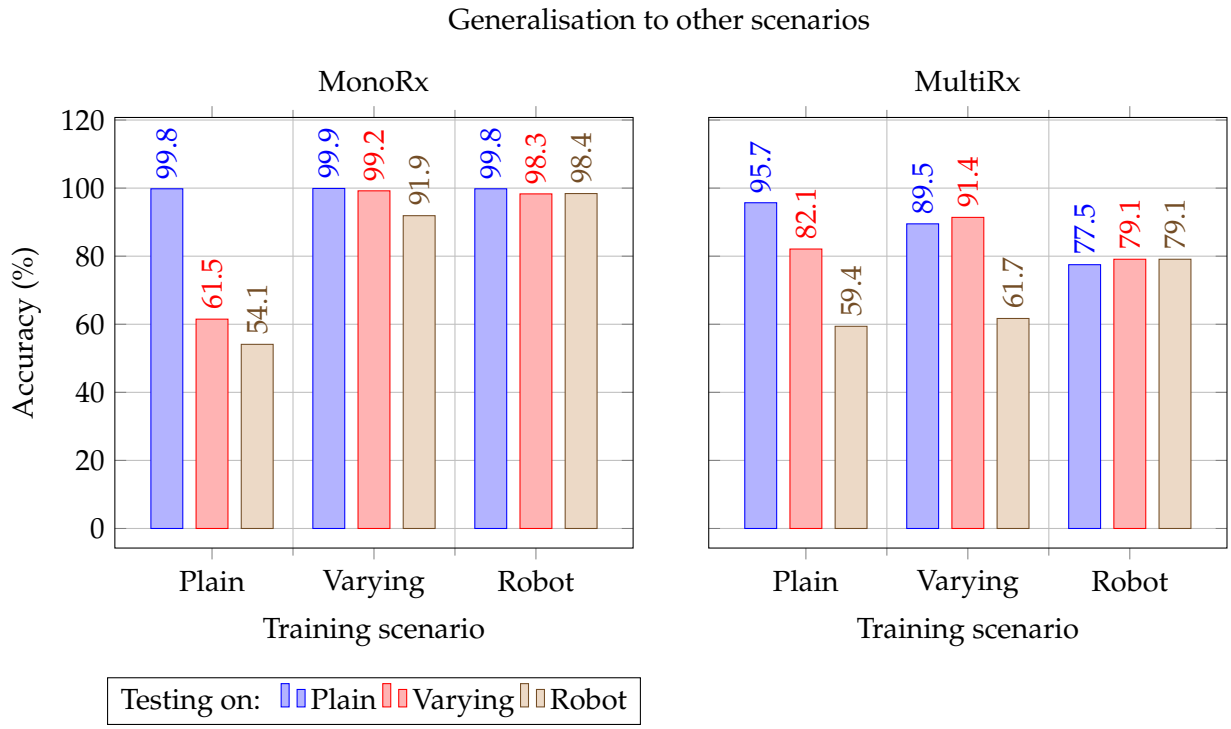

Figure 3.11.: Accuracy of the networks trained on three different scenarios (x axis legend) and tested on other datasets of the three kinds (resp Plain, Varying, Robot) as indicated by the bars' colors. In these experiments, the payload of all packets was static and the environment in the shielded room remained unchanged.

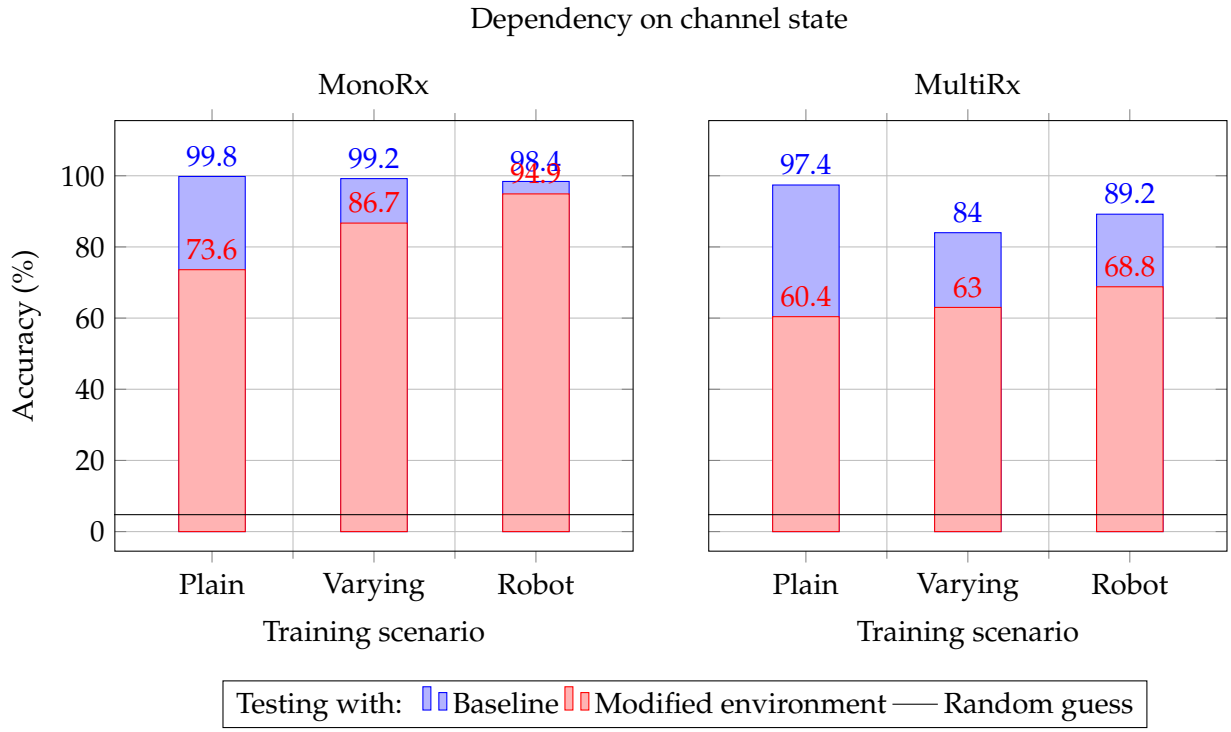

Figure 3.12.: Accuracy of networks trained on one scenario with static payloads and tested, either on data from the training dataset or on a dataset with the same scenario but with a modified environment.

On the opposite, the network trained in Plain conditions is not able to efficiently identify the transmitters when used on the other scenarios (only 61.5% and 54.1% are obtained). It is likely because when the network is learned on Plain conditions the static channels contribute to the learning and the network is not able to focus on the radio properties themselves.

As expected however, the networks trained on more complex scenarios, is more robust and is efficient on less complex scenarios. Typically, the network trained on the Varying dataset performs as well on the Plain and the Varying dataset. And the network trained on the Robot dataset performs equivalently on the three test datasets.

On the right Figure, the same behaviour are observed with MultiRx scenarios. But in addition, with the Robot, an accuracy loss is observed. When the signals are perturbed simultaneously by the channel gains, the robot and the receiver position, the learning conditions are the hardest ones. The fact that it is still possible to learn on these signals with an accuracy of 79% is quite encouraging.

3.4.3. Channel variations

All the previous results were obtained in FIT/CorteXlab under fixed conditions. As the human access to the shielded room is controlled, the environment was guaranteed unchanged during the experimentation.

Now, to evaluate the NN sensitivity to the environment, we introduce new test datasets. The environment in the shielded room is modified by adding a metallic chair in the room thus

creating additional propagation paths. The formerly trained networks are then evaluated on these new datasets.

Fig. 3.12 presents the corresponding results where the accuracy obtained from the datasets before the environment perturbation are given in blue ( ) and the accuracy obtained on the datasets obtained after the perturbation are given in red ( ) for respectively MonoRx and MultiRx conditions.

Clearly, these results show the sensitivity of the network to the environment, especially when it learned on the Plain scenario. This kind of loss in accuracy has been already reported by other authors in [93], where they introduced artificial impairments to cope with it.

In our work, especially in MonoRx, we see how learning in more complete conditions (i.e. Robot scenario) allows to reduce significantly the accuracy loss. This is fully true in the MonoRx scenario, where the network learned with the Robot scenario is almost not sensitive to the environment perturbation. However, in MultiRx, the loss is reduced but remains present. Note that the 89.2% accuracy with the Robot-Robot test was obtained when the same dataset was split and used to learn and test, while the former section (see Fig. 3.11), the Robot-Robot test was using two different datasets but without deliberate environment perturbation. Unfortunately, the gathering of a Robot dataset currently requires human intervention to setup the robot and this can lead to small involuntary channel perturbations. This means that the robot results from the previous section cannot be directly compared to the training results of Fig. 3.12. Therefore, in the result herein obtained we may assume that some information related to the channel is still used by the network to learn. However, we believe that the remaining accuracy clearly indicates that a part of the learning is performed on the RF signature and not on the channel conditions.

This is an encouraging result as it motivates the development of identification techniques based on RF signatures, and the dataset built in this work is unique in the sense that channel conditions are carefully controlled.

3.5. Encountering an unexpected bias: the synchronisation (a case study)

This section delves into an event that marked the transmitter identification dataset gathering process: a synchronisation parameter unexpectedly impacted the generalisation capabilities of the trained NN. This section does not present a real scientific breakthrough since the cause of the issue turned out to actually be quite logical and straightforward, but it can give important lessons regarding a dataset gathering process.

The story of this event will be written chronologically to show the decision making process that led to the issue and, subsequently, to its resolution.

3.5.1. The issue

The work presented in this chapter started with only the "MonoRX" experiments, using always the same node disposition inside of the FIT/CorteXlab room for reproducibility and fair comparison between different datasets. The receiver being always the node number 6, as displayed in the floor plan of Fig. 3.4.

Shortly before the design and implementation of the "MultiRX" experiments, a new feature was added to the experimentation room: a synchronisation system based on four octo-

clocks [103] in a tree structure, allowing for full oscillator and sampling frequency synchronisation between all the nodes in the room. That feature was enabled right away, thinking "More synchronisation is always good, most theoretical works rely on it for increased performance". And continued experimentation on the "MonoRX" setting showed no change in performance whatsoever, comforting us in the belief that synchronisation can indeed never hurt.

Table 3.4.: Accuracy with synchronisation enabled

Scenario	Plain			Varying		
	Static	Random	Noise	Static	Random	Noise
Test Accuracy	99.98%	94.32%	99.52%	84.34%	94.68%	80.32%
Cross test	39.09%	31.09%	36.28%	43.66%	72.06%	57.27%

Then came the "MultiRX" scenario, which, as described in the above sections, involves more than one node acting as the receiver. At that point, the trained networks could perform well when testing against the test slice of the training dataset (the examples of which were not used at training but were part of the same distribution as the training examples), but displayed an important drop in accuracy when cross-testing them against the test slice of another dataset of the same scenario, as shown in Table 3.4. The magnitude of the drop changes with the type of scenario involved, and the accuracy values themselves are different from the ones presented in the above section since the results come from datasets gathered after the issue presented here was solved, but there is at least 30 to 40 percentage points less in cross testing.

3.5.2. Search for the cause

Save for a coding error in the neural network implementation, a low performance on test data can have two causes: either the network is not able to train and converge properly, and this is not the case here, since same-dataset testing leads to high accuracy, or the underlying feature distribution in the examples is different between the training and test datasets. To eliminate possible human physical intervention and reduce potential temporal evolution in the hardware, new pairs of datasets were gathered with no time interval, monitoring that no human enters the experimentation room. With the absence of movement inside the room, both during and between experiment, and its isolated property, channel modifications were eliminated as a possible cause. In parallel, the additional datasets were used to train the neural network on more, and more diverse, data. Training on a dataset built from the combination of three experiments (that do show the low cross-test accuracy), display the same behaviour when tested on a fourth. The MonoRX scenario did not show this behaviour, until changing the receiving node. Only six nodes, when used as a receiver, allowed cross-test accuracy to remain at the level of same-set accuracy. This experiment points towards an adverse hardware effect in the USRP devices, and swapping the receiving USRP for a newer, less used one alleviates the issue, hinting at a phenomenon caused by age. The number of receiving and transmitting devices was progressively reduced in an attempt to find the tipping point between the working MonoRX and the MultiRX, with simpler and simple neural networks. This culminated in a dataset with only one transmitter and receiver, and a one neuron neural "network" able to distinguish between two datasets with examples of only 15 IQ samples instead of the original 600, while manual examination of said examples did not show significant difference.

Table 3.5.: Accuracy of networks trained on one synchronisation possibility and tested on the others

Synced radio	Tx	Both	Rx	None
Tx	81.2%	34.5%	19.2%	38.3%
Both	54.4%	45.6%	38.2%	38.2%
Rx	21.1%	28.5%	80.2%	22.3%
None	41.3%	43.8%	34.6%	81.1%

The remaining hypothesis was the device synchronisation, so new datasets were gathered with the octoclock unused, leading to the resolution of the issue. The next sections delve into the explanation of the phenomenon in two successive components: at sample level, where packet start can be detected by the software an integer number of samples early or late, and sub-sample level where the hardware sampling time of the receiver doesn't match the one of the transmitting node. The fact that several device did not show the adverse cross-test behaviour was due to an erroneous hardware setup that prevented the USRP from using the external synchronisation source, and in one instance, to a faulty synchronisation cable.

3.5.3. Sample level synchronisation

For both MonoRx and MultiRx, a standard Static Varying Scenario is gathered with no timing offset, and a network is trained on it. Then, a series of small datasets of the same scenario with various offsets is created. This is done by explicitly telling the payload extractor to record early or late.

Fig. 3.13 presents the effects of these synchronisation errors on a NN not trained to cope with them. Both settings clearly show a sharp decrease in accuracy. For MultiRx scenario, the accuracy level drops out even for an offset of 1 sample, while for MonoRx, the accuracy remains significant up to an offset of 3 samples. This effect may impact the performance in low SNR conditions, where sample synchronisation errors are more frequent.

3.5.4. Sub-sample synchronisation

Sub-sample synchronisation is a hardware effect so it needs to be simulated to allow study. A MonoRx Static Varying Scenario is gathered, with a payload oversampled by a factor of 16. To stay inside the signal processing capabilities of the hardware used, this is done by reducing the baud rate instead of increasing sample rate. Before feeding it to the neural network, this signal is separated into 16 undersampled time series with one sample taken every 16 and a starting offset between 0 and 15. Then, a network is trained on the undersampled series with an offset equal to 8 and tested on all the other offsets for four different datasets of the same scenario.

The decrease in classification accuracy with sampling time offset seen in Fig. 3.14 shows that a synchronised "naive" system is indeed unable to cope with sampling time variations. The small accuracy difference on offset 8 between the training dataset (2) and the others can be attributed to an additional, non artificial, offset smaller than the sampling time.

Table 3.5 compares four configurations:

- *Tx synch:* All the nodes are synchronised but not with the receiver.

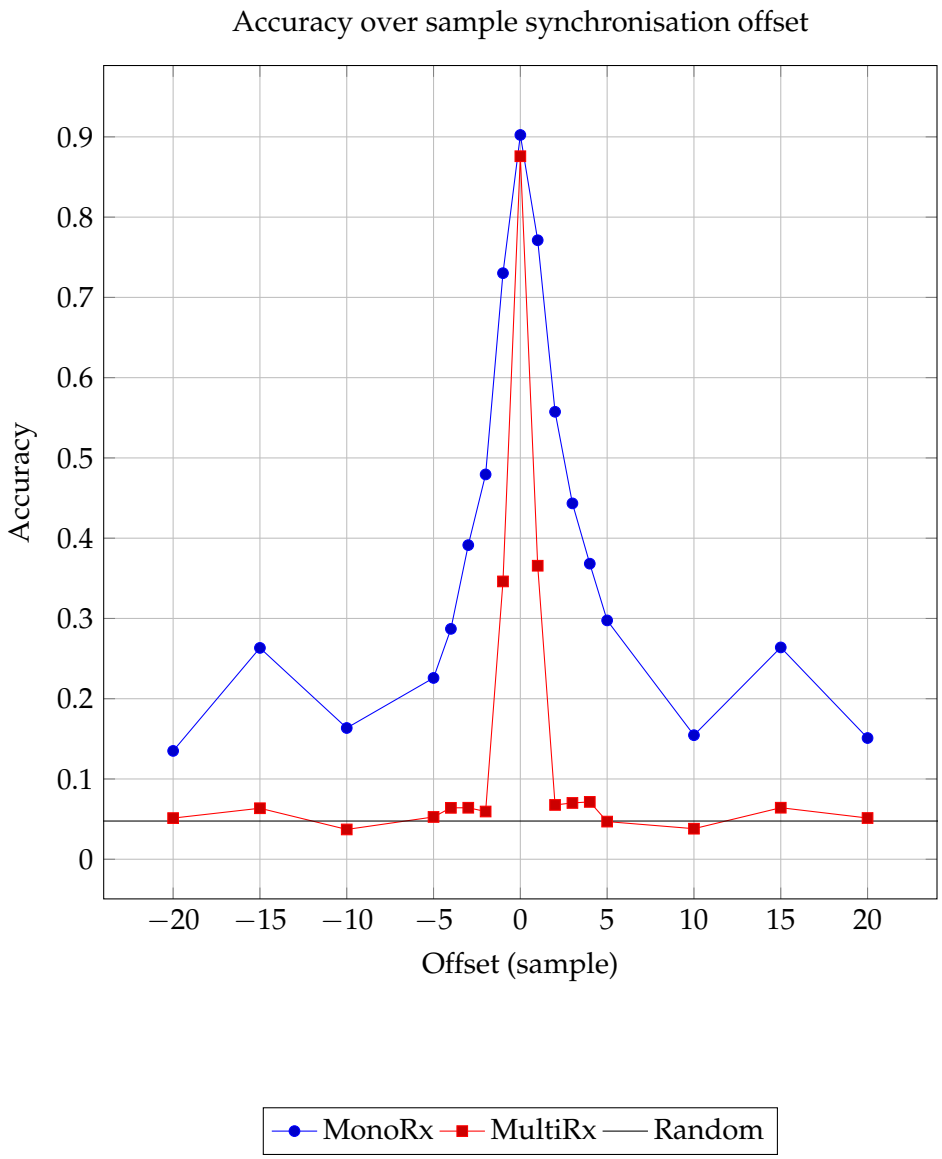

Figure 3.13.: Accuracy attained when the frame detector produces a timing offset at test time

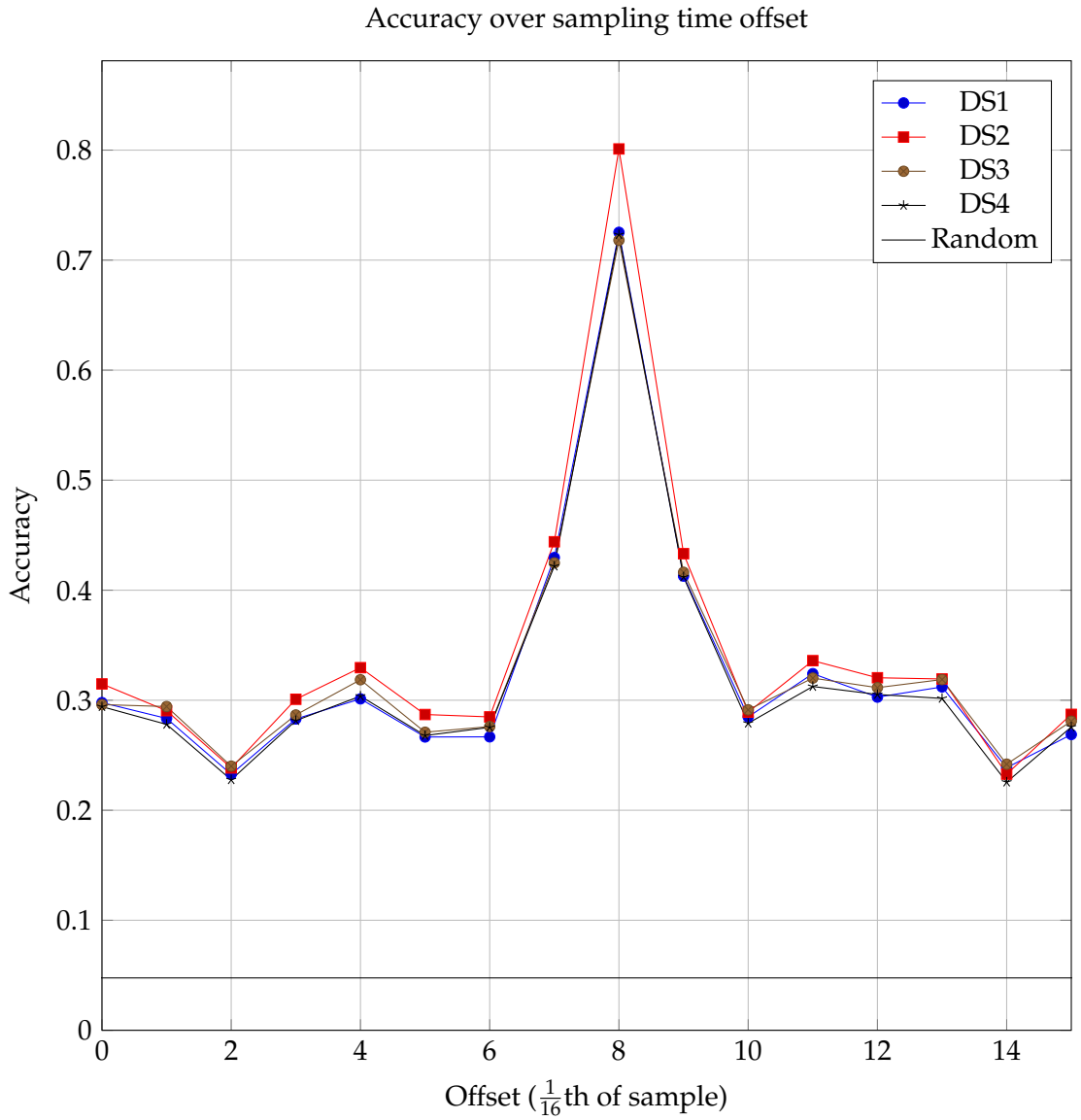

Figure 3.14.: Accuracy of a network trained on dataset 2 with an offset of eight and tested over the four datasets and 16 possible offsets

- *Tx and Rx synch:* Both nodes and receiver synchronised. This was shown to perform badly.
- *Rx synch:* Only the receiver has access to the master clock. This mode only provides to the receiver a very stable clock.
- *No synch:* Nothing is synchronised. This is the most realistic scenario for a real world deployment.

For this table, a first set of datasets was generated and used to train four networks, one for each configuration. Each NN was tested on a second set of four datasets. So the matrix diagonal shows the generalisation capability of the NN to another dataset of the same kind (but not the same dataset).

The output of this section is that when all nodes and Rx are synchronised, the NN cannot generalise. The reason follows: when the nodes are synchronised to the Rx, a timing offset exists that is constant over a dataset. But this static timing offset changes by a random amount when a new acquisition starts. The NN trained on a dataset with a fixed sampling offset is not able to adapt to a different one. On the other hand, as long as there is no synchronisation between nodes and receiver, the network can generalise since there, the offsets change even in the training data, and the network can learn to cope with it. A common trend in the community when tackling a new communication problem or approach is to first design algorithms or systems that assume simpler operating conditions, such as perfect synchronisation, to serve as a stepping stone before extending it to more challenging conditions. It appears from this section that, in this case, the stepping stone can be detrimental to the system performance instead of providing a progressive exploration of the system's capabilities.

3.6. Conclusion

3.6.1. Conclusion

This work explored a range of signal parameters that can impact the ability of a neural network to identify transmitters based on their physical layer characteristics and to generalise to more realistic environment, such as changing channel characteristics. The resulting guidelines to get a good transmitter identification is as follows: The part of a transmitted packet used for identification should be as much deterministic as possible, the physical preamble being a prime candidate. The channel encountered in the training dataset should have as much randomness as possible to force the network to learn to cope with a more realistic situation, and finally, it is not necessary to perfectly synchronise transmitters and receiver since this will only result in a poor generalisation performance.

The reader is invited to reproduce and extend these result by reusing the generated datasets and creating new ones in the FIT/CorteXlab or elsewhere using the data collection process [102] developed for this work and available online alongside the datasets.

3.6.2. Future works

To envision an industry grade implementation, several additional aspects still need to be studied:

QPSK modulation was chosen in this work as a realistic modulation example for the envisioned IoT usage of this technology. Perhaps other modulation schemes allow a more effective classification by exposing more dependable patterns or forcing amplifiers into less linear parts of their operating range such as a high peak-to-average power ratio (PAPR) OFDM setup. A study of what makes the best modulation scheme for this task could help improve the community's understanding on the matter.

Similarly, the bit sequence for the Static payload was chosen as a realistic example of an IoT type preamble bit sequence. Custom preambles could be crafted to simultaneously optimise classification accuracy and generalisation and frame detection and synchronisation performance.

The influence of the example size in term of number of samples has seen some attention in [74], but, apart from that, all the works in the literature use a different sample amount in examples. A more detailed study of this factor could help in comparing the existing works and provide guidelines and trade-offs for future implementations.

The current system setup is relevant in cases where all the users are known in advance, but it cannot handle the arrival of a new node, or the use of samples from new receivers. Architectures able either to detect intruders or erroneous users, or to add new users to the pool of known ones would be more versatile and allow for more dynamic use cases. A system able to agnostically use samples from any new receiver would greatly improve on the scalability and ease of field implementation.

The present chapter purposely avoids reliance on channel effects to allow identification regardless of position, movements and environment changes. But these can be of use in the related task of location verification [104], and could be coupled with authentication in slow varying environments or very short-term identity verification.

Finally, one of the main promises of this technology is an increase in security from a reduced ability for attackers to spoof the identity of legitimate users. However, neural networks have been shown to easily suffer from adversarial examples, both in general [105], and on the specific topic of wireless communications [106]. So, to allow real implementation, any candidate architecture needs to show some adversarial robustness both in simulation and in the field.

— 4 —

Active-User Detection

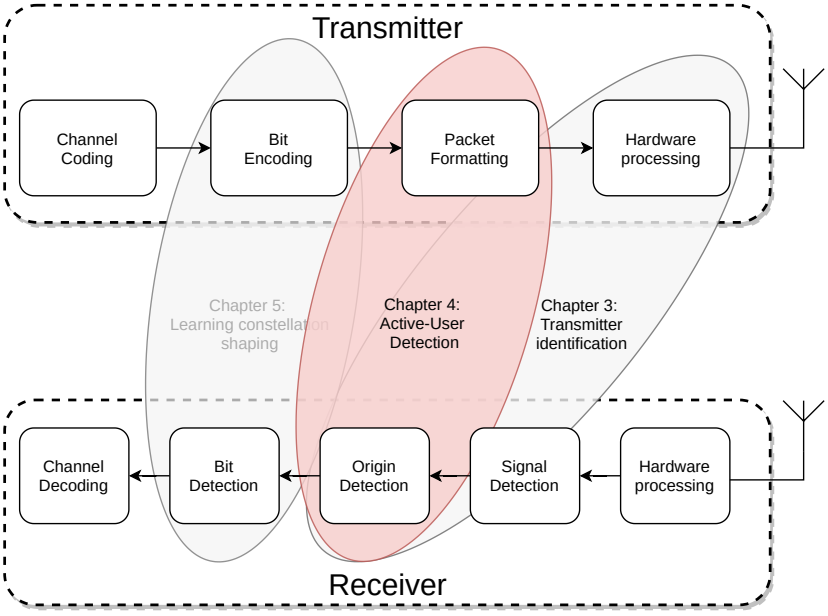

Figure 4.1.: Position of the chapter in the thesis outline

THIS chapter extends the origin detection task of the previous chapter from a unique transmitter at a time to multiple simultaneous active users. In that case, hardware fingerprinting stops being a good source for detection, and dedicated codewords are required to be added in the transmitted packets to better handle interference. The work presented in this chapter was done in collaboration with Diane Duchemin in the context of her own PhD.This collaboration joined her work on the MAP and It-MAP detectors presented in Sec.4.2 with the neural network based detectors presented throughout this thesis. It also lead to the publication of a conference article[12]

4.1. Introduction

Massive access in IoT-dedicated radio networks, especially in 5G, brings several challenges. In this setting, a huge number of sensor nodes is to be sporadically served within the specific constraints of machine type communication (MTC). These networks will be implemented mainly with low cost devices, thus having restricted radio functionalities as well as scarce power and computational resources. Besides, latency, spectrum and energy, must be held to the same efficiency demands of current communication standards, sometimes even higher, to fulfil the requirements of the foreseen tactile internet [107]. Uplink data transmissions from simplified sensor nodes aim at minimising transmission duration as well as the amount of transmissions, given their finite power resources, but also the small amount of data to be transmitted. A high signalling overhead would drastically reduce the operational life-time of such devices as they would spend more time and energy to transmit protocol-related messages than useful payload data. Unlike current 4G based access procedures, as planned in narrowband IoT (NB-IoT) (even though new releases provide a shorter access procedure [108]), an "all in one" grant-free uplink message encapsulating access request, device identifier, and data, would be ideal.

To achieve this grant-free uplink reality, the main challenge is the detection of the active subset of sensor nodes by the base station (BS), also referred to as active user detection (AUD). To enable the transmission of users' identities despite a high network density, the usage of a dedicated spectrum sharing technique is required that must be compatible with the rapidly evolving traffic load within a high number of potential users. NOMA [109], and in particular code-domain NOMA, is a good candidate [110] for such a spectrum sharing technique as it limits collisions for simultaneous transmissions without requiring to use an extremely long access sequence for each user, given the network density.

As a result, all the complexity of the AUD task is pushed to the BS. Avoiding a handshake procedure requires efficient detection algorithms to retrieve active users' identity from a "one shot" access message with limited channel state information at the receiver (CSIR). The optimal AUD as described in [111] suffers from high complexity which does not seem compatible with real time implementation. An iterative version of the optimal detector, having a lower complexity -but also lower performance-, is also introduced therein. Developing a high performance though low complexity detector is crucial for a realistic and efficient AUD implementation. Most efficient algorithms proposed in the literature to cope with this problem exploit either a Bayesian estimation formalism or the compressive sensing formulation [112], [113]. Both have many similarities but lead to different iterative algorithms. Despite their efficiency, none of these algorithms can guarantee to achieve the optimal solution as they have to trade their accuracy with complexity. Therefore, the competition is still open.

With the recent and growing interest of the community toward machine learning, and particularly DL, it has been shown that its usage can help to solve complex problems, mainly when defining good models is difficult, or when the models exist but provide solutions too complex for their exploitation. The scenario presented here falls into the second category, and appear to be a good candidate to exploit DL. The objective of this chapter is to design a DL receiver for massive NOMA, and more specifically, the AUD in non-coherent channels. Related studies have been done around this subject, for instance in [107], [114]. These works are focused on the resource optimisation problem for code domain NOMA and employ auto-encoder based solutions in both cases. They show that the encoding and decoding performance can be improved through end-to-end optimisation. The metrics used there are

symbol error rate (SER), sum rate and convergence rate. In [115], the question of imperfect CSIR is addressed for power domain NOMA. This paper is also addresses resource allocation optimisation.The authors of [116], while also dealing with power domain NOMA, focus on channel estimation and signal detection in the context of OFDM. They propose a comparison with a successive interference cancellation (SIC) based algorithm and show the interest of the DL approach. The model is nevertheless restricted to two users and the channel realisation is fixed in training and testing phases. A preamble and collision detection scheme based on DL is proposed in [117], where the study is performed on pre-processed long term evolution (LTE) random access preamble signals: the correlations with the possible Zadoff-Chu sequences are directly provided to the network. The objective of the authors include the detection of multiple collisions in order to improve contention resolution, and therefore, access probability. Whereas the collision study is realised in a massive access scenario, the detection evaluation is performed with a single user scenario only, by comparing the missed detection performance of the proposed fully connected NN with other more classic preamble detectors. All these works are closely related to the use-case of the present chapter, but none of them directly address the task at hand: to the best of our knowledge, this work is the first to apply DL on non-coherent AUD with code domain NOMA.

The rest of the chapter is organised as follows: section 4.2 presents our model and the reference schemes to which our approach is compared. Section 4.3 provides details on the implementation choice regarding the DL scheme we propose, while section 3.4 is dedicated to the evaluation of the solution. Section 4.5 concludes the chapter.

4.2. System model for the massive random access

4.2.1. Non-coherent AUD

The random access channel in massive MTC is important to guarantee a fair radio medium access. It is herein assumed that the sensor nodes, henceforth referred to simply as *nodes*, receive a random code in advance which is used to send a resource request to the BS they are associated with. In a standard approach, if two nodes request a resource in the same slot, a collision occurs and at least one of the two messages is lost. However, with the knowledge of the codes distributed to the nodes, the BS tries to determine the identity of all the nodes involved in a request. Such an approach, referred as coded random access [113], allows to reduce the number of resources reserved for the random access mechanism and can accelerate the handshake mechanism. Indeed, unlike the 4G access protocol which the NB-IoT is based on and relying on a pool of available Zadoff-Chu access sequences, this approach ensures the uniqueness of the codes employed by the nodes. This fact allows to avoid access code collisions but also additional steps, known as the contention resolution, dedicated to the identification of the users in the handshake procedure. It can also be used as a standalone mechanism in cases where the only one bit is to be transmitted, then the binary value of the nodes' activity suffices without needing further handshake.

In our model, the BS transmits a beacon allowing the nodes to be roughly synchronised and to control their power such that in average the received power at the BS is constant for all nodes. However, the instantaneous channel states are not known, and no pilots are used in this detection phase. The detector thus operates in non-coherent detection mode [111].

We adopt the following notation for the remainder of this work: $(\mathcal{U}, \Phi)$ is a measurable

space where $\mathcal{U}$ denotes the total set of nodes, with cardinality $K{=}|\mathcal{U}|$ and $\Phi{=}\mathcal{P}(\mathcal{U})$ the powerset of $\mathcal{U}$. A node subset is denoted by $\mathcal{A} \in \Phi$. For a given random access slot, we note $\underline{\mathcal{A}} \in \Phi$ a set of *active* nodes. The activity rate is assumed low (less than 0.5) implying a sparse transmission set. We further assume that the node activity follows a Poisson distribution with mean parameter λ (thus the node activity probability is $\theta = \lambda/K$). As stated previously, a unique codebook $\mathcal{C}$ is generated and shared among the network (the transmitters and BS both agree on the codes during an initial association phase). As a result, each node k owns a dedicated complex Gaussian code $\mathbf{c}_k$ of codelength M and unit power.

As mentioned previously, the received messages are considered synchronous and a perfect average power control allows the messages to be received with an average SNR ρ. The BS possesses N antennas while the nodes have a single antenna. Transmissions are subject to a flat Rayleigh block fading channel, modelled as a random vector $\mathbf{h}_k \sim \mathcal{N}_{\mathscr{C}}(0, \mathbf{I}_N)$ of size N where $\mathbf{I}_N$ is the identity matrix of dimension n and $\mathcal{N}_{\mathscr{C}}(0, \cdot)$ indicates a complex standard Gaussian distribution. The receiver noise introduces an additive white Gaussian noise (AWGN), modelled as a random vector $\mathbf{z} \sim \mathcal{N}_{\mathscr{C}}(0, \mathbf{I}_{NM})$ of size NM. It should be noted that neither the BS nor the transmitting nodes are aware of the actual channel realisations, but only know the channel statistics, as described above. For a given active node k, the channel coefficients $h_{m,n}$ are constant with respect to (w.r.t.) to m and are independent and identically distributed (i.i.d.) w.r.t. n. This means that the message is sent over a narrowband channel, typically a single carrier in an OFDM frame, as defined in NB-IoT for MTC. The proposed model is similar to the one used in [111], [112].

Let $\mathbf{y} \in \mathbb{C}^{NM}$ denote the received signal, ρ the targeted SNR and $\otimes$ the Kronecker product. The received signal is then given by:

$$\mathbf{y} = \sum_{k \in \underline{\mathcal{A}}} \sqrt{\rho}(\mathbf{I}_N \otimes \mathbf{c}_k)\mathbf{h}_k + \mathbf{z}. \tag{4.1}$$

The BS performs an AUD given $\mathbf{y}$ and prior knowledge, restricted to the codebook, the activity probability law and the statistical CSIR. The AUD algorithm is performed on a non-coherent channel, since no pilots are used for prior channel estimation. Let $\hat{\mathcal{A}}$ denote the detected active node subset. To evaluate the performance of the algorithm, the following metrics will be used: codeset error rate (CER), user error rate (UER), misdetection rate (MDR) and false alarm rate (FAR), according to the following definitions:

$$\text{MDR}: \quad \bar{\epsilon}_{md}{=}\mathbb{E}_k\left[\mathbb{P}[k \notin \hat{\mathcal{A}} | k \in \underline{\mathcal{A}}]\right] \tag{4.2}$$

$$\text{FAR}: \quad \bar{\epsilon}_{fa}{=}\mathbb{E}_k\left[\mathbb{P}[k \in \hat{\mathcal{A}} | k \notin \underline{\mathcal{A}}]\right] \tag{4.3}$$

$$\text{UER}: \quad \bar{\epsilon}_s = \bar{\epsilon}_{md} \cdot \theta + \bar{\epsilon}_{fa} \cdot (1 - \theta) \tag{4.4}$$

$$\text{CER}: \quad \bar{\epsilon}_C{=}p[\hat{\mathcal{A}} \neq \underline{\mathcal{A}}]. \tag{4.5}$$

The MDR (resp. FAR) corresponds to the false negative (resp. false positive) rate. The UER combines these errors to compute an average individual error rate. In addition, the CER is a system level error rate, that counts the rate of non-perfect codeset detection.

4.2.2. MAP detectors

Let $\mathbf{y}_n \in \mathbb{C}^M$ denote the received signal on antenna n and $\mathbf{C}_{\mathcal{A}} \in \mathbb{C}^{M \times \omega}$ the codeset of a given node subset $\mathcal{A}$ whose cardinality is ω. Its singular value decomposition (SVD) is written

$\mathbf{C}_{\mathcal{A}} = \mathbf{V\Gamma U}$, where $\mathbf{V} \in \mathbb{C}^{M\times M}$ and $\mathbf{U} \in \mathbb{C}^{\omega\times\omega}$ are unitary matrices. $\mathbf{\Gamma} \in \mathbb{C}^{M\times\omega}$ is composed of the singular values γ on its diagonal. From (4.2.1), following [111], the likelihood of a codeset is given by:

$$p(\mathbf{y}|\mathcal{A}) = \prod_{n=1}^{N} \frac{1}{\pi^M |\sigma|} \exp\left(||\tilde{\mathbf{y}}_n||_2^2 - ||\mathbf{y}_n||_2^2\right), \tag{4.6}$$

where $\sigma \in \mathbb{C}^{M\times M}$ is $\sigma = \rho\mathbf{C}_{\mathcal{A}}\mathbf{C}_{\mathcal{A}}^{\mathsf{H}} + \mathbf{I}_M$ and $\tilde{\mathbf{y}}_n$ is the projection of $\mathbf{y}_n$ onto the codeset $\mathbf{C}_{\mathcal{A}}$ space, and is defined as:

$$\tilde{\mathbf{y}}_n = diag\left(\sqrt{\frac{\rho\,|\gamma_1|^2}{1+\rho\,|\gamma_1|^2}}, \cdots, \sqrt{\frac{\rho\,|\gamma_M|^2}{1+\rho\,|\gamma_M|^2}}\right) \mathbf{V}^{\mathsf{H}}\mathbf{y}_n. \tag{4.7}$$

The maximum likelihood estimate (MLE) has been used in [111] to estimate the active set. Since we know the prior probability on $(\mathcal{U}, \Phi)$, related to the Poisson distribution, a maximum a posteriori (MAP) detector can be defined and is optimal w.r.t. to the Bayes risk minimisation, when defined from the CER.

Definition 3 (C-MAP estimate)**.** *The C-MAP estimate of the codeset detection problem is given by:*

$$\hat{\mathcal{A}}_C = \text{argmin}\, \mathcal{A} \in \Phi p[\mathcal{A} \neq \mathcal{A}|\mathbf{y}] \tag{4.8}$$
$$= \text{argmax}\, \mathcal{A} \in \Phi p(\mathbf{y}|\mathcal{A})p(\mathcal{A}), \tag{4.9}$$

where $\hat{\mathcal{A}}_C$ is the detected subset, given the received signal $\mathbf{y}$, from the knowledge of the codebook $\mathcal{C}$, the Gaussian distributions of the channels $\mathbf{h}_k$ and noise $\mathbf{z}$.

In addition, the prior probability is given by $\mathbb{P}(\mathcal{A}) = \lambda^{|\mathcal{A}|} \cdot (1-\lambda)^{|\mathcal{U}|-|\mathcal{A}|}$.

But if the objective is to minimize the user error rate, the Bayes risk is modified and leads to the following estimate.

Definition 4 (U-MAP estimate)**.** *The U-MAP estimate of the codeset detection problem is given by:*

$$\hat{\mathcal{A}}_U = \cup_{k\in\mathcal{U}} \{k|\delta_k(\mathbf{y}) = 1\}, \tag{4.10}$$

with δ the delta function and where $\delta_k(\mathbf{y}) = 1$ is given by:

$$\delta_k(\mathbf{y}) = \begin{cases} 1 \textit{ if } \sum\limits_{\substack{\mathcal{A}\in\Phi;\\ k\in\mathcal{A}}} p(\mathbf{y}|\mathcal{A})p(\mathcal{A}) > \sum\limits_{\substack{\mathcal{A}\in\Phi;\\ k\notin\mathcal{A}}} p(\mathbf{y}|\mathcal{A})p(\mathcal{A}) \\ 0 \textit{ else} \end{cases}. \tag{4.11}$$

Let us prove that U-MAP is optimal with respect to the UER metric. $P_{MD}(k|\mathbf{y})$ and $P_{FA}(k|\mathbf{y})$, the probability of Missed Detection, respectively False Alarm, of a user k given a received signal $\mathbf{y}$ are given by:

$$P_{MD}(k|\mathbf{y}) = \sum_{\substack{\mathcal{A}\in\Phi;\\ k\in\mathcal{A}}} \mathbb{1}_{[k\notin\hat{\mathcal{A}}(\mathbf{y})]} p(\mathcal{A}|\mathbf{y}) = \mathbb{1}_{[k\notin\hat{\mathcal{A}}(\mathbf{y})]} \sum_{\substack{\mathcal{A}\in\Phi;\\ k\in\mathcal{A}}} p(\mathcal{A}|\mathbf{y}), \tag{4.12}$$

and

$$P_{FA}(k|\mathbf{y}) = \sum_{\substack{\mathcal{A}\in\Phi;\\ k\notin\mathcal{A}}} \mathbb{1}_{[k\in\hat{\mathcal{A}}(\mathbf{y})]} p(\mathcal{A}|\mathbf{y}) = \mathbb{1}_{[k\in\hat{\mathcal{A}}(\mathbf{y})]} \sum_{\substack{\mathcal{A}\in\Phi;\\ k\notin\mathcal{A}}} p(\mathcal{A}|\mathbf{y}). \tag{4.13}$$

Minimising the UER thus corresponds to performing a binary test for each user, by comparing $P_{MD}(k|\mathbf{y})$ and $P_{FA}(k|\mathbf{y})$:

$$\delta k(\mathbf{y}) = 1 \text{ if } \sum_{\substack{\mathcal{A}\in\Phi;\\ k\in\mathcal{A}}} p(\mathcal{A}|\mathbf{y}) > \sum_{\substack{\mathcal{A}\in\Phi;\\ k\notin\mathcal{A}}} p(\mathcal{A}|\mathbf{y}). \tag{4.14}$$

Then, having $p(\mathcal{A}|\mathbf{y}) \propto p(\mathbf{y}|\mathcal{A})p(\mathcal{A})$, the decision given in (4.11) is optimal.

To compute the estimate given either by eq. (4.9) or (4.11), each element of the power set Φ has to be evaluated, making such kind of AUD non feasible for computational reasons.

4.2.3. It-MAP detector

As an alternative to these solution with prohibitive computational complexity, many iterative algorithms have been proposed in the literature [118],[111],[112]. Following the work presented in [111], the Iterative-MAP (It-MAP) is herein proposed as a reference solution, built as an approximation of C-MAP. The philosophy of It-MAP is similar to that of a Successive Interference Cancellation (SIC) as it processes the received signal $\mathbf{y}$ iteratively and retrieves a new detected user at each iteration i based on the assumption of the previous $\hat{\mathcal{A}}$ at $i-1$. Even if the detected subset is built iteratively, the detection rule is based on the MAP criteria given by eq. (4.9) for each i, with a search restricted on some elements of Φ. More precisely, the evaluated subsets $\mathcal{A}_i \in \Phi_i$ are built from the previously detected subset $\hat{\mathcal{A}}_{i-1}$ as follows:

$$\Phi_i = \cup_{k\in\left\{\mathcal{U}\setminus\hat{\mathcal{A}}_{i-1},\emptyset\right\}} \left\{\hat{\mathcal{A}}_{i-1}, k\right\} \tag{4.15}$$

The It-MAP detection stops as soon as two successive iterations provide the same detected subset, i.e., $\hat{\mathcal{A}}_{i-1} = \hat{\mathcal{A}}_i$.

The architecture of this MAP-based AUD algorithm makes its complexity lower than the computation requirement of the MAP, but at the cost of a reduced accuracy since the iterative detection makes the It-MAP prone to error propagation. This fact let room for other AUD algorithms seeking for a better complexity-accuracy trade-off. As DL is envisioned to accommodate well to large scale problems, a blind AUD based on DL is thus presented in the following section and will be compared to the C-MAP, U-MAP and It-MAP detectors.

4.3. A neural network based algorithm

4.3.1. The NN-MAP estimate

Definition 5 (NN-MAP estimate)**.** *A NN-MAP architecture for the AUD problem is defined as follows. The inputs to the NN-MAP detector come from* $\mathbf{y}$*, from eq.* (4.1)*, and* ρ *(in dB) as a side information. It outputs a vector* $\mathbf{p}$ *of length K containing the estimated probability that each node is active. That probability is obtained by using a* sigmoid *activation function at the end of the network and is compared with a vector of ground truth labels* $\mathbf{t}$ *of the same length through a binary cross-entropy*

loss function $\mathcal{L}$ to optimise the network's parameters:

$$\mathcal{L}(\mathbf{t}, \mathbf{p}) = -\sum_{k=1}^{K} t_k \cdot \log(p_k) + (1 - t_k) \cdot \log(1 - p_k) \tag{4.16}$$

During the training phase, the average cost over all tuples $(\underline{\mathcal{A}}_i, \mathbf{y}_i, \mathbf{p}_i)$ is :

$$\bar{\mathcal{L}} = \frac{1}{I} \sum_{i} \mathcal{L}(\mathbf{t}(\underline{\mathcal{A}}_i), \mathbf{p}_i), \tag{4.17}$$

where I is the mini-batch size.

After training, the soft probabilities are converted to hard decisions using a threshold: a user k is considered active if $p_k > 0.5$.

This choice is justified by the following theorem.

Theorem 2. *For a non-coherent AUD problem, the solution that minimizes the cost function given by* (4.17) *converges to the U-MAP estimate of definition 4 if the dataset is large enough.*

Proof. By incorporating (4.16) into (4.17), and permuting the sums w.r.t. i and k, one can write:

$$\bar{\mathcal{L}} = \sum_{k \in \mathcal{U}} \bar{\mathcal{L}}(k), \tag{4.18}$$

with :

$$\bar{\mathcal{L}}(k) = -\sum_{i=1}^{I} \mathbb{1}_{[k \in \underline{\mathcal{A}}_i]} \log(p_k(\mathbf{y}_i)) + \mathbb{1}_{[k \notin \underline{\mathcal{A}}_i]} \log(1 - p_k(\mathbf{y}_i)). \tag{4.19}$$

Then if the tests are randomly selected according to the prior probability $\mathbb{P}(\underline{\mathcal{A}})$, one have:

$$\bar{\mathcal{L}}^*(k) = \lim_{I \to \infty} \bar{\mathcal{L}}(k) \tag{4.20}$$

$$= -\mathbb{E}_{\underline{\mathcal{A}}, Y} \left[\mathbb{1}_{[k \in \underline{\mathcal{A}}]} \log(p_k(\mathbf{y})) + \mathbb{1}_{[k \notin \underline{\mathcal{A}}]} \log(1 - p_k(\mathbf{y})) \right]. \tag{4.21}$$

Finally, using the decomposition $\bar{\mathcal{L}}^*(k) = \int_{\mathcal{Y}} \bar{\mathcal{L}}^*(k|\mathbf{y}) \cdot f_Y(\mathbf{y}) \cdot d\mathbf{y}$, one gets:

$$\bar{\mathcal{L}}^*(k|\mathbf{y}) = -P(k|\mathbf{y}) \cdot \log(p_k(\mathbf{y})) + (1 - P(k|\mathbf{y})) \cdot \log(1 - p_k(\mathbf{y})). \tag{4.22}$$

The minimum of (4.17) is asymptotically achieved if the NN returns for each observation $\mathbf{y}$, the output $\mathbf{p}$ which minimizes (4.22) for any user k, independently.

Thanks to the convexity of this function w.r.t. $p_k(y)$ it is straightforward to show that the global cost function is minimal when $p_k(\mathbf{y}) = P(k|\mathbf{y})$, which is nothing but the posterior probability. Therefore, if the learning phase succeeds to find a set of parameters for the NN such that the output probability vector converges to the posterior probability distribution, the U-MAP estimate is achieved by selecting at the output all the nodes k with $p_k \geq 0.5$. Clearly, the NN architecture approximates the U-MAP. □

It is worth mentioning that there is no guarantee that a NN can achieve this global optimum. This theorem only claims that the objective function based on the cross-entropy is well-posed w.r.t. the U-MAP problem.

4.3.2. The NN-MAP system parameters

In this chapter, the NN architecture and hyper parameters have been empirically optimised as follows, for a scenario chosen with $K = 10$, $M = 8$, $N = 4$, $\lambda = 4$. One random codebook is generated and reused for all subsequent training for a fair comparison of the performance thus obtained. Unless specified otherwise, all figures correspond to this scenario.

Architecture type : The observation vector $\mathbf{y}$ is a combination of random Gaussian variables related to the properties of the codebook, channel realisations and noise. The unique correlation may come from the codebook, which is selected randomly. As such, the correlations are low and convolutional layers are not mandatory. In addition, we don't assume in this model any correlation in the data transmissions, neither between nodes, nor over time. Under these assumptions, a recurrent layer is not necessary. Consequently, the chosen architecture is a fully-connected neural network.

Layers: A set of networks was trained with an increasing amount of dense layers and a constant amount of units in each one, from 3 to 12. The search was not expanded above due to lack of significant improvement. Finally, the 5 layers network was selected.

Units: The number of units per layer was set to be proportional to K, M, and N so it can scale according to scenario complexity. The proportionality factor was chosen by increasing it by powers of 2 from 1 to 128. Performance stopped increasing after 4, so this is what is used in the following.

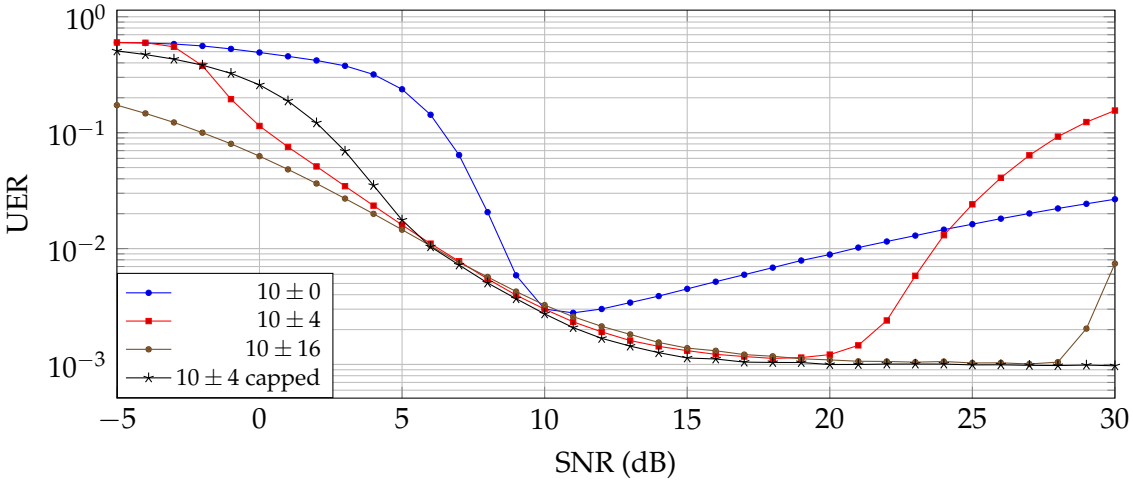

Figure 4.2.: Network trained on examples with uniform SNR. Mean is kept at 10 dB and range is varied. For the capped line, the SNR value input to the NN at test time is limited to the training range.

SNR range: Training data is generated with a given ρ, which is also input to the NN. It is therefore necessary to determine the values that will be used to train the network with. Indeed, ρ can have a big impact on the overall performance of the trained network: a too low value

Table 4.1.: Summary of network and training parameters

Parameter	Value
Layers	5
Units	$4 \times K \times M \times N$
Learning rate	1×10^{-3}
Optimiser	Adam
Batch size	4096
Training iterations	100000
Training SNR values	10 dB $\pm$ 16 dB

could generate a very noisy signal from which the NN would fail to learn anything. A too high SNR would not help the NN to learn to cope with noise. In our approach, the training dataset is generated with a range of SNR values, distributed uniformly over a specified interval. To find out a good interval, the impact of two parameters is evaluated: the range of the interval (Fig.4.2) and the mean of that interval. In those two cases, NN shows good performance only inside the SNR interval used for training, with a rapid decrease outside.

Note that when a NN has to work above its learned range, it appears more efficient to limit the SNR values input to the range bounds. This is highlighted in the curve (—⋆—), where the NN learned in the range 10 ± 4 and performs well at SNR above 14 dB. This is not the case for low SNR values, where the NN learned in the range 10 ± 16 outperforms all other curves.

The most important parameters of the NN used in this chapter are summarized in Table 4.1.

4.4. Results

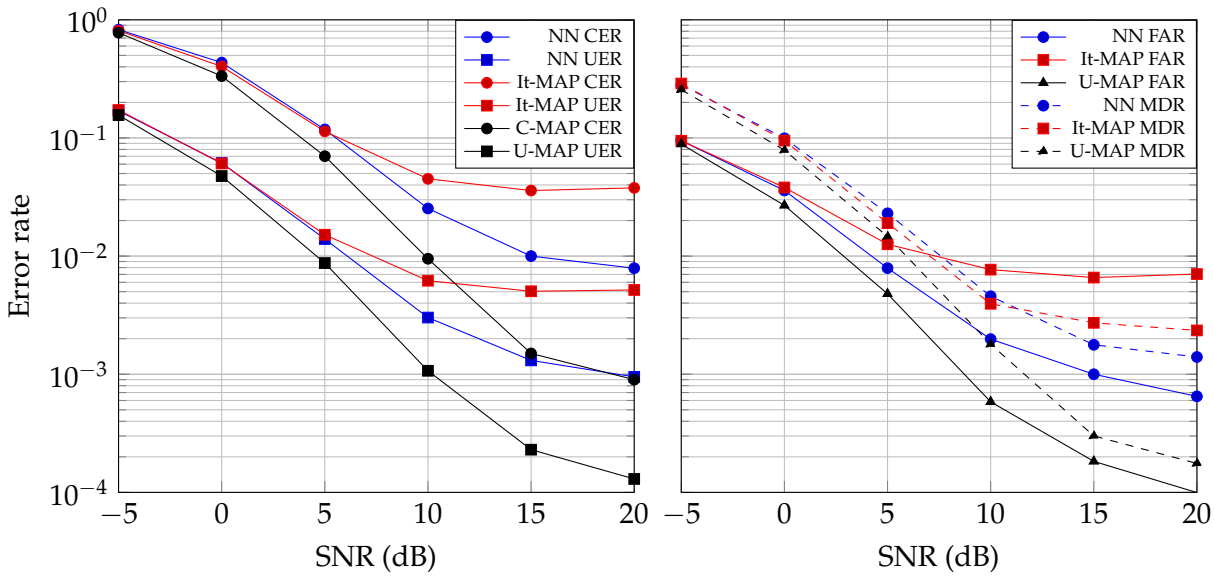

Figure 4.3.: Performance comparison between C-, U-, It- and NN-MAP algorithms for all the metrics used.

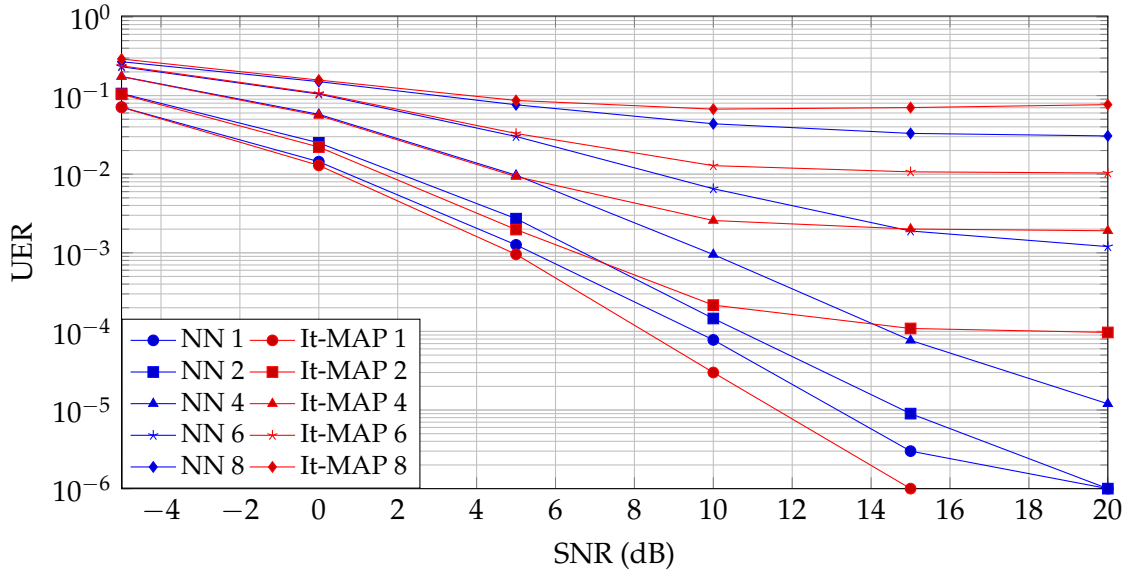

Figure 4.4.: It-MAP and NN-MAP are set and trained with $\lambda = 4$. A test time, active user number is not random and varied from 1 to 8.

Comparison between the different algorithms: The different algorithms defined in the previous section are compared for the chosen scenario ($K = 10$, $M = 8$, $N = 4$, $\lambda = 4$).

As shown in Fig. 4.3, the trained NN-MAP outperforms It-MAP, especially at high SNR w.r.t. UER and CER metrics as well. The performance of NN-MAP bridges half of the gap with the MAP's optimal given either with C-MAP or U-MAP. In addition, as can be seen in Fig.4.3, NN-MAP outperforms It-MAP for both MDR and FAR criteria. It is worth noting however that It-MAP achieves a MDR lower than FAR while it is the opposite for NN-MAP. Note that this trade-off may be easily tuned with NN-MAP. Indeed, in (4.22), we proved the relationship between the MAP and the loss function. It is known that the MAP solution is optimal only if mis-detection and false alarm errors have the same cost. However, it is known that the relative weight of these errors can be tuned to achieve any point of the ROC curve of the detector [119]. For the NN-MAP, one have two options to balance MDR and FAR: the cross-entropy can be modified or the hard decision threshold can be tuned.

All algorithms tested in this chapter assume the knowledge of λ as a prior information. Fig.4.4 shows how It-MAP and NN-MAP perform when the actual number of users deviates from the expected value (the actual number of active users is indicated in the legend). It-MAP outperforms NN-MAP only for 1 active user. It is interesting to mention that a method to increase the performance of NN-MAP in that regard could be to either train it with several values of λ, or to use a uniform distribution instead of the distribution associated to the Poisson distribution assumption.

Analysis on larger scenarios: In Fig.4.5, the performance results of It-MAP and NN-MAP are given when K is increased. Note that in this scenario, both U-MAP and C-MAP are not computable in reasonable time. There, NN-MAP outperforms It-MAP with $K < 20$, but both saturate at the same level for higher number of potential users.

The neural network can be adapted to work on bigger scenarios, such as the one in Fig.4.6,

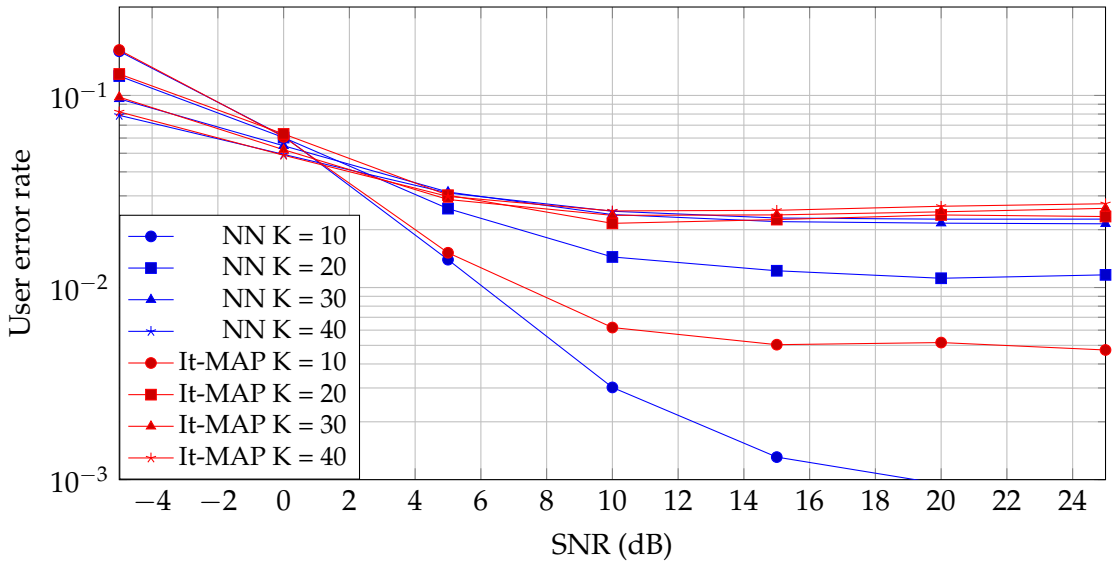

Figure 4.5.: UER comparison between NN-MAP and It-MAP algorithms with increase in potential users. $\lambda = 4, M = 8, N = 4$.

Table 4.2.: Average example processing time for a scenario with $K = 10$, $M = 8$, $N = 4$, $\lambda = 4$.

MAP CPU	It-MAP CPU	NN CPU	NN GPU
159 ms	8.25 ms	3.6 ms	542 ns

with an important, and expected impact of the codelength on the performance, but in this situation, even the It-MAP becomes difficult to compute in a reasonable time, so performance comparison is not available.

Computational considerations: On top of performance increases, the NN approach also provides a reduction of computation times as shown in table 4.4 and Fig.4.7, where CPU executions are single thread and the GPU execution allows a batch size of up to 100000. The source of this reduction is twofold: a less complex computation through simple add and multiply operations, without branching leads to a reduced load and smoother execution on the system, and it also creates the capability to massively parallelise batches of computations in an efficient manner. The second aspect is most prevalent in the present case: the It-MAP rests on sequential operations with nested loops that create inefficiencies through CPU branch prediction misses and are not trivial to convert to parallel computation, and so, to offload to high performance accelerators such as a GPU.

4.5. Conclusion

In this chapter, we have proposed to use DL for the non-coherent AUD problem based on coded domain NOMA. We have proposed a NN-MAP detector which minimises asymptotically the MAP cost function. We have shown that this approach improves on the performance of state-of-the-art iterative algorithms by a factor of five in some scenarios, especially w.r.t.

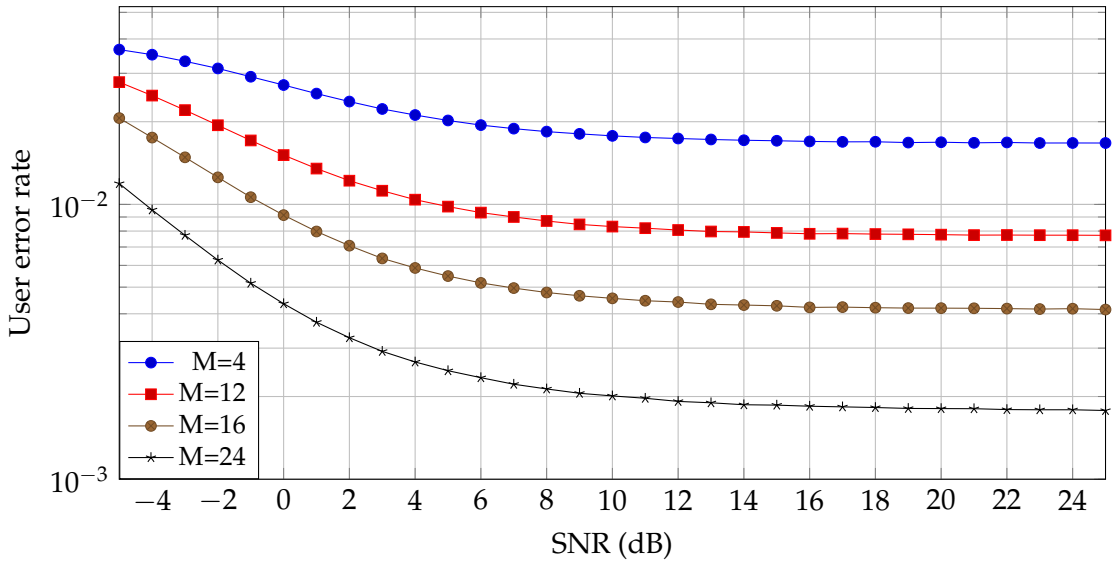

Figure 4.6.: Performance comparison between NN-MAP with increasing codelength. $K = 100$.

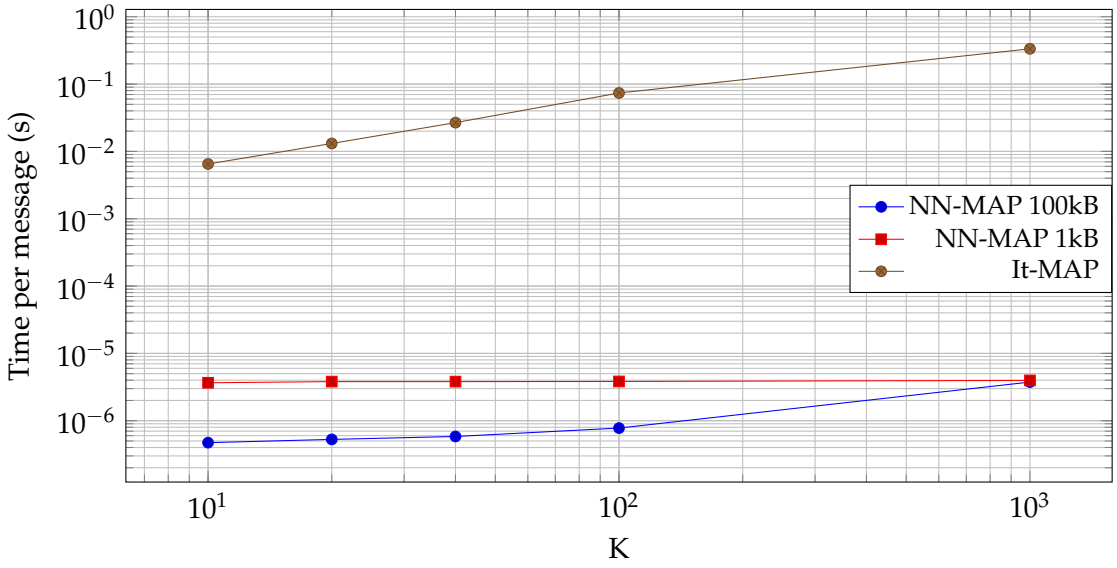

Figure 4.7.: Computation time evolution with increased K. The NN-MAP is tested with 1000 messages per batch and 100000 messages per batch. In and with $K = 1000$, processing does not fit inside the GPU, so the batch size is reduced to 10000.

the UER metric. Moreover, as it also reduces the algorithmic complexity, this work shows the interest of using DL for such a task even though more work still needs to be done to improve the scalability of the architecture to very large sets of nodes as expected for massive access IoT.

— 5 —

Geometric shaping for uplink NOMA

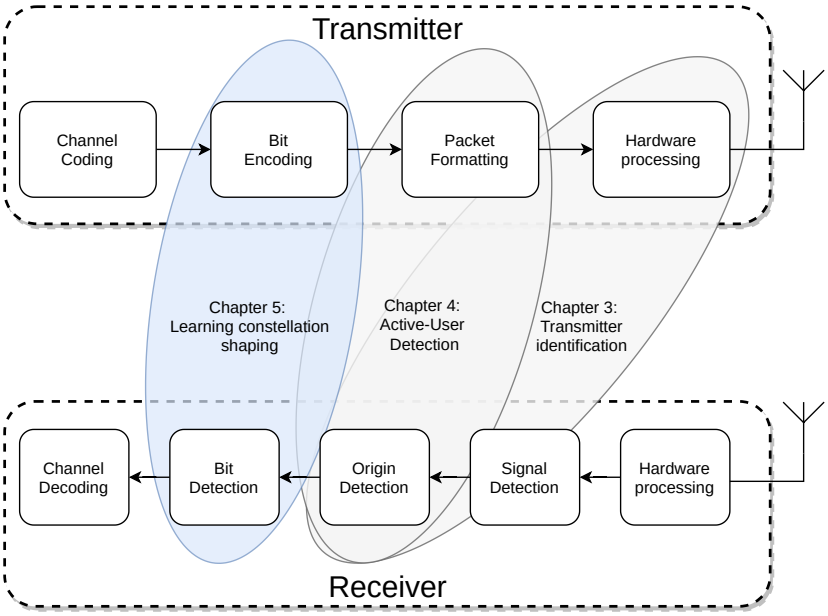

Figure 5.1.: Position of the chapter in the thesis outline

AFTER having studied the problems of detecting active users among a pool of possible devices in the previous chapters, the next step is to be able to efficiently transmit data by these devices. This chapter addresses the task of designing good modulation constellations to maximise the transmitted rate, tailored to specific channel and multi access conditions.

The first section 5.1 presents a background on constellation shaping, also called geometric

shaping (GS), then section 5.1.1 presents preliminary implementation works in a peer-to-peer setting. Section 5.2 addresses the work done to extend the deep learning geometric shaping approach to a two users uplink NOMA or multiple-access channel (MAC) scenario.

5.1. Geometric constellation shaping

In digital communications, the constellation describes a mapping between *messages* that are sequences of bits of a certain length, the constellation *order*, and tuples of N_{real} real numbers. Each possible tuple of real number is called a *symbol* and there are 2^N_{bpm} symbols in a constellation of order N_{bpm}. In other words, N_{bpm} bits per message. The act of designing a specific constellation is called constellation shaping or geometric shaping.

Traditional constellations operate on $N_{real} = 2$ describing the I and Q values of a complex symbol, as described in Chapter 2, and this can be represented on a plane, leading visually to a constellation of symbols, hence the name. But they can also have a higher dimension, being transmitted over several *channel uses*, successively in time or in frequency.

The main objective of GS is to place the symbols such that, after going through the distortions of the channel, the resulting points are as far away from each other as possible to reduce decoding errors on the receiver. The principal constraint of that is the fact that the transmission power is limited and defined by factors external to the constellation design (energy available, hardware, protocol or legal constraints). For this reason, constellations are usually designed with an average power of one, leaving the power scaling to other parts to ensure that the power is as high as possible without going over the limit. This means that, if some symbols take higher values to increase their separation, the rest needs to aggregate around zero to keep the average power at one.

The classical constellations can be separated in categories leading to different characteristics. The naming convention is to prepend the type of constellation with the number of symbols in it, e.g. 16-PSK for a constellation of 16 symbols (so four bits) of the PSK type.

Amplitude shift keying (ASK) also referred as pulse amplitude modulation (PAM), is the simplest, usually only involving one of the I and Q components of the complex symbol, and encoding the bits on the amplitude of the symbol. This constellation is very simple to implement and robust to phase noise (rotations of the constellations), but not to amplitude variations that are common in wireless settings. The simplest of them, with an order of 1 and setting the bit 0 to 0, and the bit 1 to 2 is also called on-off keying (OOK).

Phase shift keying (PSK) is designed to solve the weakness of ASK to amplitude variations by placing all the points on the same amplitude, on the unit circle. This means that the receive power does not affect the decoding at all, but that only a small amount of phase rotation is needed to create ambiguity. The 2-PSK variant, usually called binary phase-shift keying (BPSK) can also be seen as an ASK type constellation.

Quadrature amplitude modulation (QAM) is the middle ground between the previous two types, having both phase and amplitude variations. It is a better choice when the noise is evenly distributed between phase and amplitude, or the I and Q components, allowing

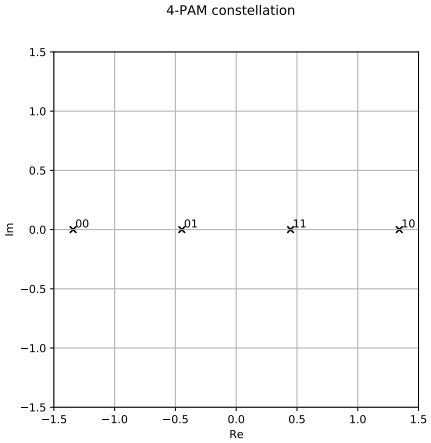

Figure 5.2.: Normalised 4-PAM constellation with Gray labelling

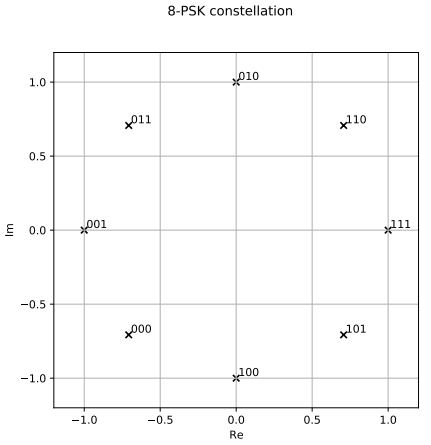

Figure 5.3.: Normalised 8-PSK constellation with Gray labelling

for more spreading of the symbols, or the usage of a higher order constellation. The 4-QAM variant, also called QPSK, can also be considered a PSK constellation.

All these constellations are designed for point-to-point (P2P) communications and they can get very close to the channel capacity, especially with an AWGN channel, as can be seen in Fig. 5.5, and protocols can be designed to automatically adapt the modulation order according to the channel quality, such as the LTE modulation and coding scheme [120]. But they are less tailored to more complex channel types and also, a loss of performance can occur around the discrete modulation order switches. A more continuous set of constellations could allow for some small gains in the zone of the steps.

All these sets of points, and the ones presented later in this chapter are the result of an optimisation process. But several metrics can be optimised for in this context. A simple objective can be the minimum distance (MD), that involves maximising the inter-symbol spacing by looking at the minimum distance between any two points. It is an intrinsic property of the constellation so simplifies operations by not requiring to test it on a specific channel for

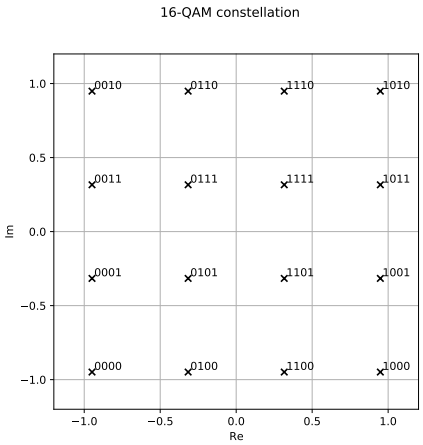

Figure 5.4.: Normalised 16-QAM constellation with Gray labelling

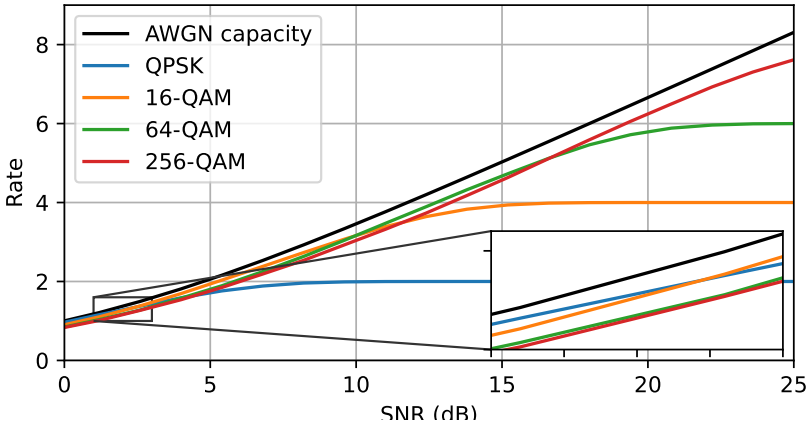

Figure 5.5.: Achievable rates for QAM constellations compared with AWGN channel capacity

optimisation. On the other hand, the result can be less adapted to channel conditions. Most operators require the highest throughput possible, to better serve customers, and increase spectrum efficiency. To meet this requirement, a mutual information (MI) or achievable rate can be used, as in Fig. 5.5. Finally, reliability can be focused on, with considerations given towards error rates. The SER measures the rate at which a transmitted symbol is mistaken for another one. In constellations with an order $N_{bpm} > 1$, a symbol decoding error impacts more than one bit, and this is where the issue of *bit labeling* comes into play. The affectation of a bit sequence to the 2^N_{bpm} symbols impacts the bit error rate (BER). As an example, let us set $N_{bpm} = 3$ and examine three symbols: A, corresponding to the bit sequence 000, B corresponding to 001, and C to 111. Let the symbol A be transmitted. If the receiver decodes B, one symbol error is recorded, and the corresponding third bit goes from 0 to 1, leading to one bit error. But is the receiver decodes C, the three bits go from 0 to 1, leading to three bits in error, while there is still only one symbol error. So SER and BER evolve differently, depending on the bit labelling scheme for the constellation. The constellation examples presented above

all use a Gray labelling scheme where adjacent symbols differ by only one bit to improve the BER. High order modulations can lead to difficulties in the design of a good Gray labelling due to the exponentially high amount of combinations.

5.1.1. Learning GS for Point-to-Point SISO communications

The challenges discussed above have lead to the idea of training neural networks to do the constellation shaping. This is originally done by viewing the end-to-end transmission of a point-to-point communication with constellation encoding, transmission over a channel, and decoding as one unique autoencoder [8]. The process leads to good convergence, but it requires that the channel be known, derivable, and implemented in a way that allows the gradient of the loss function, computed on the receiver part to be propagated up to the transmitter part. For transmitting effectively over a real channel, the authors in [121] first propose to train the autoencoder with a suitable channel model, that matches the real one as closely as possible, before fine-tuning only the receiver part to adapt it to the real, over-the-air, conditions. Another option is to use a secondary neural network, and train it as a GAN to closely mimic the true channel [122]. In this case, the signal can be transmitted on the real channel, and the gradient can be propagated through the channel modelling neural network back to the receiver. The third proposed way to operate on an unknown channel is proposed in [123], [124]. This method is inspired from policy learning of *RL* to estimate the gradient at the transmitter, based on the loss reported by the receiver. It removes the need for an explicit channel model but still relies on a perfect feedback link to report the loss values. The work in [125] shows that this type of system can also be adapted to work in duplex, also learning the feedback transmission.

The works mentioned above focus on learning the constellation shape but not the labelling. They operate *symbol wise* autoencoders that optimise for the received constellation symbols and not the received bits. As discussed in the previous section, this labelling has an impact on the transmission performance and can be challenging to perform. The authors in [126] implement a *bitwise* autoencoder where the loss function is computed on the received bits. This approach leads the system to also learn the bit labelling and it modifies the constellation towards which it converges. This paper also uses the reinforcement learning (RL) style model-free technique to perform some experiments over-the-air with the bitwise approach.

When operating over simple channels, such as AWGN, the performance gains over classical constellations are marginal, and mainly come from the ability to tailor a constellation to the current noise conditions. The interest of learning GS, combined with the training of the matched receiver becomes more apparent on more challenging channels, as can be seen in [127]. In this paper, transmission is done over an OFDM system on a Rayleigh block fading channel, and the receiver operates on the entire received frame. In this condition, the encoder can learn to superimpose information on the constellation points that allow the receiver to work without pilots, leading to important throughput gains.

To prepare the GS work presented later in this chapter, I have made an implementation of a point-to-point autoencoder in GNU Radio that can be run inside of FIT/CorteXlab to train constellations over-the-air with the RL style model-free technique of [124] in both bit-wise and symbol-wise configurations. This implementation results to a similar setup and results as the one in [126] and is presented in Appendix A and is also available online [128].

5.2. GS in multiple access

5.2.1. Uplink NOMA

The P2P system, mentioned in the previous section implies that the transmitter and receiver are alone in their environment, at least in their frequency band, without interference from the outside world. In many real operating conditions, such as cellular communications or IoT networks, a massive amount of devices may want to access the shared spectrum resource, for instance to talk to a base station. Classical methods to handle this multiple access scenario is to divide the shared spectrum resource into time and/or frequency slots and allocating these to only one user at once, in order to revert, inside of a slot, to the conditions of a P2P transmission. These methods are known as time-division multiple access (TDMA) and frequency-division multiple access (FDMA). The TDMA comes in two flavours: the maximum power constrained TDMA, where each user transmit at their maximum power, be it limited by hardware or legal constraints, a fraction of the time, and average power constrained TDMA, where each user maintains a set average power overall. In that case, since the devices only transmit a fraction of the time, they can raise their emitted power to keep the average power and increase their SNR, and thus, their transmission performance, when they are actually transmitting. It can be noted that the average power TDMA is less realistic to implement since, in most operating conditions, hardware considerations such as the amplifier characteristics or available energy, or regulatory limitations of the frequency bands limit the maximum transmit power of devices. This average power TDMA case leads to a similar behaviour as with FDMA where the transmission is always occurring, at a constant power, but over a fraction of the frequency band. When using less frequency, the available power is more concentrated, also increasing the SNR. These orthogonal multiple access (OMA) approaches allow the use of well understood P2P transmission techniques but they greatly reduce the amount of users able to use the resource at once and may require an important signalling overhead to allocate slots to users.

The NOMA approach is based on allowing multiple transmissions on the same resource slot, leading to a potential increase of capacity in theoretical results [129]. It can also help reduce latency by reducing the time to wait for an available slot. These two promises make this approach a great candidate for future cellular technologies where user counts are expected to increase in an ever more constrained spectrum resource and with the appearance of devices expecting very low latencies. The NOMA techniques are generally classified as either power-domain, with transmission power being adapted to the channel states such that the constellation of one user can fit inside the other without interference, a method known as superposition coding. The receiver can then use SIC to iteratively decode the symbols, in a decreasing power order. The other broad category is the code-domain, where each user is affected a unique code used as a spreading sequence, the codes having interesting properties such as low density and inter-correlation. This method eliminates the need to finely estimate the channel and synchronise the transmitters to tailor transmission powers, but increases user interference. A recent type of code-domain NOMA, sparse code multiple access (SCMA) [130], focuses on designing sparse codes an modulations over multiple channel uses. These two classes operate on top of existing constellations, but these classical constellations, and the work presented in the previous sections, are adapted for use in OMA systems, but not for NOMA, where the deliberate interference would greatly reduce the decoding performance. For this reason, there is a need to develop performant constellations tailored to the NOMA

situations and this section proposes to continue the deep learning geometric shaping approach previously developed for P2P and extend it to the two users NOMA setting. The proposed technique of designing low interfering constellations can be seen as a form of code-domain NOMA, where the constellation act as short coding sequences, but with an added power-domain NOMA aspect in the form of constellation power allocation. The previous chapter also handle code-domain NOMA, but with a focus on designing the receiver, without looking at designing the coding sequences, and it also focused on a scenario with more than two users, and with longer codes.

5.2.2. State of the art

Some work has been done in the past decade to study the performance of digital, finite order constellations in this single-input single-output (SISO) uplink NOMA setting. Table 5.1 provides a comparison of these works along several categories:

Channel conditions: The channel conditions greatly affect the performance of a given constellation. The majority of papers use a purely Gaussian channel model while the others use Rayleigh fading. To further distinguish between the various Rayleigh fading implementations, the *Diff rot* category corresponds to the cases where the channel model can cause different phase rotations for the two constellations, a phenomenon that is not possible with Gaussian channel. The *symmetry* column corresponds to the cases where the different user's transmission are symmetric in term of received power or SNR level at the receiver. As will be discussed in Section 5.5.3, a high level of asymmetry allows more performance gains compared to traditional orthogonal methods.

Constellation shaping: Even though all these works study systems with digital constellations, not all of them try to improve these constellations. The *GS* category lists those that do geometric shaping, while the *Const opti* provides more insight on the manner the shaping is performed. It distinguishes between the cases where the entire constellations are optimised as one, and not (and this is mostly the case for neural network implementations) on a symbol by symbol basis. More information on this distinction can be found later, in Section 5.4.3.

Metrics: The *QAM Comp* column list works that compare the obtained constellations (only applicable if the paper performs shaping) to traditional QAM constellations (or other types of classical constellations). The *OMA Comp* category lists the paper that compares obtained performance results in this non-orthogonal system with more widespread orthogonal techniques (in this case TDMA). It can be noted that only one paper in this list, [129] does this comparison and it is quite surprising: The NOMA approach claims performance increases from the OMA techniques but this claim is rarely supported with comparison results. Non-orthogonal multiple access implies more than one user, so more than one performance metrics, leading to possible trade-offs between the competing users. The *rate tradeoff* column lists the papers that study this tradeoff. Again, all but one paper only limit the study to aggregate metrics such as sum rate or average error rates.

Transmission scenario: Two main types of scenarios are tackled in the literature relative to the number of users U, and the the number of channel uses the constellations span T:

Table 5.1.: Comparison of SISO Uplink NOMA literature papers using digital constellations

Paper	Channel	Diff rot	GS	Const opti	QAM Comp	OMA Comp	Rate Tradeoff	U	T	M	Symmetry
[129]	Gaussian	No	Yes	Yes	Yes	Yes	Yes	2	1	2	Yes
[131]			Yes	No	Yes	No		2	1	2,3	1.29-2.15dB
[132]			Yes	Yes	No			2	1	2	Yes
[133]			Yes	Yes	No			2	1	2	Yes
[134]			Yes	Yes	Yes			2	1	1,2	$P_1 = 1.5P_2$
[135]			Yes	No	No			6	4	2	Yes
[136]			Yes	Yes	Yes			6	4	2	Yes
[137]			Yes	No	No			6	4	2	Yes
[138]			Yes	Yes	No			6	4	2	Yes
[107]			Yes	No	No			6	4	2,3,4	Yes
[139]			No	No	No			6	4	2	Yes
[116]	Rayleigh	Yes	No	N/A	N/A			2	1	2	Yes
[140]		Yes	No	No	Yes			2	1	4	$P_1 = P_2 + 10dB$
[141]		No	Yes	Yes	Yes			2	1	2,4	Full range
[142]		No	Yes	Yes	Yes			6,10	4,5	2,4	Yes
[143]		Unclear	Yes	No	No			6	4	2	4,4,4,1,1,1
[144]		Yes	Yes	No	Yes			6	4	2	Yes
[145]		Yes	No	N/A	N/A			100	5	1	Yes
[146]		Yes	No	N/A	N/A			2000	500	2	Distance sim

- $U = 2$ and $T = 1$
- $U = 6$ and $T = 4$

In both scenarios, the constellation order (M) is of two to four bits, meaning that the two user scenario has four times the bit rate compared to the six users scenario while it has more users per channel uses. On the other hand, the constellations are smaller and thus simpler to optimise.

Harshan et Rajan are the first in this series to tackle the constellation shaping problem, in the Gaussian channel with symmetric propagation conditions. They first compute and study the capacity regions of standard constellations, then propose a scheme where standard constellations are rotated to improve the capacity of the resulting system via optimisation of the mutual information metric in [129]. In [131], they propose another method where standard constellations are scaled up and down alternatively, maintaining an average power constraint, and show this approach to have better performance than their previous constellation rotation method but only with regards to sum capacity. The constellation rotation approach is continued in [141] where the rotations are done to optimise for a minimum distance criterion allowing a closed form computation of the optimal rotation angle and a sum rate increase compared to the previous method. This paper involves Rayleigh fading, but the channel coefficient are set to be identical for the two devices, allowing a conservation of the constellation rotation regardless of the channel effect. The authors of [138] also build on this approach, using it to handle six users with multi dimensional constellations of four channel uses. In [132] and [133], the authors use a real valued Gaussian channel to study LDPC decoding with two users, and they propose their own constellation schemes for it, but the constellations are limited to the space of reals. Two more papers study the performance of two users systems, with Rayleigh fading [116], [140], but they do not learn constellation shaping, focusing on the receiver side. The most recent paper regarding two users, [134], is also the most relevant to the proposed approach, on a channel with some asymmetry. The constellations are not based on rotated standards, but are build as parallelogram shapes, optimising the minimum distance between symbols of the resulting combination. This papers compares to and outperforms [141] that compares to and outperforms [129].

The papers with six users scenarios demonstrate the use of deep learning for constellation shaping [107], [135], [137], [143], [144], where neural networks are used to encode symbols and decide time allocations along the four available channel uses. We can note that the networks are directly tasked with encoding the messages into symbols, and not to create explicit constellations that can then be used for encoding. The encoders also do not take into account the noise level to compute their encoding.

This chapter proposes the use of Deep learning to learn constellations for the two users case, focusing on the performance comparison with competing orthogonal methods and the rate trade-offs available from using a non orthogonal approach to transmission.

5.3. System model

This work simulates a system with two users, TX 1 and TX 2, not co-located and with only statistical channel state information (CSI), simultaneously transmitting data over a channel, into a unique receiver RX. In contrast to the previous chapters, the two users are always active and the receiver expects them to be. The two users transmit complex data signals x_1 and x_2

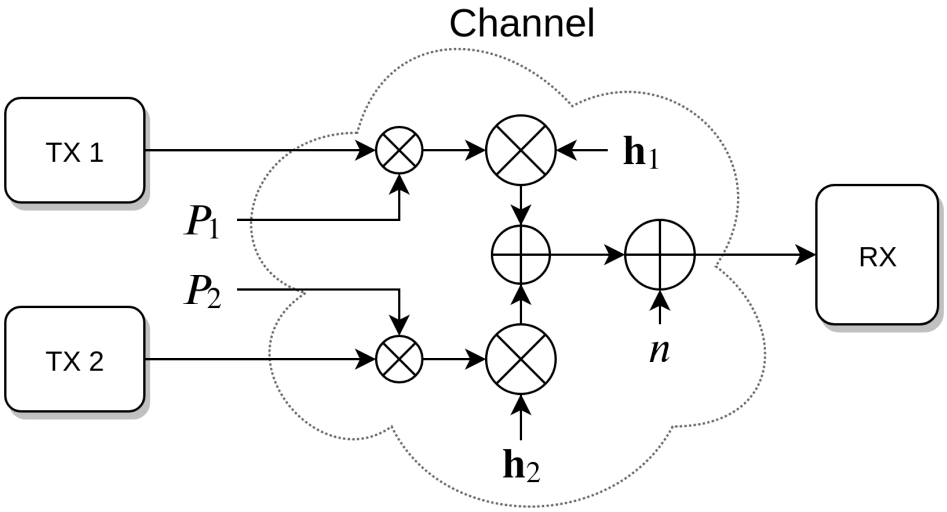

Figure 5.6.: Simulated transmission model. In the Gaussian scenario, $\mathbf{h}_1 = \mathbf{h}_2 = 1$ and in the other $\mathbf{h}_1$ and $\mathbf{h}_2$ are generated according to a correlated OFDM model.

with an average power normalised to 1. Since they are not co-located, their signals experience different path loss conditions, represented by their average received powers P_1 and P_2. As shown in Fig. 5.6, the two signals are then affected by the complex channel coefficients h_1 and h_2 that have different properties depending on the chosen scenario (described later) but have an average power of 1 (i.e. $\mathbb{E}\left[||h||_2\right] = 1$) to not interfere with the power constraints defined above. The two paths are then added together, followed by Gaussian noise $n \sim \mathcal{N}_{\mathbb{C}}(0,1)$, leading to the received signal y:

$$y = (x_1 \times \sqrt{P_1} \times h_1) + (x_2 \times \sqrt{P_2} \times h_2) + n \tag{5.1}$$

In this generic model, x, h, n, and y are scalar values, but, depending on the scenario and implementation, they can be manipulated as vectors and matrices, replacing the scalar addition and multiplications in Eq. 5.1 by their element-wise counterparts.

Since $\mathbb{E}\left[||n||_2\right] = 1$, the SNR values are directly equal to the receive power constraints $SNR_1 = P_1$ and $SNR_2 = P_2$.

In later implementation and results, the channel will follow one of two scenario: an AWGN channel and a more realistic but also more challenging correlated OFDM channel.

5.3.1. AWGN scenario

In this case, $h_1 = h_2 = 1$, leading to no distortion to the signal additional to the noise. This is a similar scenario to [129], [131]–[134], but, in these, only [131], [134] allow for asymmetry, $P_1 \neq P_2$. This scenario, also later referred to as *Gaussian scenario*, acts as a stepping stone, allowing to test the geometric shaping for two users uplink NOMA on a simple and well understood channel before moving on to more realistic and challenging conditions. It is also the scenario that allows comparison with relevant previous works.

5.3.2. Correlated OFDM scenario

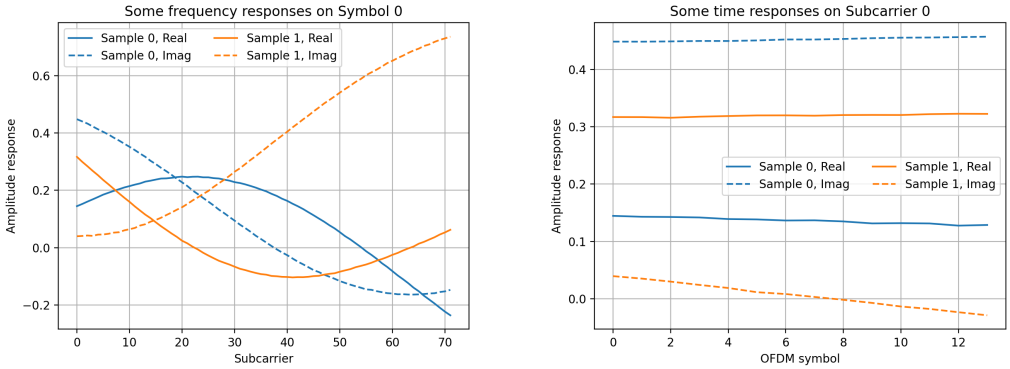

Figure 5.7.: Channel coefficient evolution examples on an OFDM frame. The channel parameters correspond to Table 5.5

The second scenario is a simulation of an OFDM frame going through a Rayleigh channel, using a Clarke-Jakes model based autocorrelation matrices [147, Chap-3]. N_S subcarriers, spaced by Δf, make up one OFDM symbol, and N_T of those are in a frame, leading to $N_D = N_S \times N_T$ ressource element (RE). N_P pilots per user can be inserted into this frame to help with equalisation, allowing for $N_{re} = N_D - 2 \times N_P$ data messages per frame. The transmitted constellations spans N_{cu} RE or channel uses, leading to a number of messages per frame of $N_{mf} = \frac{N_{re}}{N_{cu}}$ The pilots need to be sent on RE dedicated to one user to allow for an unambiguous channel estimation.

The resulting OFDM frame is then sent on a simulated channel with a carrier at f, with a delay spread of $\Delta\tau$, a velocity of v and the 3GPP TDL-A power delay profile. Two channel covariance matrices $\mathbf{R}_1, \mathbf{R}_2 \in \mathbb{C}^{N_D}$ are generated in an experiment and used to create the $\mathbf{H}_1, \mathbf{H}_2 \in \mathbb{C}^{N_S \times N_T}$ channel matrices at run time by sampling from $\mathcal{N}_{\mathscr{C}}(0, R)$. These matrices correspond to both the frequency and time correlations in the considered channel and are separable:

$$\mathbf{R}_1 = \mathbf{R}_{f_1} \otimes \mathbf{R}_{t_1} \tag{5.2}$$

where $\mathbf{R}_{f_1} \in \mathbb{C}^{N_S \times N_S}$ is the frequency correlation, and $\mathbf{R}_{t_1} \in \mathbb{C}^{N_T \times N_T}$ is the temporal component.

With $J_0(.)$ being the zero-order Bessel function of the first kind and cm/s the speed of light, the temporal correlation between the i^{th} and k^{th} OFDM symbols ($0 \leq i, k < N_T$) is given in [148] as:

$$[\mathbf{R}_t]_{i,k} = J_0\left(2\pi \frac{v}{c} f \frac{1}{\Delta f}(i-k)\right) \tag{5.3}$$

While the frequency component (this time between the i^{th} and k^{th} subcarriers ($0 \leq i, k < N_S$))is computed according to [147] as the discrete Fourier transform of the power delay profile of L normalised echoes of power S_l and delay τ_l :

$$\left[\mathbf{R}_f\right]_{i,k} = \sum_{l=1}^{L} S_l \exp\left(j2\pi\tau_l \Delta\tau \Delta f (i-k)\right) \tag{5.4}$$

The channel realisation are obtained in a flattened vector form with $\mathbf{h}_1 = vec(\mathbf{H}_1)$ by first sampling an uncorrelated Gaussian vector of the same length $\mathbf{w}_1 \sim \mathcal{N}_{\mathbb{C}}(0, \mathbf{I}_{N_D})$ and filtering it with the eigenvalue decomposition of the corresponding covariance matrix $\mathbf{R}_1 = \mathbf{U}_1 \mathbf{\Lambda}_1 \mathbf{U}_1^{\mathsf{H}}$ as:

$$\mathbf{h}_1 = \mathbf{U}_1 \mathbf{\Lambda}_1^{\frac{1}{2}} \mathbf{w}_1 \tag{5.5}$$

As mentioned above, $\mathbf{H}_1$ and $\mathbf{H}_2$ are normalised to an average power of 1 per realisation. This means that the instantaneous power for a given RE is not constrained to 1, so the resulting SNR will only be equal to the target SNR in average and fading can occur.

The main difference between the two scenarios is differential rotations. In the Gaussian channel, since $h_1 = h_2 = 1$, if the two constellations are sent with an angle $\Delta\theta_{TX}$, it will arrive at the receiver with the same angle. Even in the case of a Rayleigh broadcast channel, with $h_1 = h_2 \in \mathbb{C}$, the two constellations would rotate but of the same amount, maintaining their relative angle to $\Delta\theta_{RX} = \Delta\theta_{TX}$. In the case of the correlated OFDM scenario, since $h_1 \neq h_2 \in \mathbb{C}$, the two signals experience two different rotations θ_1 and θ_2.

$$\Delta\theta_{RX} = \Delta\theta_{TX} + (\theta_1 - \theta_2) \tag{5.6}$$

The same is also true with amplitude, but it is a less prominent effect. This can lead to situations where $\Delta\theta_{RX}$ leads to the superposition of symbols in the combined constellation expected by the receiver. Even if it has perfect CSI, thus knows θ_1 and θ_2, it cannot separate the ambiguous symbols, and suffers from a reduced performance. If the transmitters had full instantaneous CSI, they could adjust $\Delta\theta_{TX}$ to move it outside of the ambiguity zone, but it is a quite unrealistic assumption, and thus not considered in this model. So the correlated OFDM model, on top of the usual limitations it leads to, such as errors in channel estimation, or temporary increases in SNR due to fading, already encountered in P2P and OMA, introduces new adverse conditions in the NOMA setting.

5.4. Implementation

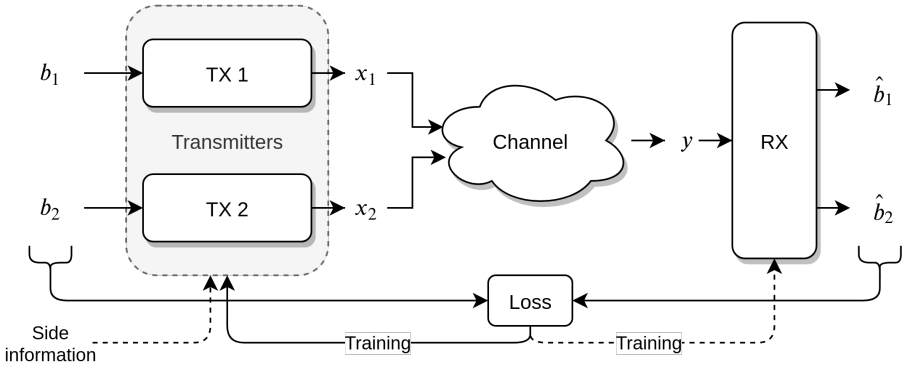

Figure 5.8.: The training framework

Fig. 5.8 describes the framework used here to train constellations: two bit streams b_1 and b_2 are generated and input to the transmitters. In the transmitters, N_{bpm} bits are assembled to form the message streams m_1 and m_2, and are then encoded to form the output signal

stream x_1 and x_2. These signal go through the channel described previously and is input to the receiver. Soft bit estimations $\hat{b}_1$ and $\hat{b}_2$ are then jointly computed. The original bit stream and its estimation is compared by the loss function and the resulting loss is fed back to the transmitter, and to the receiver in cases where it's learned, to update it.

The exact shape and size of the streams depends on the channel scenario. For convergence stability and computational efficiency on GPUs, the streams are managed in batches of B examples. In the Gaussian scenario, each example correspond to one constellation symbol. This training process is repeated for I iterations, allowing for convergence.

Table 5.2.: Tensor shapes given channel Scenario

Tensor	Shape (Gaussian)	Shape (OFDM)
$b_1, b_2, \hat{b}_1, \hat{b}_1$	$B \times N_{bpm}$	$B \times N_{mf} \times N_{bpm}$
x_1, x_2, y	$B \times N_{cu}$	$B \times N_S \times N_T$

5.4.1. Metrics

The basic metrics to compare the classical and learned approaches are BER and symbol error rates, that can be measured easily by outputting hard decisions from the network and the ML decoder.

A more interesting metric is the bitwise mutual information (BMI), which is an achievable rate for a bit-metric decoding (BMD) system such as the one studied here, as defined in [149].

$$R_{BMI} := H(X) - \sum_{i=1}^{N_{bpm}} H(B_i|Y) \tag{5.7}$$

$$\begin{aligned} H(B_i|Y) &= -\mathbb{E}_{y,b_i}\left[\log\left(p(b_i|y)\right]\right. \\ &= -\mathbb{E}_y\left[\mathbb{E}_{b_i}\left[\log\left(p(b_i|y)\right)\right]\right] \\ &= \mathbb{E}_y\left[H(B_i, B_i|y)\right] \end{aligned} \tag{5.8}$$

When doing Monte Carlo sampling with N examples, this can be estimated by:

$$H(B_i|Y) \approx -\frac{1}{N}\sum_{n=1}^{N} p(b_i = 1)\log\left(p(b_i = 1|y_n)\right) + p(b_i = 0)\log\left(p(b_i = 0|y_n)\right) \tag{5.9}$$

With $\hat{b}_i = p(b_i = 1|y_n)$, the output of the receiver. b_i is what is known to be transmitted so its distribution is fully known, and it can only take binary values so $p(b_i = 1) = b_i$, meaning that we can rewrite eq. 5.9 as:

$$H(B_i|Y) \approx -\frac{1}{N}\sum_{n=1}^{N} b_i \log\left(\hat{b}_i\right) + (1 - b_i)\log\left(1 - \hat{b}_i\right) \tag{5.10}$$

This corresponds to the standard binary CE loss function. When using it at the output of the receiver, from 5.7 and 5.10, the achievable rate per channel use for a transmission R_{TX} can

thus be measured as such:

$$R_{TX} = \frac{N_{bpm}}{N_{cu}}(1 - \mathcal{L}), \tag{5.11}$$

$\mathcal{L}$ being the cross entropy loss obtained after a number of examples.

These basic metrics are linked to only one transmitter. The vast majority of existing works in the domain of uplink NOMA with digital constellations use aggregate metrics to combine the two transmissions in one number, be it an average BER, a sum rate,... This approach allows for more convenient results, with only one dimensional curves to study. But it reduces the information that can be gained from these metrics by completely hiding the possible tradeoffs that can exist between the performance of the two users. Indeed, for a given sum-rate of 4 bits per channel use, one cannot distinguish between many cases: $R_1 = R_2 = 2\,\text{bit/cu}$, $R_1 = 4\,\text{bit/cu}$ and $R_2 = 0\,\text{bit/cu}$, and all the possible combinations in-between. Another aspect it that a possible operator may not look at maximising overall performance, but at satisfying some QoS elements, for instance making sure that $R_2 > 0.5\,\text{bit/cu}$. For these reasons, in this work, the metrics are not aggregated but presented separately, putting the possible pairs of rate on a 2D plot, tracing a *rate region*.

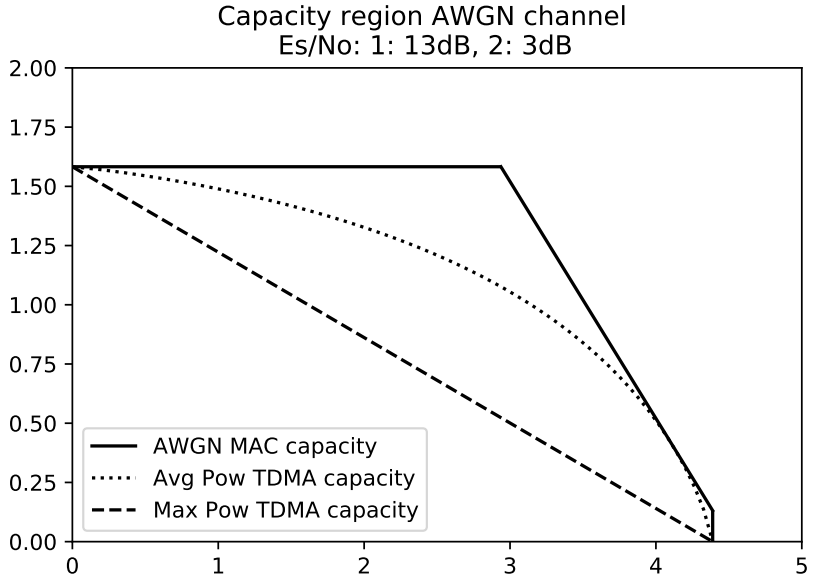

Figure 5.9.: Examples of capacity region, for an AWGN channel.

On the Gaussian channel, capacity bounds are known for both the TDMA and NOMA approaches [150, Chap. 15]. The NOMA or MAC capacity region is the combination of three different requirements: The rate for one given user cannot exceed the capacity that would be available if the user was transmitting alone, and the sum of the rates cannot exceed ce capacity of the sum of the powers:

$$R_1 \leq \log\left(1 + SNR_1\right) \tag{5.12}$$

$$R_2 \leq \log\left(1 + SNR_2\right) \tag{5.13}$$

$$R_1 + R_2 \leq \log\left(1 + SNR_1 + SNR_2\right) \tag{5.14}$$

The solid black line of Fig. 5.9 meets these three requirements. For the orthogonal methods, the user 1 transmits a fraction τ of the time, or, if considering FDMA, of the available frequencies. For the maximum power constrained TDMA, this leads to the following rate pair:

$$R_1 = \tau \log\left(1 + SNR_1\right) \tag{5.15}$$
$$R_2 = (1-\tau) \log\left(1 + SNR_2\right) \tag{5.16}$$

And for the average power constrained TDMA, the user can transmit at a power inversely proportional to the transmission frequency, leading to this rate pair:

$$R_1 = \tau \log\left(1 + \frac{SNR_1}{\tau}\right) \tag{5.17}$$
$$R_2 = (1-\tau) \log\left(1 + \frac{SNR_2}{(1-\tau)}\right) \tag{5.18}$$

When plotting all the rate pairs that meet these requirements, one can trace the capacity regions for the three approaches, as in Fig. 5.9. From this figure, there are zones where the NOMA approach is expected to lead to rate gains compared to the OMA approaches.

In the correlated OFDM scenario, an upper bound will be used for the TDMA, in the form of an achievability assuming full CSI and appropriate bit loading at the transmitter, but with constant transmission power. This is not feasible in practice but would allow the transmitter to take into account instantaneous fading effects in the channel, leading to this rate pair that will be estimated using Monte Carlo sampling:

$$R_{1_{inst}} = F \log\left(1 + h_1 \frac{SNR_1}{F}\right) \tag{5.19}$$
$$R_{2_{inst}} = F \log\left(1 + h_2 \frac{SNR_2}{F}\right) \tag{5.20}$$
$$R_1 = \mathbb{E}_{h_1}\left[R_{1_{inst}}\right] \tag{5.21}$$
$$R_2 = \mathbb{E}_{h_2}\left[R_{2_{inst}}\right] \tag{5.22}$$
$$\tag{5.23}$$

One of the main concerns of this works revolves around verifying if the proposed NOMA constellations compares favourably with the standard, simpler, and better understood orthogonal constellations, something that is rarely done in the literature. For this purpose, obtained rate regions will be plotted against achievable rates in TDMA, both with Average and Maximum power constraints. The question then arises as to which constellation scheme to use for a fair comparison. In a case where $N_{bpm} = 4$, $N_{cu} = 1$ in the NOMA system, is it fairer to compare with TDMA constellations of the same order, i.e. 16-QAM, or with constellations transmitting the same amount of bits as the sum of the two NOMA transmissions, i.e. 256-QAM? When considering standard M-QAM constellations options, the answer can depend on the channel conditions. As can be see in Fig. 5.10, at lower SNR levels (left plot), the 16-QAM gets more performance, while it is the opposite at higher SNR, where the lower order 16-QAM saturates. For this reason, later results will be compared against performance obtained by trained constellations able to more smoothly adapt to the noise level. This is

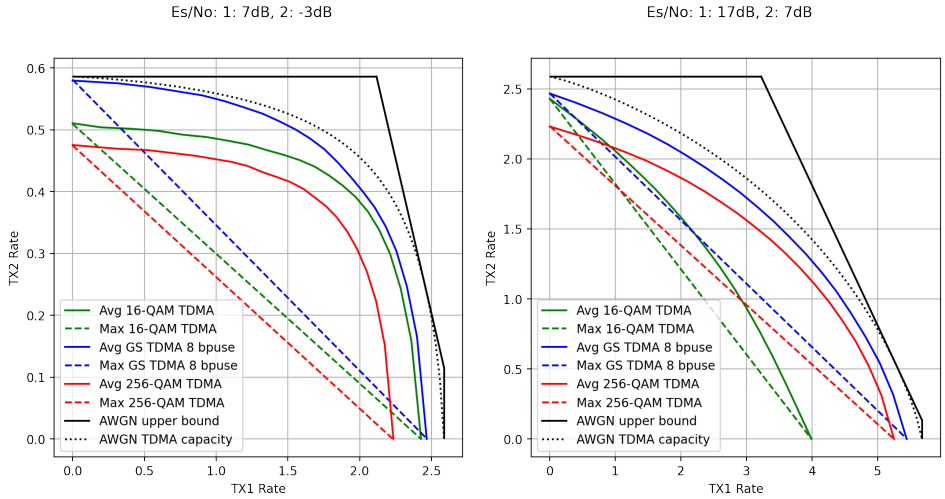

Figure 5.10.: Rate region comparisons between 16-QAM ($N_{bpm} = 4$), 256-QAM ($N_{bpm} = 8$), and learned constellations ($N_{bpm} = 8$). Gaussian scenario, $N_{cu} = 1$.

done by using the NN transmitter (described later) outputting only one constellation and optimised on a P2P transmission. Fig. 5.10 shows that this method consistently provides better performance than the standard constellation. Please note however that the rate region obtained with this GS method does not correspond to one unique constellation, but, especially with the Average power constraint, a different constellation pair for each point, tailored to the corresponding SNR level at that point.

5.4.2. Design of the joint loss function

As mentioned in Chapter 2, the selection of the proper loss function is paramount to the good performance of a trained machine leaning system. In the case at hand, the choice is lead by the reflections of the previous section: The CE loss function has been shown in 5.10 to be an estimator of $H(B_i|Y)$ so a network trained to minimise it will maximise the BMI, which is directly the metrics we are interested in. Optimising a bitwise metric instead of a symbol wise one also allows to optimise the bit labelling of the symbols. As was the case in the previous chapter, the industry default binary cross-entropy loss turns out to be the proper function to optimise.

Now that the basic loss function is chosen, a similar reflection to the one in the previous metrics section is present relative to the way it is handled. Two parallel transmissions from TX1 and TX2 lead to two different loss values $\mathcal{L}_1$ and $\mathcal{L}_2$.

$$\mathcal{L}_1 = -\sum_{n=1}^{N_{bpm}} b_{1_n} \cdot \log\left(\hat{b}_{1_n}\right) + (1 - b_{1_n}) \cdot \log\left(1 - \hat{b}_{1_n}\right) \tag{5.24}$$

$$\mathcal{L}_2 = -\sum_{n=1}^{N_{bpm}} b_{2_n} \cdot \log\left(\hat{b}_{2_n}\right) + (1 - b_{2_n}) \cdot \log\left(1 - \hat{b}_{2_n}\right) \tag{5.25}$$

$$\tag{5.26}$$

The eq. 5.24 corresponds to the loss for one example for the Gaussian scenario, the correlated OFDM scenario also involves averaging over N_{mf} messages per frame. These two values are independent and need to be joined to provide one unique value to train the system. This is especially true in the case that will be described later where a unique neural network designs both constellations at once, but even in a case with completely independent transmitters, a joint loss allows to explicitly target a specific area of the rate region without being constrained to an equal rate situation.

Let $\alpha \in [0, 1]$ be the parameter to direct the trade-off between the transmitter performance, with $\alpha = 1$ corresponding to full power given to TX1, $\alpha = 0$ to TX2, and $\alpha = 0.5$ to a balanced situation.

Two methods are proposed to combine $\mathcal{L}_1$ and $\mathcal{L}_2$ into $\mathcal{L}_{Total}$ according to α. First is a simple weighted average of the two that will be called *weighted loss*.

$$\mathcal{L}_{Total} = \alpha * \mathcal{L}_1 + (1 - \alpha) * \mathcal{L}_2 \tag{5.27}$$

Second is a projection on a unit vector of the desired slope in the rate domain that will be called *projection loss*. The slope could be directly tied to α, but to keep the usability requirements above, it is defined as follows:

$$slope : s = \begin{cases} 2(1-\alpha) & \text{if } \alpha \geq 0.5 \\ \dfrac{1}{2\alpha} & \text{otherwise} \end{cases} \tag{5.28}$$

The two vectors of the projection can then be defined: $\mathbf{u}$ the unit vector and $\mathbf{r}$ the rate vector.

$$\mathbf{u} = \begin{pmatrix} \frac{1}{\sqrt{1+s^2}} \\ \frac{s}{\sqrt{1+s^2}} \end{pmatrix} \mathbf{r} = \begin{pmatrix} R_1 \\ R_2 \end{pmatrix} \tag{5.29}$$

where R_1 and R_2 denote the achievable rates for TX1 and TX2 derived from eq. 5.11.

$$R_{1_{proj}} = \mathbf{u}_x \times \mathbf{u} \cdot \mathbf{r} \tag{5.30}$$

$$R_{2_{proj}} = \mathbf{u}_y \times \mathbf{u} \cdot \mathbf{r} \tag{5.31}$$

$$R_{projsum} = R_{1_{proj}} + R_{2_{proj}} \tag{5.32}$$

This gives the projected sum-rate that is desired to be maximised, but the training phase expects a function to be minimised so it needs to be inverted, and, to keep the loss within reasonable bounds and improve readability, it is compared to the Shannon capacity of Gaussian channel with the corresponding average SNR levels.

$$C_{sum} = \log_2 (1 + SNR_1 + SNR_2) \tag{5.33}$$

$$\mathcal{L}_{Total} = C_{sum} - R_{projsum} \tag{5.34}$$

This C_{sum} is only one of the constraints to the Shannon capacity, as described above, and the inclusion of the two other terms would provide a tighter bound to the possible sum-rate. But it would only add a constant to the loss function, and so not affect the gradient computation and the convergence of the system.

5.4.3. transmitter

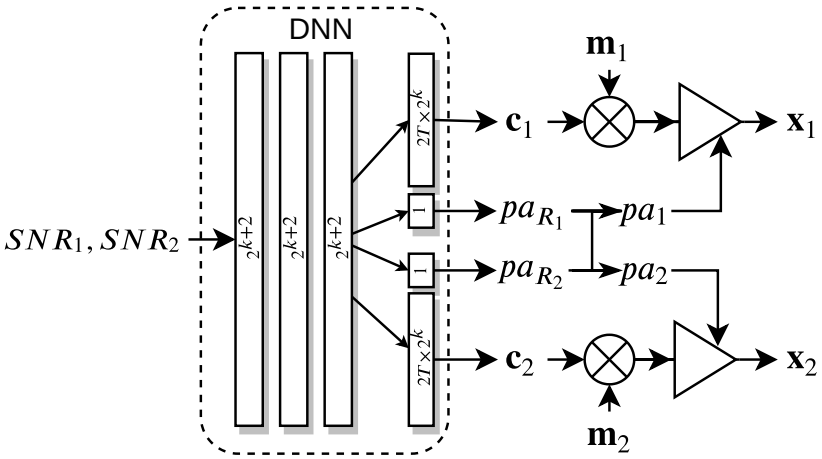

Figure 5.11.: Neural network transmitter. For the sake of readability, $k = N_{bpm}$ and $T = N_{cu}$.

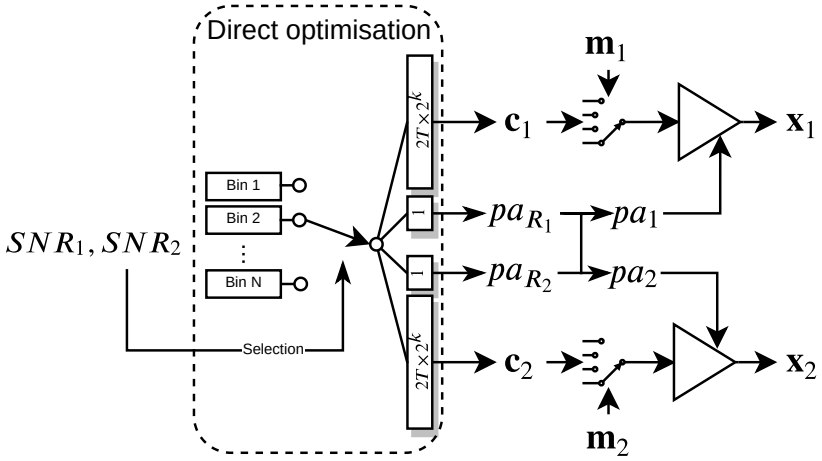

Figure 5.12.: Direct optimisation transmitter. For the sake of readability, $k = N_{bpm}$ and $T = N_{cu}$.

Two different constellation design transmitters are proposed, a neural network based, and a direct optimisation (DO) one, both being trained. They take as input the snr level of the users, and output a constellation and a power allocation request scalar $pa_R \in [0, 1]$ for each user, that can then be used to encode the messages to transmit $\mathbf{m}_1, \mathbf{m}_2 \in \left[0, 2^{N_{bpm}}\right[$. The constellations vectors are real valued with $2 \times 2^{N_{bpm}}$ element, with pairs of real values consisting of the real and imaginary components of the complex symbols to transmit.

Full constellation optimisation

The proposed transmitter designs output an entire constellation, this is in contrast to previous machine learning geometric shaping works, such as [107], [135], [137], [144] where the NN

takes the messages as input, with eventual side information such as the SNR level and outputs the corresponding symbol. In that case, the messages are encoded independently from each other and the resulting constellation can be extracted by feeding the possible messages one at a time to get all the possible symbols. This approach will be referred as independant symbols system (ISS). The choice to explicitly optimise the entire constellation at once in the present work, hereby called full constellation system (FCS), stems from both implementation and architecture simplicity aspects:

Implementation aspects: Since the system to train is expected to be able to adapt to several noise conditions, the SNR level is given as input to the network and training needs to provide examples from all the expected noise conditions. There are two possibilities to do that: either the SNR level is changed between each training iterations, meaning that all the examples in a batch share the same conditions (*batch level* channel conditions), or the noise level is varied from example to example (*example level* channel conditions). The first option is simpler to implement, but the second one leads to a smoother gradient descent, with each batch better representing the entire distribution of expected channel conditions,improving the optimisation performance. The second element is regarding normalisation. As describe in the system model, the transmitted encoded symbols x_1, x_2 have an average power to one, so the output of the encoding system needs to be normalised to meet this requirement. Again, two possibilities exist to do that: either the normalisation happens over an entire batch, relying on an assumption of equiprobable symbols and a big enough batch size to properly sample the expected output as a Monte Carlo simulation (*batch level* normalisation), or it happens separately on each example (*batch level* normalisation). The second option is not available to an ISS: it would cause all symbols to have the same power of one, leading to a PSK type constellation with all the points lying on the unit circle, not allowing other types of constellations.

An ISS implementing batch level normalisation and example level channel conditions risks encountering convergence issues. As illustrated in Fig. 5.13, the error rates present a dip at mid SNR but quickly rise back up at higher SNR. The location of the dip relates to the middle of the training range. The explanation of this phenomenon is the following: The normalisation of transmitted signals is done by batch, so that the average norm transmitted over a batch is 1. This means that the system can learn to give more power to low SNR symbols and less to high ones inside of a batch. So the receiver learns to work with a low power level for high SNR. But a test time, the SNR values are not mixed, so received power is always 1, for each SNR level. This means that, at test time and low noise level, the received power is not what was present at training, and the error rate increases.

The conclusion of this example is that an ISS is limited to batch level channel conditions which are less efficient for convergence. A FCS allows the use of both example level normalisation and channel condition, resolving this issue.

Architecture aspects: The constellation to be optimised contains a fixed amount of symbols whose value is not directly dependant on the message to be encoded. There is not a clear numerical relationship between two message-symbol pairs. The FCS explicitly enforces these characteristics by having the output be of the required shape of the entire constellation, and symbols (slices of the output vector) not numerically linked to each other. This is in contrast with the ISS where the system needs to learn the modulation order and the symbols

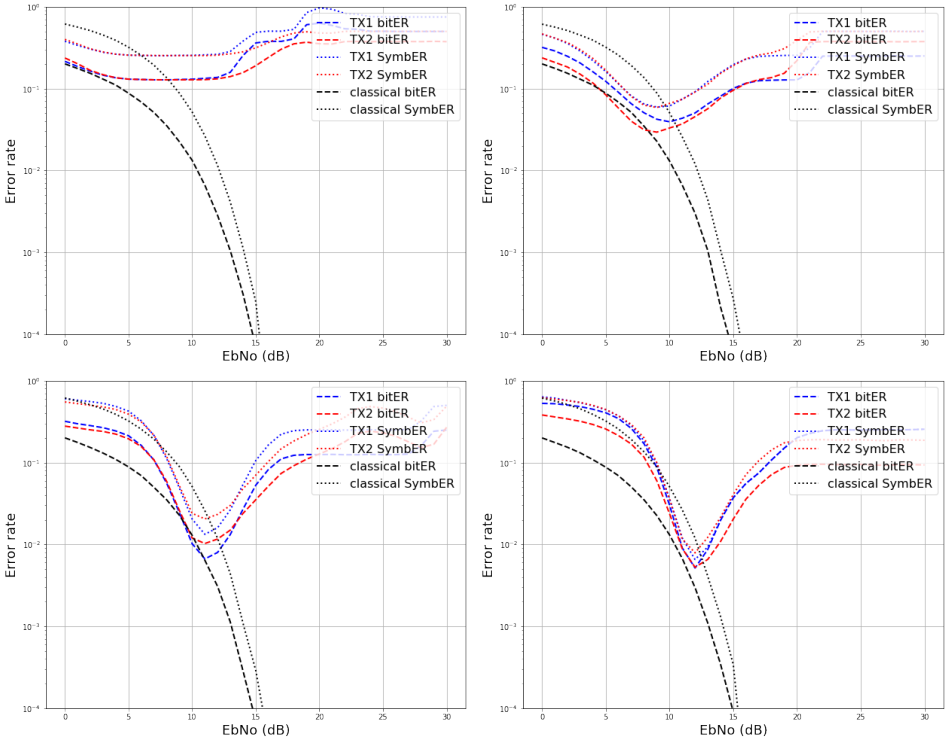

Figure 5.13.: Symbol and bit error rates for an ISS implementing batch level normalisation and example level channel conditions, compared with a 16-QAM TDMA transmission. Gaussian scenario, $P_1 = P_2$, $N_{bpm} = 2$, $N_{cu} = 1$. Training conditions are 1st: $\frac{Eb}{N0} \in [0, 10]$ dB, 2nd: $\frac{Eb}{N0} \in [0, 15]$ dB, 3rd: $\frac{Eb}{N0} \in [0, 25]$ dB, 4th: $\frac{Eb}{N0} \in [0, 30]$ dB

are presented as the continuous set of floating point real values, so it also has to learn to create decision regions inside that input set to separate the possible messages. This means that the ISS has more training to do to perform the same task. On top of this, the FCS outputting the entire instantaneous constellation during training, it simplifies the task of implementing the ML receiver that will be described in the next section.

Two transmitter designs

The two constellation design proposed transmitters are different in the way the output is derived from the input: The NN transmitter implements a DNN that operates on the SNR input, and uses three fully connected hidden layers, as shown in Fig. 5.11. The DO transmitter directly optimises the constellation matrices. It contains a pool of constellations and pa_R scalars, each affected to a SNR bin, that is selected according to the conditions in a deterministic manner.

As can be seen in Table 5.3, the DO transmitter is simpler, with less parameters to train, but the amount of bins is fixed whereas the NN transmitter allows for a continuous parsing ot the SNR space without discretisation. Narrow bins allow for a more precise tailoring of the constellation to a specific SNR but may require more bins to cover a desired SNR range, increasing training time and memory footprint.

The pa_R scalars allow for one user to reduce its transmitter power to give more room to

Table 5.3.: Number of parameters to train for the two transmitter types

Type	Layer	Parameters
DO	constellations	$2N_{bpm} \times 2^{N_{bpm}+1} \times bins$
	pa_R	$2 \times bins$
NN	Input	$2 \times 2^{N_{bpm}+2}$
	Hidden 1	$(2^{N_{bpm}+2}+1) \times 2^{N_{bpm}+2}$
	Hidden 2	$(2^{N_{bpm}+2}+1) \times 2^{N_{bpm}+2}$
	constellations	$2N_{bpm} \times (2^{N_{bpm}+2}+1) \times 2^{N_{bpm}+1}$
	pa_R	$2 \times (2^{N_{bpm}+2}+1) \times 2$

the other device's constellation. The pa_{R1} and pa_{R2} requests are transformed into used power allocation scalars $pa_1, pa_2 \in \left[0, 2^k\right]$ as follows:

$$pa_1 = \frac{pa_{R1}}{\max pa_{R1}, pa_{R2}}, \quad pa_2 = \frac{pa_{R2}}{\max pa_{R1}, pa_{R2}} \tag{5.35}$$

The square root of these pa values is then used to scale the encoded symbols. This balancing eliminates the possibility of having both users transmitting with reduced power, simplifying the training task.

The NN transmitter is implemented as a single neural network operating for both devices at the same time. A real implementation scenario would not have the two devices co-located or close enough to actually share a single neural network. A reasonable approach would be to have the NN hosted by the base station, at least for training, sending regular updated constellations to the devices, and in production, the same architecture could be kept, or the NNs duplicated and sent to the two devices.

Classical methods call for a random initialisation of weights in a neural network. This technique was found to lead to difficult convergence and reduce performance. To alleviate this issue and jump-start the transmitters, they are initialised to output a classical constellation at the beginning of the training, providing a first working step. The DO system sees its bin matrices filled with the corresponding classical constellation and the NN system sees 300 iterations where it is pre-trained to output said constellation. The starting constellation is a M-QAM of order $\frac{N_{bpm}}{N_{cu}}$.

Even with this initialisation, the resulting performance is somewhat random, with some dispersion in the resulting rate pairs after training of multiple transmitters, even with the exact same parameters. To work with this limitation, in the result section, multiple transmitters are trained for each α parameter, and it is repeated for 21 evenly spaced values of α between 0 and 1. The trained transmitters are all evaluated and the ones on the Pareto envelop of the rate region are selected.

5.4.4. receiver

The focus of this work is on optimising constellation shaping at the transmitter, but the system still needs a receiver to decode the transmitted signal and compute a loss from the estimated bit. Two receivers are used for this. The first one called *MLRX* is a classical, non learned maximum likelihood decoder, that uses knowledge of the current constellation made possible by the FCS architecture to decode the signal and outputs a log-likelihood ratio (LLR) for each

possibly transmitted bit from [127] (this paper implements it for a P2P channel).

Let $\mathbf{C}_{comb} = \left(\mathbf{r}_0, \ldots, \mathbf{r}_k, \ldots, \mathbf{r}_{2N_{bpm}}\right)$ be the combined constellation expected at the receiver, with each combined symbol $\mathbf{r}_k$ a vector of length $2N_{cu}$ real channel uses, coding for a vector of $2N_{bpm}$ bits $\mathbf{b}$. If $\mathbf{C}_1 = \left(\mathbf{x}_{1,0}, \ldots, \mathbf{x}_{1,N_{bpm}}\right), \mathbf{C}_2 = \left(\mathbf{x}_{2,0}, \ldots, \mathbf{x}_{2,N_{bpm}}\right)$ are the constellations actually transmitted,

$$\mathbf{C}_{comb_k} = P_1 \mathbf{C}_{1_{k\%N_{bpm}}} + P_2 \mathbf{C}_{2_{k\div N_{bpm}}} \tag{5.36}$$

where $\div$ is the Euclidean division and % the modulo operator, resulting in the sum of all possible transmitted symbols combinations.

In the Gaussian scenario, the LLR for the i^{th} bit, for the received signal $\mathbf{y}$ of $2N_{cu}$ real channel uses is computed as follows:

$$LLR(\mathbf{y}, i) = \ln\left(\frac{\sum_{\mathbf{r}_k|\mathbf{b}_i=1} \exp\left(-\frac{||\mathbf{y}-\mathbf{r}_k||^2}{\sigma^2}\right)}{\sum_{\mathbf{r}_k|\mathbf{b}_i=0} \exp\left(-\frac{||\mathbf{y}-\mathbf{r}_k||^2}{\sigma^2}\right)}\right) \tag{5.37}$$

where σ is the noise variance, and $\sigma = 1$, as defined in the system model. The first half of the bits correspond to those transmitted by TX1, and the second half to TX2.

For the correlated OFDM scenario, the receiver first performs a LMMSE channel estimation before the ML decoding using the knowledge of the channel covariance matrices $\mathbf{R}_1, \mathbf{R}_2$. To allow estimation, N_P pilots per user are inserted into the OFDM frame. This leads to the pilot matrices $\mathbf{P}_1, \mathbf{P}_2, \in \mathbb{C}^{N_S \times N_T}$, with $\mathbf{P}_{i,j,k} = 1$ if the i^{th} user has a pilot on the RE of the j^{th} subcarrier and k^{th} OFDM symbol, and $\mathbf{P}_{i,j,k} = 1$ otherwise. If $\mathbf{P}_{1,j,k} = 1$, $\mathbf{P}_{2,j,k} = 0$. The pilots are on isolated RE from each other. These pilot matrices will be handled in vectorised form with $\mathbf{p}_1 = \mathbf{P_1}, \mathbf{p}_2 = \mathbf{P_2}$ and $\mathbf{\Pi}_1, \mathbf{\Pi}_2$ the $N_P \times N_D$ matrices to selectively filter the pilots from the data RE.

For TX1, the channel estimation is computed as follows:

$$\hat{\mathbf{h}}_1 = \mathbf{R}_1 \, diag\left(\mathbf{p}_1\right)^H \mathbf{\Pi}_1^H (\mathbf{\Pi}_1 (diag\left(\mathbf{p}_1\right) \mathbf{R}_1 \, diag\left(\mathbf{p}_1\right)^H + \sigma^2 \mathbf{I}_n) \mathbf{\Pi}_1^H)^{-1} \mathbf{y}_{p1} \tag{5.38}$$

$$\mathbf{y}_{p1} = \mathbf{\Pi}_1 (diag\left(\mathbf{p}_1\right) \mathbf{h}_1 + \mathbf{n}) \tag{5.39}$$

where $\mathbf{y}_{p1}$ is the vector of pilot symbols for that transmitter and $\mathbf{n}$ the noise vector.

The channel estimation is not perfect, leading to the channel estimation errors $\tilde{\mathbf{h}}_1 = \mathbf{h}_1 - \tilde{\mathbf{h}}_1, \tilde{\mathbf{h}}_2 = \mathbf{h}_2 - \tilde{\mathbf{h}}_2$. These errors have correlation matrices $\tilde{\mathbf{R}}_1, \tilde{\mathbf{R}}_2 \in \mathbb{C}^{N_D \times N_D}$ as,

$$\tilde{\mathbf{R}}_1 = \mathbb{E}\left[\tilde{\mathbf{h}}_1 \tilde{\mathbf{h}}_1^H\right] \tag{5.40}$$

$$= \mathbf{R}_1 - \mathbf{R}_1 \, diag\left(\mathbf{p}_1\right)^H \mathbf{\Pi}_1^H (\mathbf{\Pi}_1 (diag\left(\mathbf{p}_1\right) \mathbf{R}_1 \, diag\left(\mathbf{p}_1\right)^H + \sigma^2 \mathbf{I}_n) \mathbf{\Pi}_1^H)^{-1} \mathbf{\Pi}_1 \, diag\left(\mathbf{p}_1\right) \mathbf{R}_1 \tag{5.41}$$

As was the case for the Gaussian scenario, an expected combined constellation is computed at the receiver. For a constellation starting on the l^{th} received RE:

$$\mathbf{C}_{comb_k} = \left[\hat{\mathbf{h}}_{1,l} \ldots \hat{\mathbf{h}}_{1,l+N_{cu}}\right] \mathbf{C}_{1_{k\%N_{bpm}}} + \left[\hat{\mathbf{h}}_{2,l} \ldots \hat{\mathbf{h}}_{2,l+N_{cu}}\right] \mathbf{C}_{2_{k\div N_{bpm}}} \tag{5.42}$$

Leading to the following LLR computation, where $\tilde{\sigma}_l^2 = \tilde{\mathbf{R}}_1 + \tilde{\mathbf{R}}_2 + \sigma^2$ is the sum of the

noise and the expected error from channel estimation.

$$LLR(\mathbf{y}_l, i) = \ln\left(\frac{\sum_{\mathbf{r}_k|\mathbf{b}_i=1} \exp\left(-\frac{||\mathbf{y}_l-\mathbf{r}_k||^2}{\tilde{\sigma}_l^2}\right)}{\sum_{\mathbf{r}_k|\mathbf{b}_i=0} \exp\left(-\frac{||\mathbf{y}_l-\mathbf{r}_k||^2}{\tilde{\sigma}_l^2}\right)}\right) \tag{5.43}$$

The value of $\tilde{\sigma}_l^2$ is tied to a specific RE but used for the decoding of the entire symbol. It could be adapted to constellations spanning multiple channel uses as long as it can be considered stable $N_{cu} = 1$ when using this receiver in this scenario.

The desired output of our receiver to compute the loss and the achievable rate metric is $\hat{b}_i = p(b_i = 1|y)$, not $LLR(y, i)$. A sigmoid operation is needed to obtain it:

$$\begin{aligned}
\text{sigmoid}\, LLR(y,i) &:= \frac{\exp\left(LLR(y,i)\right)}{\exp\left(LLR(y,i)\right)+1} \\
&= \frac{\exp\left(\ln\left(\frac{p(b_i=1|y)}{p(b_i=0|y)}\right)\right)}{\exp\left(\ln\left(\frac{p(b_i=1|y)}{p(b_i=0|y)}\right)\right)+1} \\
&= \frac{\frac{p(b_i=1|y)}{p(b_i=0|y)}}{\frac{p(b_i=1|y)}{p(b_i=0|y)}+1} \\
&= \frac{p(b_i=1|y)}{p(b_i=1|y)+p(b_i=0|y)} \\
&= \frac{p(b_i=1|y)}{p(b_i=1|y)+(1-p(b_i=1|y))} \\
&= p(b_i=1|y) = \hat{b}_i
\end{aligned} \tag{5.44}$$

Table 5.4.: Detail of the ResNet receiver architecture

Layer	Channels	Kernel size	Dilation rate
Input Conv2D	128	(3,3)	(1,1)
ResNet 1	128	(7,7)	(7,2)
ResNet 2	128	(7,7)	(7,1)
ResNet 3	128	(7,5)	(1,2)
ResNet 4	128	(5,3)	(1,1)
ResNet 5	128	(3,3)	(1,1)
ResNet 6	128	(3,3)	(1,1)
ResNet 7	128	(3,3)	(1,1)
Output	$2 * \frac{N_{bpm}}{N_{cu}}$	(1,1)	(1,1)

The second receiver type, the *DLRX*, only used in the correlated OFDM scenario, is a fully 2D convolutional residual network. It operates on an entire frame at a time, so its input is a tensor of size $N_T \times N_S \times 2$ (two channels for the I and Q components) and its output is a tensor of size $N_T \times N_S \times 2 * \frac{N_{bpm}}{N_{cu}}$, the matrix of bits corresponding to each resource element. The architecture is detailed in Table 5.4. The first and last layers are single 2D convolution

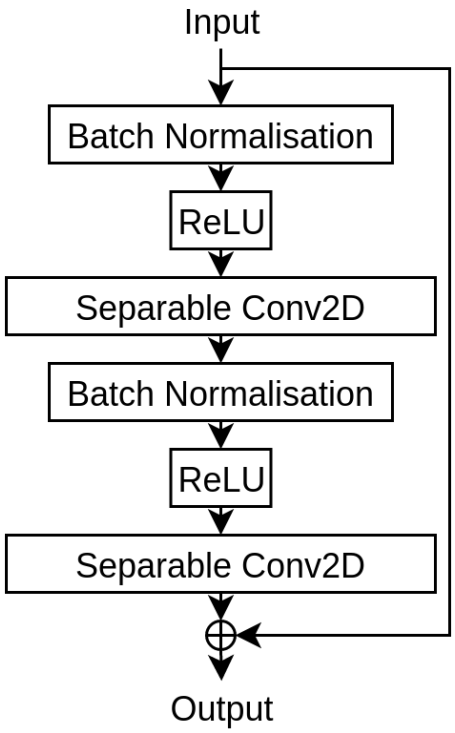

Figure 5.14.: Detail of the residual element for the fully convolutional receiver.

layers, while the hidden layers are organised in residual modules described in Fig. 5.14. The number of channel, kernel size, and dilation rate presented in the table are used in the "Separable Conv2D" layers of the residual element. The convolutions are given a 2D kernel, but they operate on a 3D tensor of size $N_T \times N_S \times \mathit{Channels}$ with the number of input channels being either 128 for the hidden layers, or two for the first layer. When this is the case, a separate 2D filter is used on each channel, and the output of these are summed. This corresponds to having a 3D kernel, the third dimension of which is fixed to being the number of input channels. In P2P, this type of architecture has been shown to allow for an operation without explicit pilots in the frame [127]. In that case, the learned constellations each contain a small amount of asymmetry, allowing for channel estimation. Such an operation allows the increase of data rate in a frame by reusing the resource elements freed by the elimination of explicit pilots. Attempts have been made to replicate this in the current MAC setting, but the increased complexity of training the transmitter and receiver at the same time lead to convergence issues and poor performance. For this reason, training with the DLRX is done in two consecutive steps: first the constellation is trained in a transmission chain containing a MLRX, with explicit pilots. After convergence of the transmitter, the MLRX is switched for the DLRX, and it is trained, keeping the transmitter fixed. The two receiver types can also be used in a P2P configuration for a fair comparison with TDMA methods. They also both compute outputs for RE with pilots, but they are ignored for the metrics computation.

5.5. Gaussian Channel results

5.5.1. Comparison with state of the art

To the best of our knowledge, there is no previous work studying geometric shaping for a 2-user MAC in a Rayleigh fading scenario, but there are some that do it in a Gaussian channel. The closest to the studied system is from Liu *et al* [134], that already compares to and outperforms [141] that compares to and outperforms [129]. For this reason, the present section will only compare results with this paper.

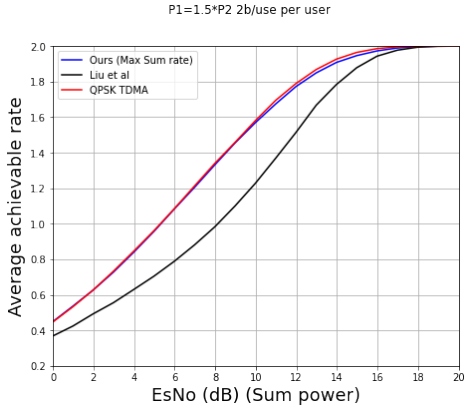

Figure 5.15.: Comparison between maximum average rates achievable by the constellation proposed by Liu et al [134]. Gaussian Scenario, $N_{bpm} = 2, N_{cu} = 1$, NN transmitter, compared with same order constellation QPSK TDMA (average power constraint)

Fig. 5.15 builds from the metrics studied in [134] and shows the evolution of the average achievable rate between the two users, with SNR. The SNR in this case is based on the total transmitted power (sum of the powers of the two transmitted signals), with $P_1 = 1.5P_2$ or $P_1 = P_2 + 1.761$ dB. The constellation building method described in the paper is used in our framework to directly compare with our method by computing the same achievable rate metrics. To compute the curve for our proposed method, multiple transmitters are trained for various α values as described in section 5.4.3 and at each SNR, the constellation leading to the maximum sum rate is retained. It can be seen from the figure that our proposed method outperforms the previous approach, even in this more limited metric, but it is on par with a traditional QPSK average power constrained TDMA.

In Fig.5.16, the comparison is widened to the entire rate region and to higher performing TDMA constellations. The first element to note is the fact that the constellation from [134] leads to only one point because the method provides only one constellation per SNR pair. It is also interesting to see that, although it allows for less rate than the average power TDMA, it does outperform the more realistic maximum power TDMA. Overall, it can be seen from this figure that our proposed method leads to both better performance and a more complete coverage of the rate region than our predecessor. But it does not provide better rates than the simple TDMA for reasons that will be discussed later.

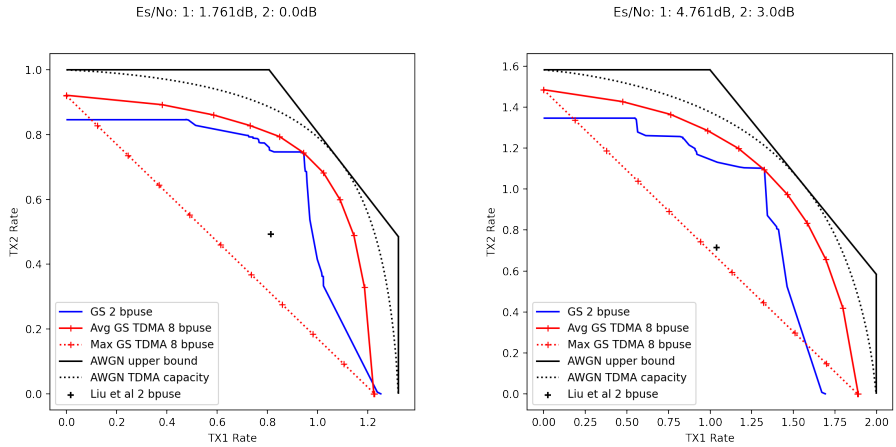

Figure 5.16.: Comparison with the performance of the constellation proposed by Liu et al [134]. Gaussian Scenario, NN transmitter, $N_{bpm} = 2, N_{cu} = 1$, compared with 8-GS TDMA transmissions for two SNR levels

5.5.2. Neural encoder vs direct constellation optimisation

In this chapter, two competing transmitter designs are proposed. This part attempts to compare the performances that can be obtained from these two. Fig. 5.17 provides rate region comparisons of the two approaches at various SNR levels with a constant power difference $P_1 = P_2 + 10\,\text{dB}$. The NN transmitters are trained to perform with this constant power difference and $P_2 \in [-10\,\text{dB}, 25\,\text{dB}]$. The DO transmitters have bins centred on integer SNR levels $\pm 0.5\,\text{dB}$, the narrow bins leading to an more precise tailoring of the constellations to the channel conditions and thus better performance. The NN transmitter is over-performing the DO in the majority of cases, especially at higher SNR levels and relative to TX2, the transmitter with the worst channel conditions. Please note in this comparison that the rate regions are not exactly continuous, they are comprised of a finite amount of constellations, marked in this figure by plusses. Intermediary rates could be obtained by alternating between adjacent points in the rate envelop. This is also true for the other rate regions in this section, but they are highlighted here to show one other difference between the two transmitter types: The NN transmitter shows a more continuous sweep of the rate region, allowing for more precise trade-off without having to resort to the method describe above. The present results lead us to select the NN transmitter type for the rest of this chapter, but the DO type could still be of use in cases where memory is limited, due to it having a greatly reduced number of parameters to train. In the case presented here, with $N_{bpm} = 4, N_{cu} = 1$, the NN transmitters have 12868 parameters, and the DO 2376 with 36 bins between $-10\,\text{dB}$ and $25\,\text{dB}$ (computed according to Table 5.3).

5.5.3. Rate region characteristics

The results comparing with the constellations from [134], having a small amount of asymmetry did not show performance improvements over the average power TDMA. Fig. 5.18 and Fig. 5.19 show the evolution of the rate regions when the power difference is increased, maintaining P_2 at a constant level. In Fig. 5.18, low level of asymmetry do not allow the GS

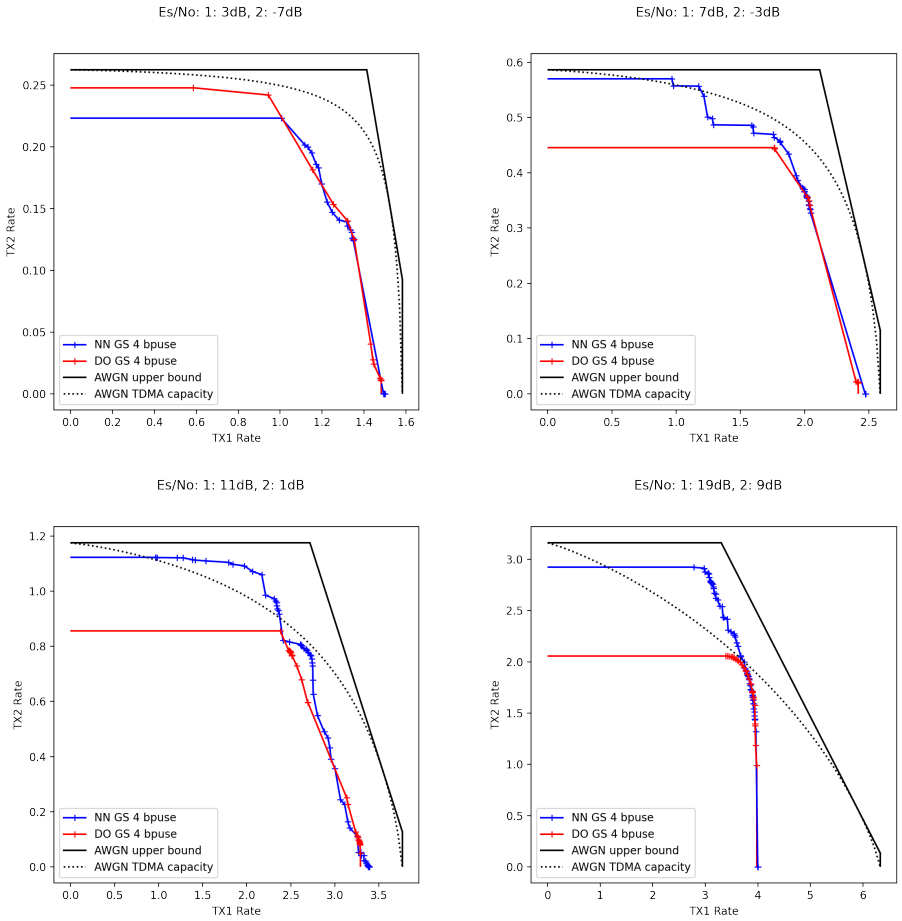

Figure 5.17.: Rate regions comparison between the two transmitter types. Gaussian Scenario, $N_{bpm} = 4, N_{cu} = 1$, compared with 8bit GS TDMA transmission. The DO transmitter has bins 1 dB wide, centred on the plots SNR.

regions to significantly go above the average power TDMA (it still is above the maximum power TDMA), while Fig. 5.19 show performance gains above 6 dB difference. Above that, the proposed GS NOMA constellations can significantly outperform the TDMA, even going outside of the TDMA capacity , especially at higher power differences. It can be concluded from this result that the NOMA approach is more pertinent at high asymmetry level, this was already pointed by theoretical results, notably from the increased gap between the TDMA and MAC capacity regions, but was not studied by the literature on geometric shaping for NOMA. For this reason also, the following results have a 10 dB power difference between the two transmitters.

This power difference is kept constant in Fig. 5.20 while the SNR level is increased. At very high noise level for TX2, the proposed constellations are close, but below the TDMA, but it quickly shows gains and step outside of the capacity region of the TDMA. At higher SNR, the constellations start saturating. Since $N_{bpm} = 4$, no more than four bits can be transmitted at once, limiting the possible rates for this particular constellation order. To go above, N_{bpm} can be increased.

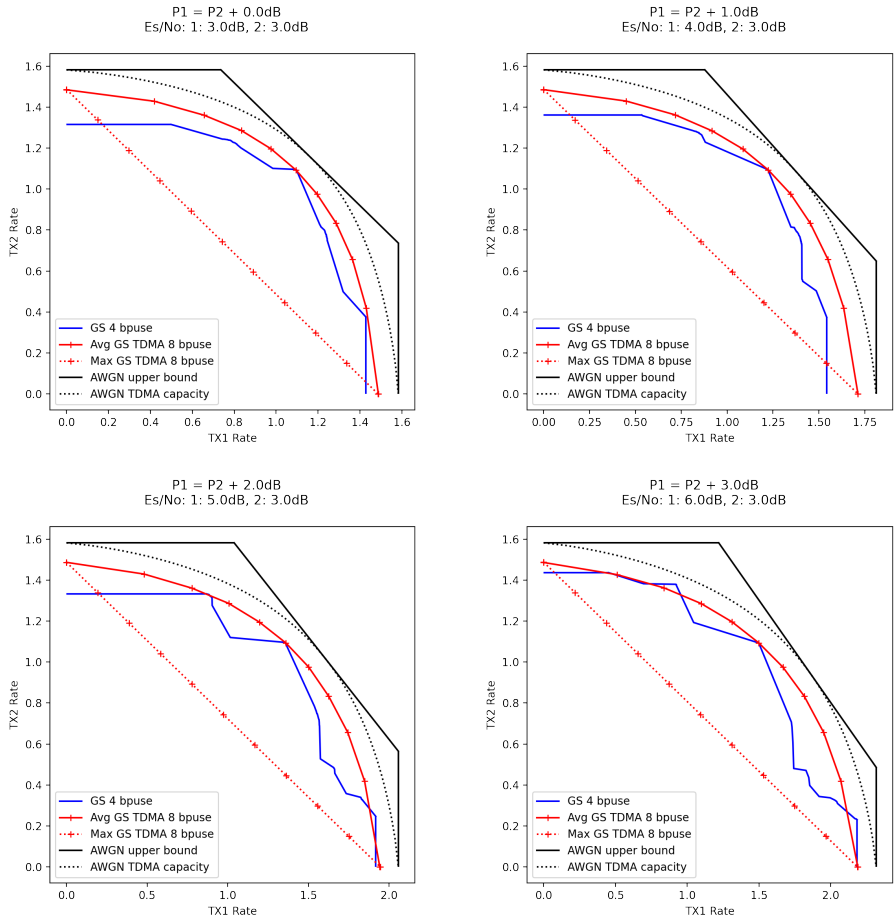

Figure 5.18.: Rate regions with low level of power asymmetry between the two users. Gaussian Scenario, NN transmitter, $N_{bpm} = 4$, $N_{cu} = 1$, compared with 8bit GS TDMA transmissions

5.5.4. Obtained constellations

Figures 5.21, 5.22, and 5.23, show examples of constellations at three zones of the rate region for $P_1 = P_2 + 10\,\text{dB}$, $P_2 = 3\,\text{dB}$ shown in Fig. 5.20. The top part provides the combined and isolated constellations as they are expected at the receiver, scaled by P_1 and P_2, while the bottom part displays the decision regions at the MLRX decoder for each bit of each user. The red color is associated with a bit to 0, blue to 1, and white and paler colors shows zones of ambiguity, meaning that the decoder can make erroneous decisions for symbols in the zone. Fig. 5.21 corresponds to the top part of the rate region, with more attention given to the rate of TX2. The constellation for this user is very close to a 16-QAM with all symbols equally spaced, while TX1 has a constellation with superposed symbols, leading to some ambiguity for bits 1 and 4. This reduction in symbols gives more space to decode TX2. The power allocation here is 1 for both transmitters, TX1 constellation has symbol spaced far enough apart to fit the entire TX2 constellation.

Fig. 5.22 is from a point in the middle of the rate region, with similar emphasis for both

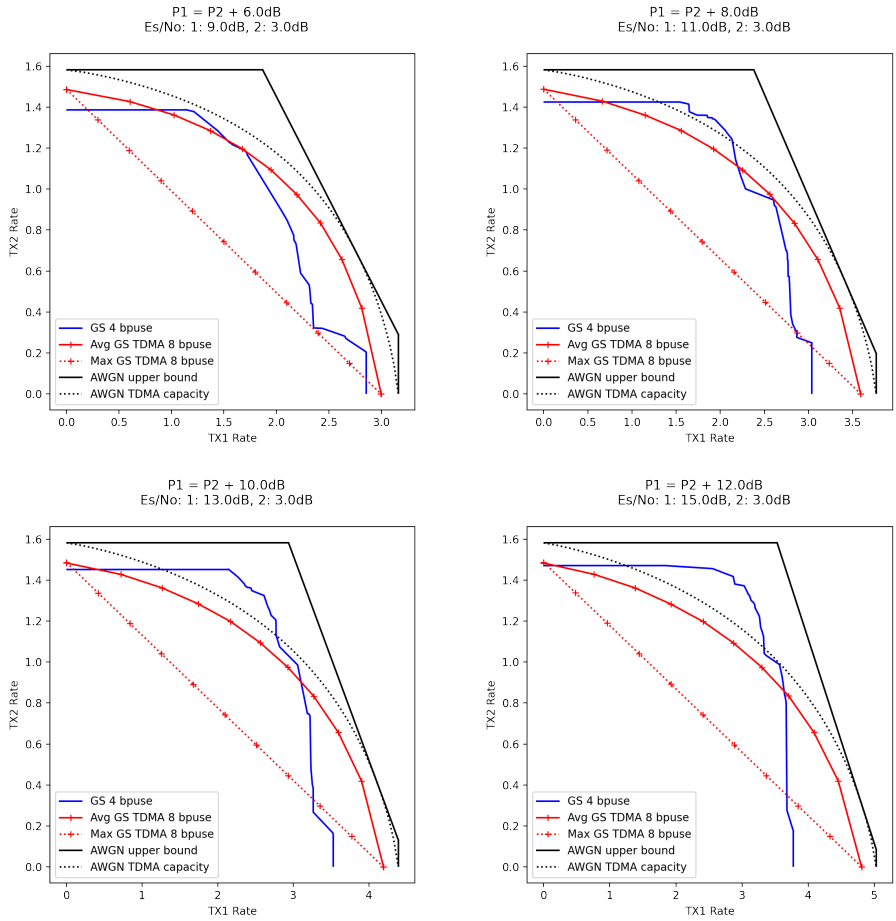

Figure 5.19.: Rate regions with high level of power asymmetry between the two users. Gaussian scenario, NN transmitter, $N_{bpm} = 4$, $N_{cu} = 1$, compared with 8bit GS TDMA transmissions

devices, so TX2 cannot fully expand into a 16-QAM with some symbols closer to each others, leading to more ambiguity in the decoding. The constellation of user 1 leads to a more compact and regular tiling of the space in the combined constellation, without the gaps that could be seen in the previous figure, but it still has superposed symbols.

Finally, Fig. 5.23 is from the very bottom of the rate region, with everything given to TX1. There, the corresponding constellation can fully expand into 16 equally spaced symbols. It is interesting to note here that the power allocation for TX2 is still very close to 1, and that the constellation is also widely spaced. At first glance, one would expect that this would degrade the performance of both users, but the decision regions show that the bits for TX1 are well defined, while the symbols of TX2 are very close to each other, leading to degraded bit decoding, as can be seen from the pale colours of the decoding regions.

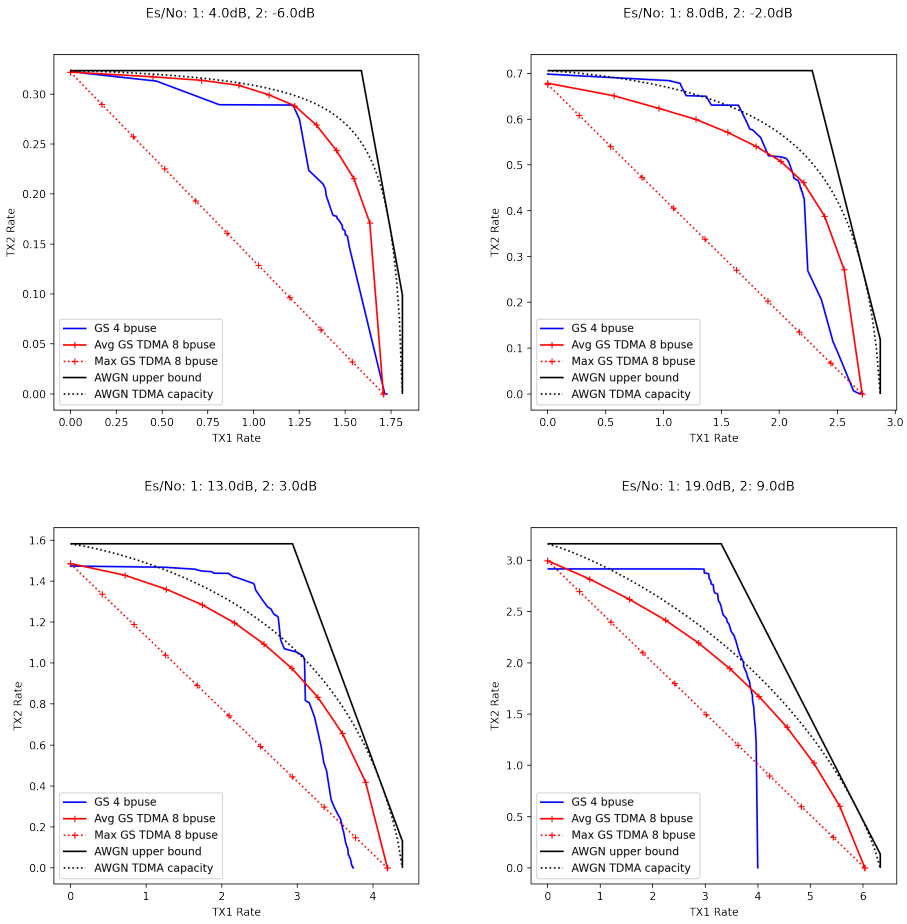

Figure 5.20.: Rate regions with $P_1 = P_2 + 10\,\mathrm{dB}$. Gaussian scenario, NN transmitter, $N_{bpm} = 4$, $N_{cu} = 1$, compared with 8bit GS TDMA transmissions

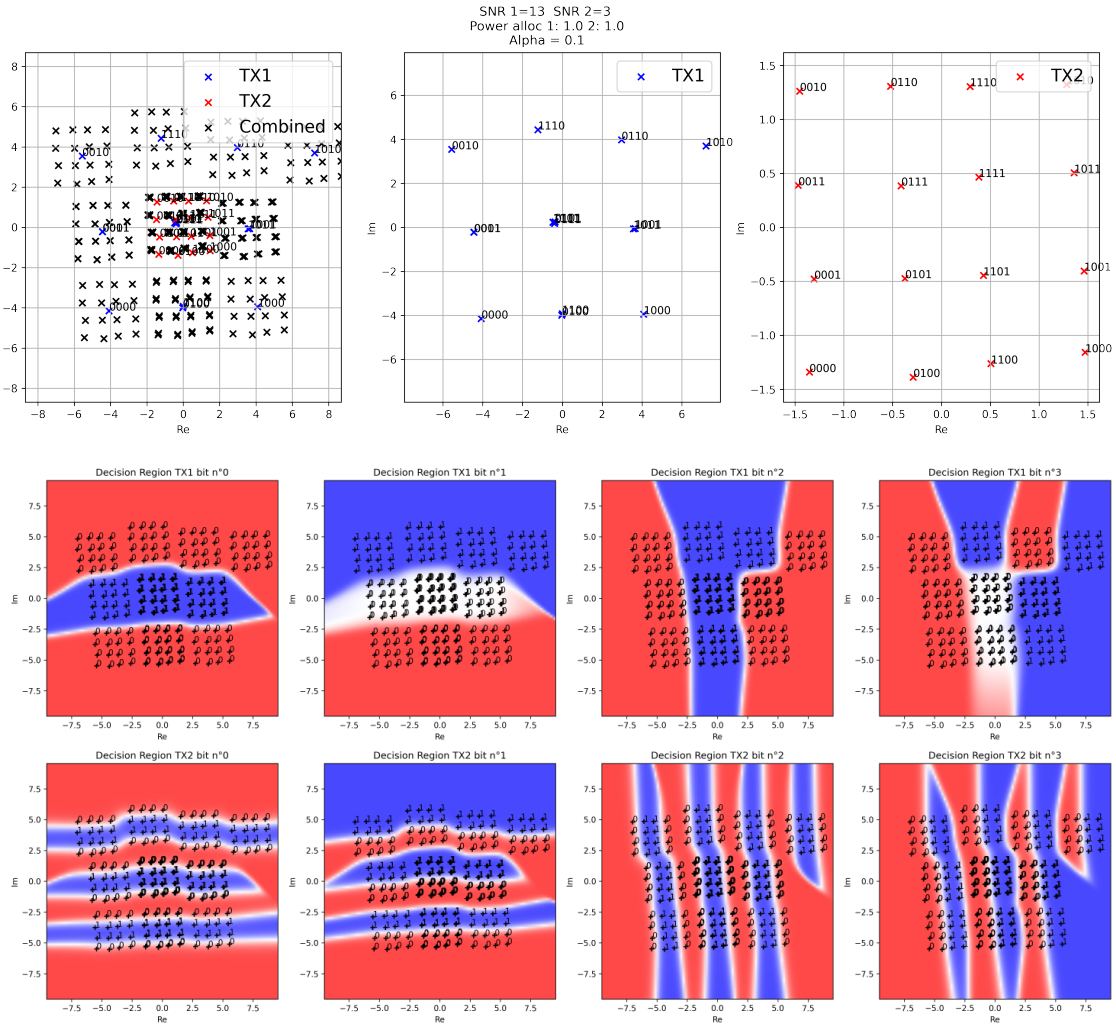

Figure 5.21.: Constellations obtained for $P_1 = P_2 + 10\,\text{dB}$, $P_2 = 3\,\text{dB}$ with constellation plot and corresponding decision regions. Gaussian Scenario, NN transmitter, $N_{bpm} = 4$, $N_{cu} = 1$. Corresponding rate: $R_1 = 1.9$, $R_2 = 1.44$

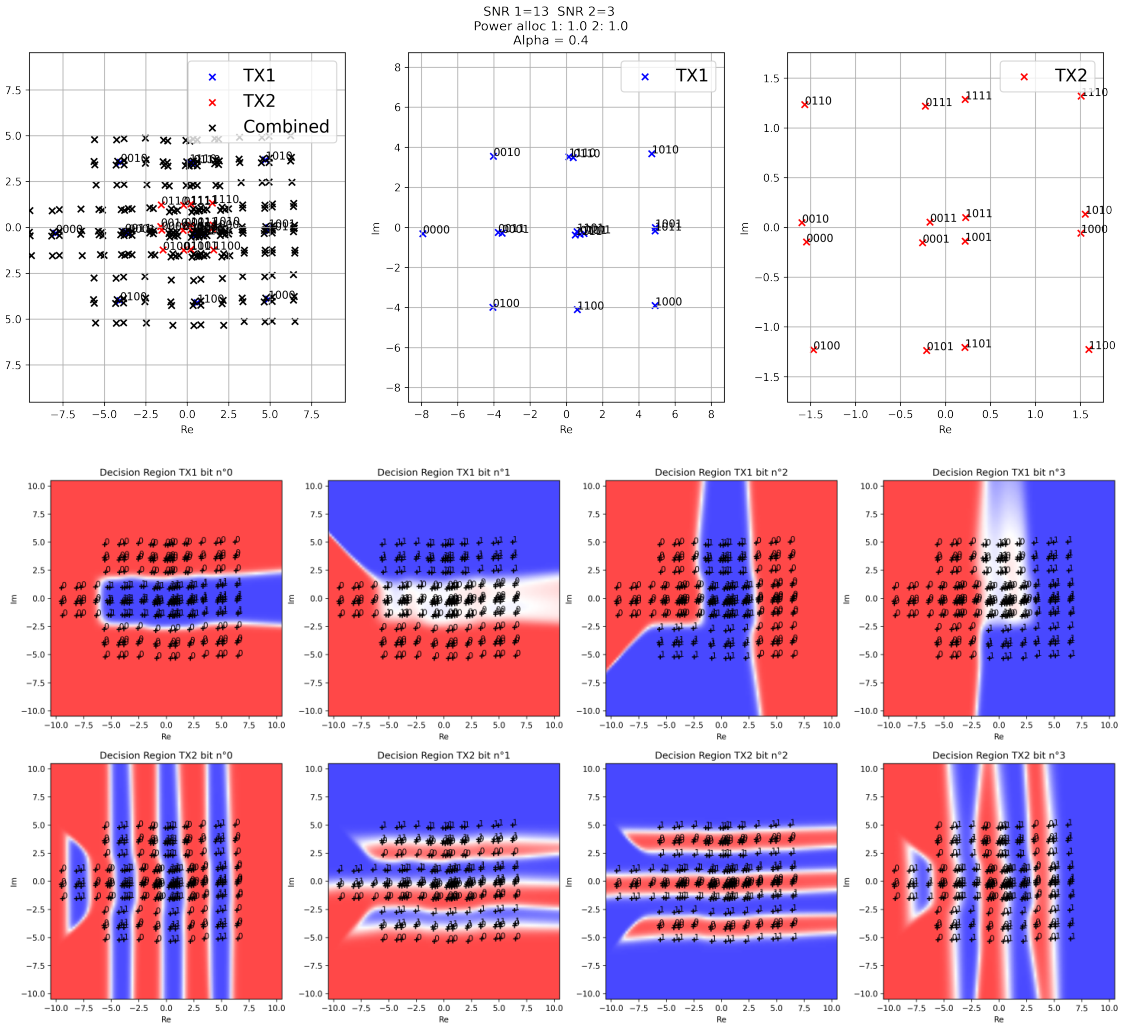

Figure 5.22.: Constellations obtained for $P_1 = P_2 + 10\,\text{dB}$, $P_2 = 3\,\text{dB}$ with constellation plot and corresponding decision regions. Gaussian Scenario, NN transmitter, $N_{bpm} = 4$, $N_{cu} = 1$. Corresponding rate: $R_1 = 2.78$, $R_2 = 1.11$

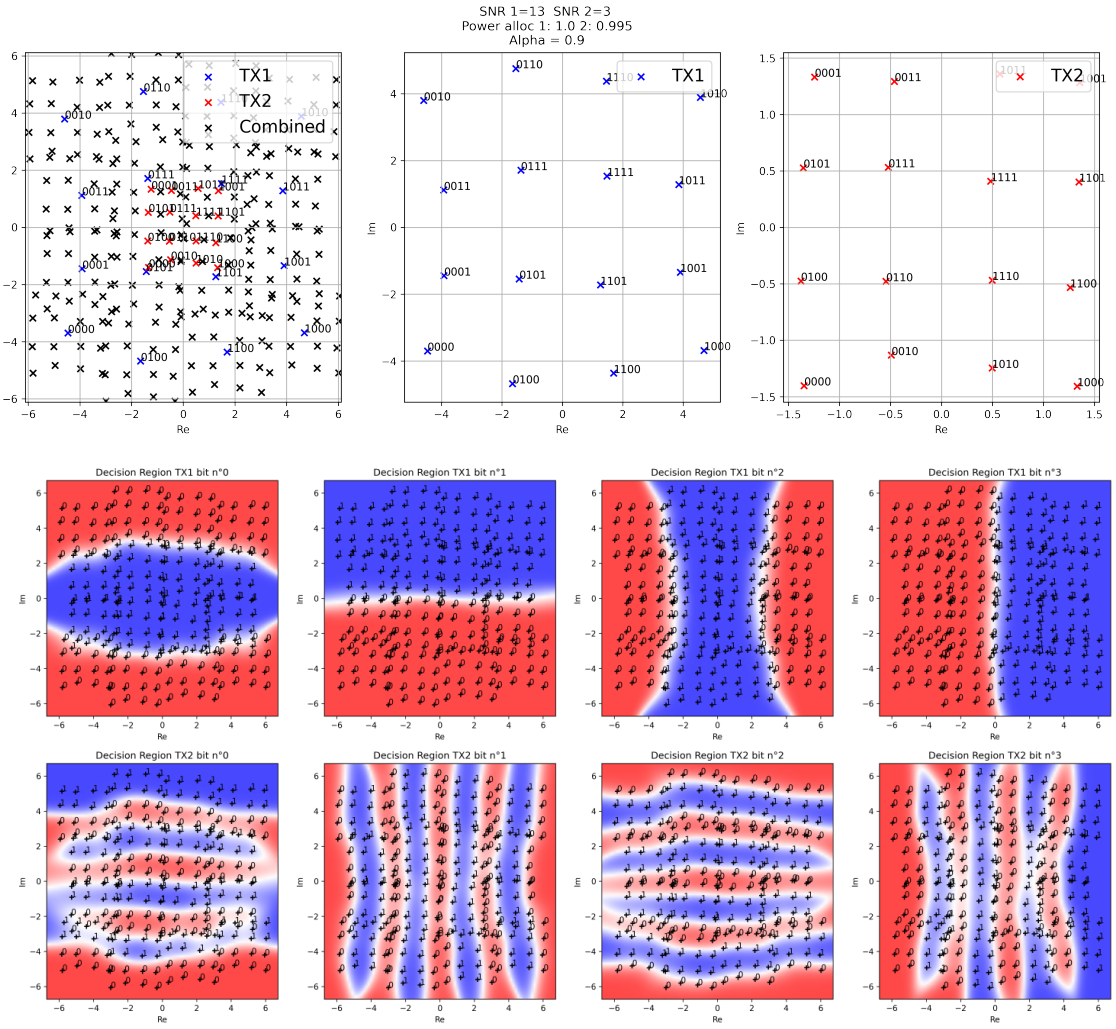

Figure 5.23.: Constellations obtained for $P_1 = P_2 + 10\,\text{dB}$, $P_2 = 3\,\text{dB}$ with constellation plot and corresponding decision regions. Gaussian Scenario, NN transmitter, $N_{bpm} = 4$, $N_{cu} = 1$. Corresponding rate: $R_1 = 3.73$, $R_2 = 0.00156$

5.6. Correlated OFDM results

Table 5.5.: Common OFDM rayleigh channel parameters

Channel parameter	Value
N_S	72
Δf	15 kHz
N_T	14
f	2.6 GHz
$\Delta\tau$	100 ns
v	1.4 m/s
N_P	18
N_{bpm}	4
N_{cu}	1

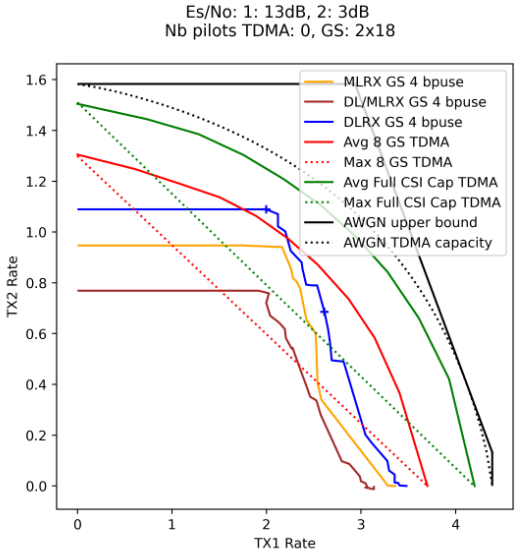

Figure 5.24.: Rate regions with $P_1 = P_2 + 10$ dB. Correlated OFDM scenario, NN transmitter with DLRX and MLRX. Compared with 8 GS and $N_P = 0$

To evaluate performance on the correlated OFDM scenario, the same 10 dB power difference and NN transmitter setup are selected, building on the results in the Gaussian scenario. Fig. 5.24 shows the rate regions obtained with this. A small number of pilots is used here (18 per user) to stress the channel estimation capabilities of the two receiver types, while the GS TDMA is brought to its full potential by using a DLRX able to operate without pilots. Table 5.5 summarises the parameters used in this section. The rates account for the loss in available data resource elements. The AWGN capacity regions are provided for context and upper bounds, but they are not achievable in this Rayleigh fading channel.

The DLRX and MLRX rate regions are obtained as described previously, by training a pool of transmitters, and keeping the best performing ones. The DL/MLRX region is obtained by

testing only the constellations from the DLRX curve, with the MLRX receiver.

The first observation that can be made from this figure is that the DLRX is the best performing one, with a large zone outside the average power TDMA GS region. But it does not show as much gains as it did in the previous Gaussian scenario.

The second observation comes when comparing the MLRX and DL/MLRX curves. The constellations leading to the best performance in DLRX are not the ones leading to the best performance in MLRX, far from it. This hints to the fact that the best constellations in DLRX contain features that help the receiver estimate the channel, at the cost of some reduction in symbol decoding capabilities. The channel estimation improvement would outweigh the decoding performance reduction. But the MLRX is hardwired to only use the pilot symbols for channel estimation, so it only sees the reduced symbol decoding performance.

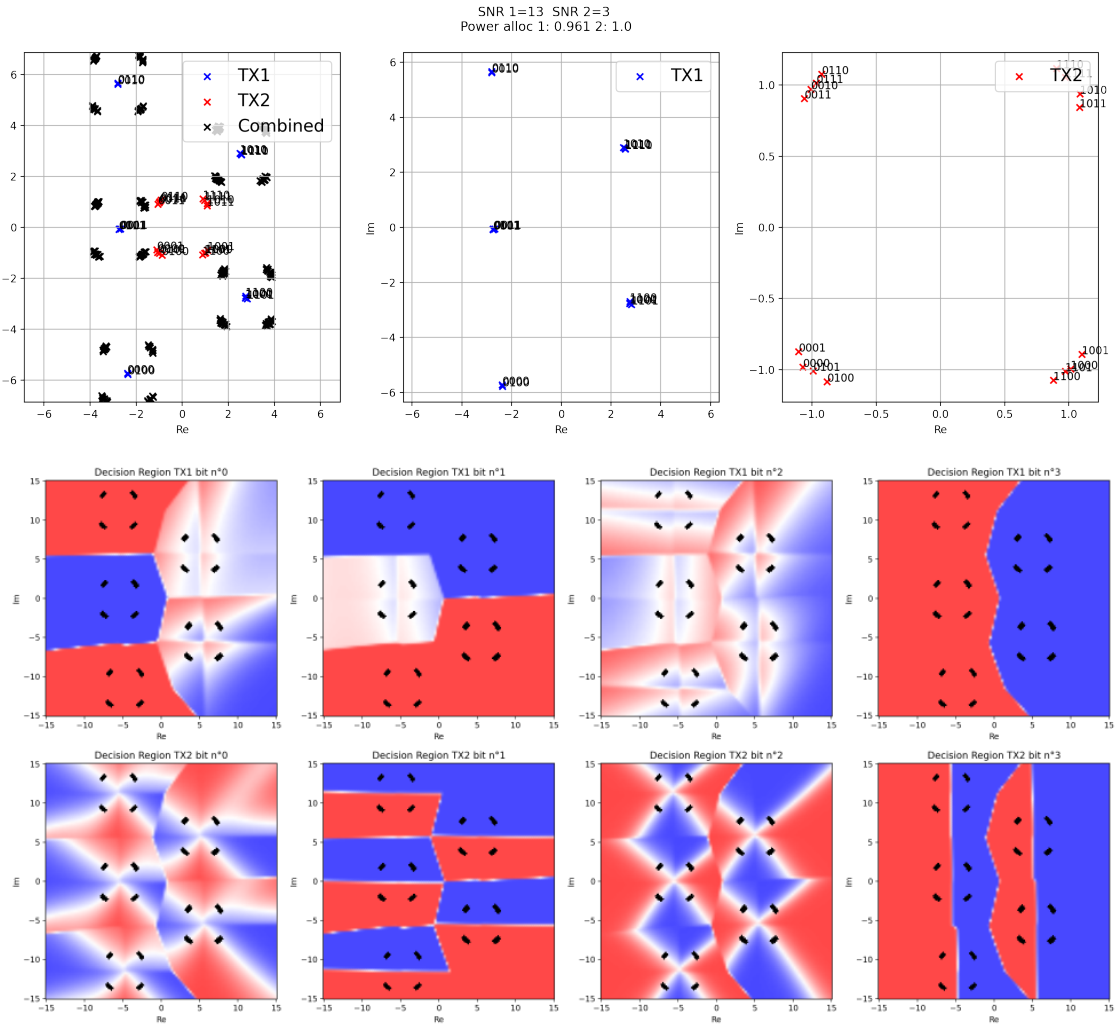

Figure 5.25.: Constellations obtained for $P_1 = P_2 + 10\,\text{dB}$, $P_2 = 3\,\text{dB}$ with constellation plot and corresponding decision regions. correlated OFDM scenario, NN transmitter with DLRX. Corresponding rate: $R_1 = 2.07$, $R_2 = 1.13$, top marker in Fig. 5.24.

Figures 5.25 and 5.26 are two examples of constellations for DLRX (marked in the rate

region plot). They show similar characteristics as the ones in the Gaussian scenario, but with a less regular spreading of symbols in the available complex space, and more distinct clusters, making it easier to see which fractions of bits are well defined and which are not. Here, the power allocation is also more heavily used to separate the two constellations.

In this case again, only part of one bit can be sacrificed, a good example being bit 1 of TX1 in Fig. 5.25 where eight symbols are in the white zone while four are in each of the red and blue zone, meaning that, for this bit is well defined half of the time. For this transmitter, bit number 3 is fully defined, bit number 2 is completely ambiguous. The first bit is more complex to analyse, with four symbols (bit to 0) in the red zones, two pairs of 0 and 1 bits in the white zone, and six symbols with the bit to 1 and two with the bit to 0 in the blue zone, leading to about half the symbols to be well defined. The combination of these partial bit definitions can help understand the overall rate of 2.07. The same analysis can be done for the other constellation. TX2 has symbols that are closer to each other due to its inferior power, leading it to be more easily affected to noise, even when the decision regions are well defined, reducing the achievable rate.

5.7. Conclusion

We have shown that a dense neural network can learn constellations tailored to the two users uplink NOMA scenario with important gains seen against the classical TDMA approaches and previous literature results. This is the case even with the more optimistic and less realistic average power constrained TDMA, and even in Rayleigh fading channel conditions. We have thus shown that discrete constellations can achieve better rates in NOMA than in OMA, something that was only theorised before. The study of this multiple access system leads to important trade-off considerations that cannot be captured with only looking at aggregate metrics such as the sum-rate.

However, training in the more realistic Rayleigh fading conditions leads to difficult convergence, but gains could be made in this area by improving the training process, with a possibility of learning channel estimation without dedicated resource elements for explicit piloting.

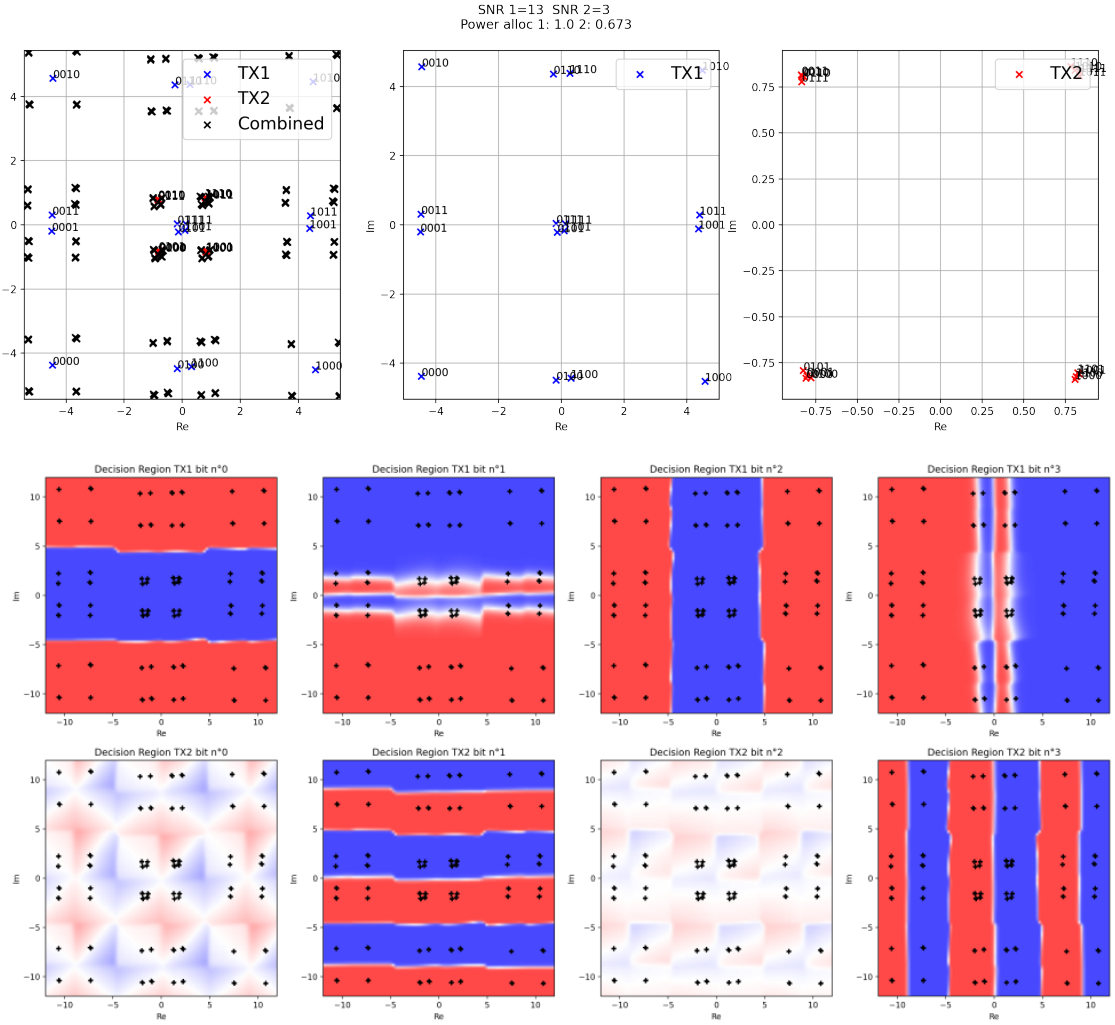

Figure 5.26.: Constellations obtained for $P_1 = P_2 + 10\,\text{dB}$, $P_2 = 3\,\text{dB}$ with constellation plot and corresponding decision regions. correlated OFDM scenario, NN transmitter with DLRX. Corresponding rate: $R_1 = 2.71, R_2 = 0.711$, bottom marker in Fig. 5.24.

— 6 —

Conclusion

THIS thesis has explored and shown the effectiveness of the use of the deep learning tool in detection tasks for wireless physical layer with multiple users, by either working in poorly modelled situations (Chapter 3, Chapter 5), or by improving performance and processing speed compared with more traditional algorithms (Chapter 4). DL is a tool well suited for this situation, especially since these detection tasks are executed frequently, for each received packet, or even each received symbol. This means that large quantities of data can be generated and are relatively simple to label to feed supervised machine learning algorithms. DL is a powerful tool that can avoid the need for construction of expert models, but can be complex and time consuming to implement due to the difficulty in finding performant neural network architectures and suitable hyperparameter tunings. This is compounded by the fact that the machine learning approach best shows its potential when working with real world data points. The design and implementation of experiments with test-beds and real hardware is thus paramount to the validation of DL systems but is a challenging task in itself and can lead to unexpected blockages, as seen in Chapter 3.

The works presented in this thesis, as well as most of the ones present in the literature of machine learning for communication are simulations or test-bench implementations. To envision putting them in production, deploying them efficiently and robustly on consumer devices, three main challenges need to be overcome by the community: efficient neural implantation, online training protocols, and reliability:

- The neural networks presented here have been implemented and trained on a server with CPU and GPUs available. These processing units are powerful, able to handle massive networks with important memory bandwidth, floating point operations, and large memory storage space. But consumer, and even more so IoT, devices are space, energy, and cost constrained. Dedicated GPU units are generally not available on them, as well as powerful CPUs. For this reason, dedicated tensor processing units are being developed to allow for more efficient inference. But, to fit in these, special attention has to be given to proposed DL schemes to reduce network size and simplify computations, especially using quantisation to reduce the size of numbers storing the parameters, from floating points numbers of 32 or even 64 bits, to integers of a few bits. This quantisation step can impact the resulting network performance, but, as mentioned in Chapter 2,

measures can be taken to alleviate this issue [53].

- By their nature, deep learning systems require large amounts of data to train. Even though data can be generated at high speed when using transmitted symbols or packets from high throughput protocols, data gathering and training can last several minutes, or even hours. And at the physical layer, operating conditions can change quickly, especially on mobile devices. Thus frequent re-training or fine-tuning can be expected in production systems. So protocols have to be devised to handle feedback links, transmission of training data, gradients, and/or network parameters for online update, all while not disturbing the standard communication stream. Techniques to reduce of the amount of new examples needed for update, such as one-shot learning or transfer learning, is also an important enabling factor to allow dynamic and adaptable systems.

- A reliability issue of deep learning approaches come from the lack of explainability of the networks. The high dimensionality of the network parameters means that the learned function is highly complex and challenging to understand and so is the impact and importance of the different input features. So, whereas edge conditions can occur with traditional algorithms that can be relatively simple to characterise, the multidimensional "edges" of a neural network can lead to a large amount of dispersed, performance holes in the input space. This reliability issue interacts with the security aspect when considering the threat of adversarial attacks [105], [106]. There, learning systems are trained to find and exploit the numerous vulnerabilities of a legitimate neural network. The very nature of physical layer wireless processing, listening to any wireless transmission, means that the systems at this level are directly exposed to malicious agents and thus can be vulnerable. On the other hand, attacks on many of these system would "only" lead to performance reductions in their functions, with an effect similar to already existing jamming attacks (with the exception of user authentication). The other protection comes from the very adversary of wireless communications, noise and channel effects: many adversarial techniques rely on precisely crafted malicious signals to be effective, and the unpredictable distortion caused by the channel between attacker and target can render them useless. But some attacks have been shown to work, even in the presence of noisy, real world signals [151].

Part I.

APPENDICES

— A —

P2P autoencoder GNU Radio implementation

A.1. System description

The objective of this implementation work is to integrate an autoencoder able to learn constellation shaping in the encoder, and decode it at the receiver, inside of a GNU Radio OFDM transmission, replacing the standard constellation encoder and decoder by learnable versions. As shown in Fig. A.1, the generation of bits to transmit, creation of packets, and allocation of encoded symbols in OFDM frames, modulation,... at the transmitter, and signal detection, synchronisation, header decoding, OFDM demodulation and equalisation is handled by common GNU Radio blocks. This approach allows the reuse of the created learning blocks with different modulations (OFDM or baseband), hardware, or even MAC layer protocols. Because there are GNU Radio blocks and hardware between the encoder and decoder, a gradient of the loss function computed at the receiver cannot be propagated back to the transmitter to train it. The approach proposed in [124] inspired by reinforcement learning is used to estimate the gradient at the transmitter from the loss.

In this framework, transmission occurs in packets of data contained inside ODFM frames of N_T ofdm symbols, including two synchronisation symbols and one header symbol. Symbols comprise of $N_S = 64$ subcarriers, 44 of them being used for the payload. Four pilots are inserted at regular intervals throughout each symbol, and are used by the GNU Radio equalisation blocks and discarded before the decoding block. The value of N_S is fixed while N_T depends on the quantity of bits to transmit. Each frame contains N_{mf} messages of N_{bpm}. These are encoded into the same amount of complex symbols.

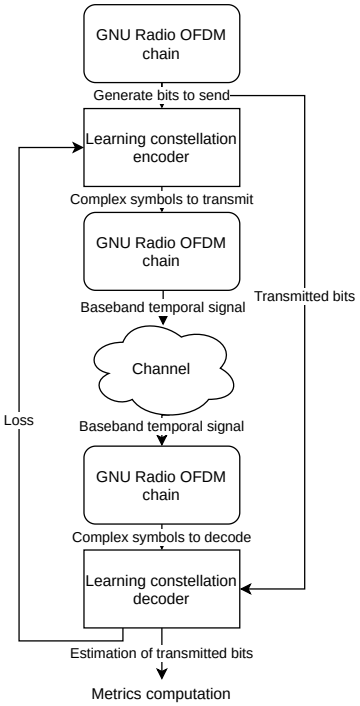

Figure A.1.: Flowchart of the autoencoder implementation

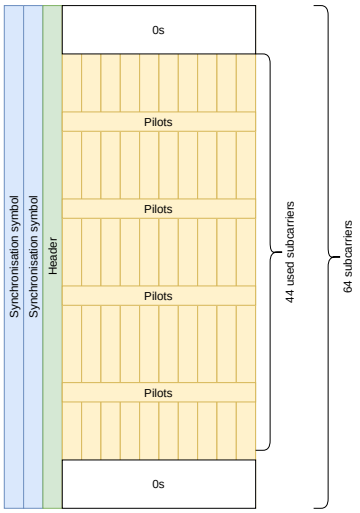

Figure A.2.: Diagram of the used OFDM frame

A.2. Labelling data

The proposed approach here implements supervised learning. Therefore, during the learning phase, labels have to be sent to the decoder. In GNU Radio, if everything is done on one computer, and on the same flowgraph, it can be done by connecting the decoder block to the same block that feeds bits to the encoder to get a perfect feedforward link. For a more realistic implementation with multiple computers, the labels need to have a dedicated and noiseless channel. A ZeroMQ broker is used to send the transmitted bits to the receivers over Ethernet.

An issue comes from the fact that the transmission link, can have frame losses, if the synchronisation block doesn't detect a frame preamble. To handle this situation, a packet number is added as a tag at the beginning of each block of bits that will constitute a frame's payload. It is coded over twelve bits, so it loops back to zero after 4096 transmitted packets. That number is then carried over the air in the frame header and read and put back in tag form at the receiver.

Two streams of tagged data thus arrive at the decoder, but with potentially missing packets in one or the other. A custom block is necessary to synchronise the two streams by dropping a packet from a steam that doesn't have a packet number match on the other stream. It does this by looking at all the packets in its first input buffer in chronological order, and for each of them, check if there is a packet with the same number in the other buffer. Stopping when the first match is found.

This works well but can cause bottleneck issues if packet loss and/or transmission delay is high: the packets can stack up in the label stream, and the block waits to see one packet number match between its two inputs to be able to drop the packets that arrived before that pair. And, at some point, the label buffer can get full, preventing the bit generation at the beginning of the chain from creating new data, so, no new data is sent and received, and the whole chain stops. This bottleneck is only present when both the transmission and reception occur on the same computer.

A.3. Decoder

The decoder is implemented as a small dense network, with two modes of operation: symbol- and bit-wise and trained as a classical supervised training task. It operates separately on each received complex symbol y, so a frame is treated as a batch of examples to the network. In symbol-wise operation, the output is a vector of all $2^{N_{bpm}}$ possible messages with the label $\mathbf{t}$ a one-hot vector. A softmax activation function is applied to the raw output of the network, leading to the estimated probability vector $\hat{\mathbf{t}}$ and the loss $\mathcal{L}$ is the categorical cross-entropy.

$$\mathcal{L} = -\sum_{n=1}^{2^{N_{bpm}}} \mathbf{t}_n \cdot \log\left(\hat{\mathbf{t}}_n\right) \tag{A.1}$$

In bitwise operation, labels are the sequence of bits $\mathbf{b}$ encoding the messages, and the network outputs a vector of probability over these bits $\hat{\mathbf{b}}$ after a sigmoid activation. The loss is a binary cross-entropy.

$$\mathcal{L} = -\sum_{n=1}^{N_{bpm}} \mathbf{b}_n \cdot \log\left(\hat{\mathbf{b}}_n\right) + (1 - \mathbf{b}_n) \cdot \log\left(1 - \hat{\mathbf{b}}_n\right) \tag{A.2}$$

A.4. Encoder

The encoder is implemented as an embedding whose output is normalised by power. This means that the parameters to train θ_T do not comprise a neural network, but a matrix with an entry for each $2^{N_{bpm}}$ possible message m. Its training is based on [124] and operates as follows. When training is enabled, an exploration noise $w \sim \mathcal{N}_{\mathbb{C}}(0, \sigma_\pi)$ is added to the normalised output $f^T_{\theta_T}(m)$ leading to a noisy output $x_p(m) = \sqrt{1-\sigma_\pi^2} f^T_{\theta_T}(m) + w$ to conserve the transmission power normalisation. It allows to estimate the gradient of the loss computed at the decoder with regards to the parameters of the transmitter. When operating on a batch of K transmitted messages (usually $K = N_{mf}$)

$$\nabla_{\theta_T} \hat{\mathcal{L}} = \frac{2}{K} \sum_{k \in K} \nabla_{\theta_T} f^T_{\theta_T}(m_k) \frac{w}{\sigma_\pi^2} \mathcal{L}_k \tag{A.3}$$

For this, the decoder sends a message after each received packet containing the individual losses for each message of this packet. The encoder computes and stores $\frac{w}{\sigma_\pi^2}$ and when the loss corresponding to a packet is received by message, the supervised loss $\mathcal{L}_k$ is multiplied by the stored one, and the recording of computation is used to compute the gradient of that loss and update the embedding.

A.5. GUI and interactions

Some parameters of the simulation and learning blocks are exposed for modification at runtime with a graphical user interface (GUI):

- Simulation
  - Tx and RX USRP gains
  - Added noise level
  - Reset BER computation
- Encoder
  - Start/stop training
  - Training alternance speed
  - Reset network
  - Save and load network
  - Learning rate
  - Exploration noise level
- Decoder
  - Start/stop training
  - Training alternance speed
  - Reset network
  - Save and load network

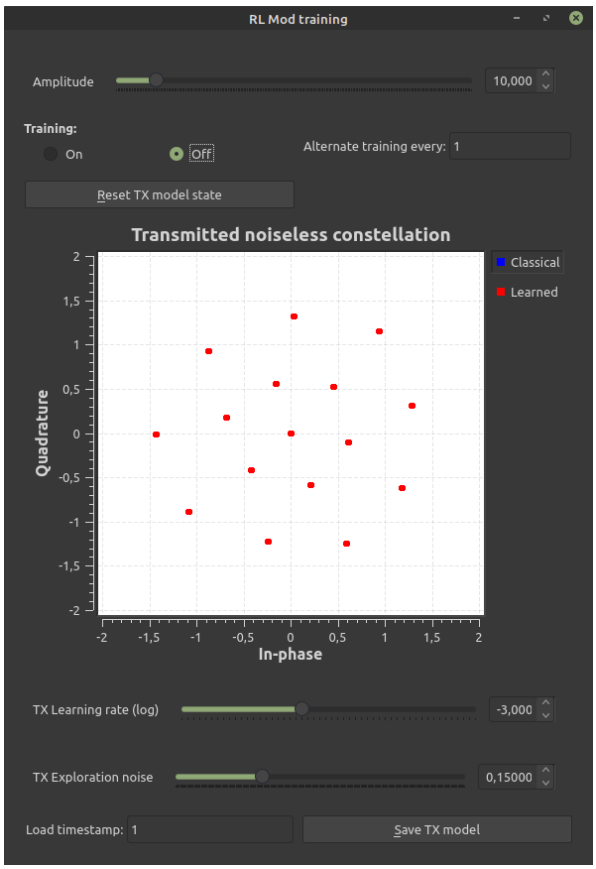

Figure A.3.: GUI element for the transmitter. Symbol-wise optimisation on an AWGN channel at 10dB SNR

– Learning rate

This as also been adapted to the use of the gr-bokehgui display framework [152] to allow monitoring and interactions inside of the FIT/CorteXlab testbed. Fig.A.3 and A.4 present the resulting GUI windows with examples of constellations obtained after convergence using symbol-wise and bit-wise loss functions. It can be noted that convergence with the symbol-wise optimisation is faster and more robust than the other. This is due to the fact that it does not require to learn an efficient bit labelling, so the task is simpler.

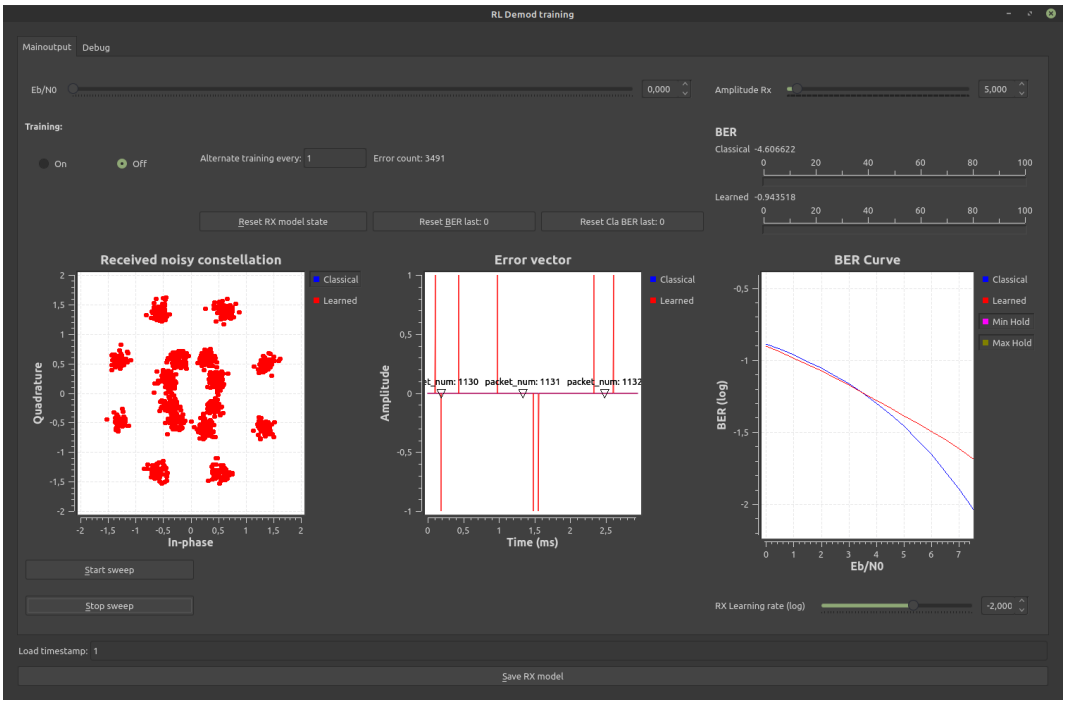

Figure A.4.: GUI element for the receiver. Bit-wise optimisation on an AWGN channel at 4dB SNR

— A —

Bibliography

[1] A. Massouri, L. Cardoso, B. Guillon, F. Hutu, G. Villemaud, T. Risset, and J.-M. Gorce, "Cortexlab: an open FPGA-based facility for testing SDR & cognitive radio networks in a reproducible environment", in *Computer Communications Workshops (INFOCOM WKSHPS), 2014 IEEE Conference on*, IEEE, 2014, pp. 103–104. DOI: 10.1109/INFCOMW.2014.6849176.

[2] D. Evans. (2011). The internet of things: how the next evolution of the internet is changing everything, [Online]. Available: https://www.cisco.com/c/dam/en_us/about/ac79/docs/innov/IoT_IBSG_0411FINAL.pdf.

[3] J. Iwata. (2012). Making markets: smarter planet, [Online]. Available: https://web.archive.org/web/20171122045938/https://www.ibm.com/investor/events/investor0512/presentation/05_Smarter_Planet.pdf.

[4] M. Ibnkahla, "Applications of neural networks to digital communications – a survey", *Signal Processing*, vol. 80, no. 7, pp. 1185–1215, 2000, ISSN: 0165-1684. DOI: https://doi.org/10.1016/S0165-1684(00)00030-X. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S016516840000030X.

[5] T. J. O'Shea, J. Corgan, and T. C. Clancy, "Convolutional radio modulation recognition networks", in *International conference on engineering applications of neural networks*, Springer, 2016, pp. 213–226.

[6] E. Nachmani, Y. Be'ery, and D. Burshtein, "Learning to decode linear codes using deep learning", in *2016 54th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, IEEE, 2016, pp. 341–346.

[7] T. J. O'Shea, K. Karra, and T. C. Clancy, "Learning to communicate: channel auto-encoders, domain specific regularizers, and attention", in *2016 IEEE International Symposium on Signal Processing and Information Technology (ISSPIT)*, IEEE, 2016, pp. 223–228.

[8] T. O'Shea and J. Hoydis, "An introduction to deep learning for the physical layer", *IEEE Transactions on Cognitive Communications and Networking*, vol. 3, no. 4, pp. 563–575, 2017. DOI: 10.1109/TCCN.2017.2758370.

[9] J. Jagannath, N. Polosky, A. Jagannath, F. Restuccia, and T. Melodia, "Machine learning for wireless communications in the internet of things: a comprehensive survey", *Ad Hoc Networks*, vol. 93, p. 101 913, 2019.

[10] C. Morin, L. Cardoso, J. Hoydis, and J.-M. Gorce, "Channel resilience inducing dataset construction for physical layer device fingerprinting", *IEEE Transactions on Cognitive Communications and Networking*, 2021.

[11] C. Morin, L. Cardoso, J. Hoydis, J.-M. Gorce, and T. Vial, "Transmitter Classification With Supervised Deep Learning", in *Crowncom 2019 - 14th EAI International conference on Cognitive Radio Oriented Wireless Networks*, Poznan, Poland, Jun. 2019, pp. 1–14.

[12] C. Morin, D. Duchemin, J.-M. S. Gorce, C. Goursaud, and L. Sampaio Cardoso, "Active user blind detection through deep learning", in *Crowncom 2020 - 15th EAI International Conference on Cognitive Radio Oriented Wireless Networks*, Rome, Italy, Nov. 2020, pp. 1–14.

[13] C. Morin, L. Cardoso, J. Hoydis, and J.-M. Gorce, "Neural Constellation shaping for the two-user uplink NOMA gaussian channel", in *IEEE Global Communications Conference*, Madrid, Spain, Dec. 2021.

[14] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, `http://www.deeplearningbook.org`.

[15] T. M. Mitchell *et al.*, "Machine learning", 1997.

[16] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition", *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[17] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: a survey", *ACM computing surveys (CSUR)*, vol. 41, no. 3, pp. 1–58, 2009.

[18] ——, "Anomaly detection for discrete sequences: a survey", *IEEE transactions on knowledge and data engineering*, vol. 24, no. 5, pp. 823–839, 2010.

[19] Z. Pan, W. Yu, X. Yi, A. Khan, F. Yuan, and Y. Zheng, "Recent progress on generative adversarial networks (gans): a survey", *IEEE Access*, vol. 7, pp. 36 322–36 333, 2019.

[20] P. Cortez and A. d. J. R. Morais, "A data mining approach to predict forest fires using meteorological data", 2007.

[21] M. Fernández-Delgado, M. S. Sirsat, E. Cernadas, S. Alawadi, S. Barro, and M. Febrero-Bande, "An extensive experimental survey of regression methods", *Neural Networks*, vol. 111, pp. 11–34, 2019.

[22] Y. Wu, M. Schuster, Z. Chen, Q. V. Le, M. Norouzi, W. Macherey, M. Krikun, Y. Cao, Q. Gao, K. Macherey, *et al.*, "Google's neural machine translation system: bridging the gap between human and machine translation", *arXiv preprint arXiv:1609.08144*, 2016.

[23] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[24] H. Steinhaus, "Sur la division des corps matériels en parties", *Bull. Acad. Polon. Sci*, vol. 1, no. 804, p. 801, 1956.

[25] H. Hotelling, "Analysis of a complex of statistical variables into principal components.", *Journal of educational psychology*, vol. 24, no. 6, p. 417, 1933.

[26] B. E. Boser, I. M. Guyon, and V. N. Vapnik, "A training algorithm for optimal margin classifiers", in *Proceedings of the fifth annual workshop on Computational learning theory*, 1992, pp. 144–152.

[27] A. Kratsios, "The universal approximation property", *Annals of Mathematics and Artificial Intelligence*, pp. 1–35, 2021.

[28] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks", in *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, JMLR Workshop and Conference Proceedings, 2011, pp. 315–323.

[29] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines", in *Icml*, 2010.

[30] A. L. Maas, A. Y. Hannun, and A. Y. Ng, "Rectifier nonlinearities improve neural network acoustic models", in *Proc. icml*, Citeseer, vol. 30, 2013, p. 3.

[31] D.-A. Clevert, T. Unterthiner, and S. Hochreiter, "Fast and accurate deep network learning by exponential linear units (elus)", *arXiv preprint arXiv:1511.07289*, 2015.

[32] F. Yu and V. Koltun, "Multi-scale context aggregation by dilated convolutions", *arXiv preprint arXiv:1511.07122*, 2015.

[33] S. Hochreiter and J. Schmidhuber, "Long short-term memory", *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[34] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder-decoder for statistical machine translation", *arXiv preprint arXiv:1406.1078*, 2014.

[35] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks", *Advances in neural information processing systems*, vol. 25, pp. 1097–1105, 2012.

[36] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition", in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[37] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, *et al.*, "Tensorflow: a system for large-scale machine learning", in *12th {USENIX} symposium on operating systems design and implementation ({OSDI} 16)*, 2016, pp. 265–283.

[38] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: an imperative style, high-performance deep learning library", in *Advances in Neural Information Processing Systems 32*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds., Curran Associates, Inc., 2019, pp. 8024–8035. [Online]. Available: `http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf`.

[39] N. Qian, "On the momentum term in gradient descent learning algorithms", *Neural networks*, vol. 12, no. 1, pp. 145–151, 1999.

[40] D. P. Kingma and J. Ba, "Adam: a method for stochastic optimization", 2015.

[41] A. Paullada, I. D. Raji, E. M. Bender, E. Denton, and A. Hanna, "Data and its (dis)contents: a survey of dataset development and use in machine learning research", *arXiv:2012.05345 [cs]*, Dec. 9, 2020. arXiv: 2012.05345. [Online]. Available: http://arxiv.org/abs/2012.05345 (visited on 02/23/2021).

[42] C. Shorten and T. M. Khoshgoftaar, "A survey on image data augmentation for deep learning", *Journal of Big Data*, vol. 6, no. 1, pp. 1–48, 2019.

[43] L. Von Ahn, M. Blum, N. J. Hopper, and J. Langford, "Captcha: using hard ai problems for security", in *International conference on the theory and applications of cryptographic techniques*, Springer, 2003, pp. 294–311.

[44] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: a large-scale hierarchical image database", in *2009 IEEE conference on computer vision and pattern recognition*, Ieee, 2009, pp. 248–255.

[45] K. Yang, K. Qinami, L. Fei-Fei, J. Deng, and O. Russakovsky, "Towards fairer datasets: filtering and balancing the distribution of the people subtree in the ImageNet hierarchy", *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency*, pp. 547–558, Jan. 27, 2020. DOI: 10.1145/3351095.3375709. arXiv: 1912.07726. [Online]. Available: http://arxiv.org/abs/1912.07726 (visited on 02/23/2021).

[46] T. J. O'Shea and N. West, "Radio machine learning dataset generation with GNU radio", *Proceedings of the GNU Radio Conference*, vol. 1, no. 1, Sep. 6, 2016, Number: 1. [Online]. Available: https://pubs.gnuradio.org/index.php/grcon/article/view/11 (visited on 02/23/2021).

[47] J. Dastin, "Amazon scraps secret AI recruiting tool that showed bias against women", *Reuters*, Oct. 10, 2018. [Online]. Available: https://www.reuters.com/article/us-amazon-com-jobs-automation-insight-idUSKCN1MK08G (visited on 02/23/2021).

[48] V. U. Prabhu and A. Birhane, "Large image datasets: a pyrrhic win for computer vision?", *arXiv:2006.16923 [cs, stat]*, Jul. 23, 2020. arXiv: 2006.16923. [Online]. Available: http://arxiv.org/abs/2006.16923 (visited on 03/02/2021).

[49] A. Pinkus, "Approximation theory of the MLP model in neural networks", *Acta Numerica*, vol. 8, pp. 143–195, Jan. 1999. DOI: 10.1017/S0962492900002919.

[50] T. J. O'Shea, L. Pemula, D. Batra, and T. C. Clancy, "Radio transformer networks: attention models for learning to synchronize in wireless systems", in *2016 50th Asilomar Conference on Signals, Systems and Computers*, IEEE, 2016, pp. 662–666.

[51] J. S. Bridle, "Probabilistic interpretation of feedforward classification network outputs, with relationships to statistical pattern recognition", in *Neurocomputing*, Springer, 1990, pp. 227–236.

[52] K. Hirose, K. Ando, K. Ueyoshi, M. Ikebe, T. Asai, M. Motomura, and S. Takamaeda-Yamazaki, "Quantization error-based regularization in neural networks", in *International Conference on Innovative Techniques and Applications of Artificial Intelligence*, Springer, 2017, pp. 137–142.

[53] M. A. Carreira-Perpinán and Y. Idelbayev, "Model compression as constrained optimization, with application to neural nets. part ii: quantization", *arXiv preprint arXiv:1707.04319*, 2017.

[54] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting", *Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[55] T. Elsken, J. H. Metzen, and F. Hutter, "Neural architecture search: a survey", *Journal of Machine Learning Research*, vol. 20, no. 55, pp. 1–21, 2019. [Online]. Available: `http://jmlr.org/papers/v20/18-598.html`.

[56] A. Heuillet, F. Couthouis, and N. Diaz-Rodriguez, "Explainability in deep reinforcement learning", *Knowledge-Based Systems*, vol. 214, p. 106 685, 2021.

[57] N. Burkart and M. F. Huber, "A survey on the explainability of supervised machine learning", *Journal of Artificial Intelligence Research*, vol. 70, pp. 245–317, 2021.

[58] L. Sampaio Cardoso, O. Oubejja, G. Villemaud, T. Risset, and J. M. Gorce, "Reliable and Reproducible Radio Experiments in FIT/CorteXlab SDR testbed: Initial Findings", in *Crowncom*, Lisbon, Portugal, Sep. 2017. [Online]. Available: `https://hal.inria.fr/hal-01598491`.

[59] D. Rupprecht, K. Jansen, and C. Pöpper, "Putting LTE security functions to the test: a framework to evaluate implementation correctness", in *10th USENIX Workshop on Offensive Technologies (WOOT 16)*, Austin, TX: USENIX Association, 2016.

[60] F. Fund. (). Run a man-in-the-middle attack on a wifi hotspot, [Online]. Available: `https://witestlab.poly.edu/blog/conduct-a-simple-man-in-the-middle-attack-on-a-wifi-hotspot/`.

[61] M. Mitev, A. Chorti, M. Reed, and L. Musavian, "Authenticated secret key generation in delay-constrained wireless systems", *EURASIP Journal on Wireless Communications and Networking*, vol. 2020, no. 1, pp. 1–29, 2020.

[62] K. Ellis and N. Serinken, "Characteristics of radio transmitter fingerprints", *Radio Science*, vol. 36, no. 4, pp. 585–597, 2001.

[63] X. Wang, P. Hao, and L. Hanzo, "Physical-layer authentication for wireless security enhancement: Current challenges and future developments", *IEEE Communications Magazine*, vol. 54, no. 6, pp. 152–158, Jun. 2016. DOI: `10.1109/MCOM.2016.7498103`. [Online]. Available: `http://ieeexplore.ieee.org/document/7498103/`.

[64] L. Xiao, L. J. Greenstein, N. B. Mandayam, and W. Trappe, "Using the physical layer for wireless authentication in time-variant channels", *IEEE Transactions on Wireless Communications*, vol. 7, no. 7, pp. 2571–2579, 2008.

[65] L. Xiao, L. Greenstein, N. Mandayam, and W. Trappe, "Fingerprints in the ether: using the physical layer for wireless authentication", in *2007 IEEE International Conference on Communications*, IEEE, 2007, pp. 4646–4651.

[66] Y. Chen, H. Wen, J. Wu, H. Song, A. Xu, Y. Jiang, T. Zhang, and Z. Wang, "Clustering based physical-layer authentication in edge computing systems with asymmetric resources", *Sensors*, vol. 19, no. 8, p. 1926, Jan. 2019. DOI: `10.3390/s19081926`. [Online]. Available: `https://www.mdpi.com/1424-8220/19/8/1926` (visited on 02/05/2020).

[67] L. Senigagliesi, M. Baldi, and E. Gambi, "Statistical and machine learning-based decision techniques for physical layer authentication", pp. 1–6, 2019.

[68] A. Weinand, M. Karrenbauer, R. Sattiraju, and H. Schotten, "Application of machine learning for channel based message authentication in mission critical machine type communication", in *European Wireless 2017; 23th European Wireless Conference*, VDE, 2017, pp. 1–5.

[69] L. J. Wong, W. C. Headley, and A. J. Michaels, "Emitter identification using cnn iq imbalance estimators", *arXiv preprint arXiv:1808.02369*, 2018.

[70] G. Li, J. Yu, Y. Xing, and A. Hu, "Location-invariant physical layer identification approach for WiFi devices", *IEEE Access*, vol. 7, pp. 106 974–106 986, 2019, ISSN: 2169-3536. DOI: 10.1109/ACCESS.2019.2933242.

[71] O. Ureten and N. Serinken, "Wireless security through RF fingerprinting", *Canadian Journal of Electrical and Computer Engineering*, vol. 32, no. 1, pp. 27–33, 2007, ISSN: 0840-8688. DOI: 10.1109/CJECE.2007.364330.

[72] X. Wang, J. Duan, C. Wang, G. Cui, and W. Wang, "A radio frequency fingerprinting identification method based on energy entropy and color moments of the bispectrum", in *2017 9th International Conference on Advanced Infocomm Technology (ICAIT)*, ISSN: null, Nov. 2017, pp. 150–154. DOI: 10.1109/ICAIT.2017.8388905.

[73] N. Hu and Y.-D. Yao, "Identification of legacy radios in a cognitive radio network using a radio frequency fingerprinting based method", in *2012 IEEE International Conference on Communications (ICC)*, ISSN: 1550-3607, Jun. 2012, pp. 1597–1602. DOI: 10.1109/ICC.2012.6364436.

[74] S. S. Hanna and D. Cabric, "Deep learning based transmitter identification using power amplifier nonlinearity", in *2019 International Conference on Computing, Networking and Communications (ICNC)*, IEEE, 2019, pp. 674–680.

[75] Y. Pan, S. Yang, H. Peng, T. Li, and W. Wang, "Specific emitter identification based on deep residual networks", *IEEE Access*, vol. 7, pp. 54 425–54 434, 2019, ISSN: 2169-3536. DOI: 10.1109/ACCESS.2019.2913759.

[76] B. Chatterjee, D. Das, and S. Sen, "Rf-puf: iot security enhancement through authentication of wireless nodes using in-situ machine learning", in *2018 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, IEEE, 2018, pp. 205–208.

[77] A. Aghnaiya, A. M. Ali, and A. Kara, "Variational mode decomposition-based radio frequency fingerprinting of bluetooth devices", *IEEE Access*, vol. 7, pp. 144 054–144 058, 2019, ISSN: 2169-3536. DOI: 10.1109/ACCESS.2019.2945121.

[78] A. Candore, O. Kocabas, and F. Koushanfar, "Robust stable radiometric fingerprinting for wireless devices", in *2009 IEEE International Workshop on Hardware-Oriented Security and Trust*, ISSN: null, Jul. 2009, pp. 43–49. DOI: 10.1109/HST.2009.5224969.

[79] S. Chen, F. Xie, Y. Chen, H. Song, and H. Wen, "Identification of wireless transceiver devices using radio frequency (RF) fingerprinting based on STFT analysis to enhance authentication security", in *2017 IEEE 5th International Symposium on Electromagnetic Compatibility (EMC-Beijing)*, ISSN: null, Oct. 2017, pp. 1–5. DOI: 10.1109/EMC-B.2017.8260381.

[80] G. Baldini, G. Steri, R. Giuliani, and C. Gentile, "Imaging time series for internet of things radio frequency fingerprinting", in *2017 International Carnahan Conference on Security Technology (ICCST)*, ISSN: 2153-0742, Oct. 2017, pp. 1–6. DOI: 10.1109/CCST.2017.8167861.

[81] G. Baldini, R. Giuliani, and F. Dimc, "Physical layer authentication of internet of things wireless devices using convolutional neural networks and recurrence plots", *Internet Technology Letters*, vol. 2, no. 2, e81, Mar. 2019, ISSN: 24761508. DOI: 10.1002/itl2.81. [Online]. Available: http://doi.wiley.com/10.1002/itl2.81 (visited on 02/04/2020).

[82] S. Riyaz, K. Sankhe, S. Ioannidis, and K. Chowdhury, "Deep learning convolutional neural networks for radio identification", *IEEE Communications Magazine*, vol. 56, no. 9, pp. 146–152, 2018.

[83] K. Youssef, L. Bouchard, K. Haigh, J. Silovsky, B. Thapa, and C. Vander Valk, "Machine learning approach to rf transmitter identification", *IEEE Journal of Radio Frequency Identification*, vol. 2, no. 4, pp. 197–205, 2018.

[84] I. O. Kennedy, P. Scanlon, F. J. Mullany, M. M. Buddhikot, K. E. Nolan, and T. W. Rondeau, "Radio transmitter fingerprinting: a steady state frequency domain approach", in *2008 IEEE 68th Vehicular Technology Conference*, ISSN: 1090-3038, Sep. 2008, pp. 1–5. DOI: 10.1109/VETECF.2008.291.

[85] H. Jafari, O. Omotere, D. Adesina, H.-H. Wu, and L. Qian, "IoT devices fingerprinting using deep learning", in *MILCOM 2018 - 2018 IEEE Military Communications Conference (MILCOM)*, ISSN: 2155-7578, Oct. 2018, pp. 1–9. DOI: 10.1109/MILCOM.2018.8599826.

[86] D. Roy, T. Mukherjee, M. Chatterjee, E. Blasch, and E. Pasiliao, "RFAL: adversarial learning for RF transmitter identification and classification", *IEEE Transactions on Cognitive Communications and Networking*, vol. 6, no. 2, pp. 783–801, 2020, ISSN: 2332-7731. DOI: 10.1109/TCCN.2019.2948919.

[87] Y. Li, X. Chen, Y. Lin, G. Srivastava, and S. Liu, "Wireless transmitter identification based on device imperfections", *IEEE Access*, vol. 8, pp. 59 305–59 314, 2020, Conference Name: IEEE Access, ISSN: 2169-3536. DOI: 10.1109/ACCESS.2020.2981428.

[88] K. Merchant, S. Revay, G. Stantchev, and B. Nousain, "Deep learning for rf device fingerprinting in cognitive communication networks", *IEEE Journal of Selected Topics in Signal Processing*, vol. 12, no. 1, pp. 160–167, 2018.

[89] J. Yu, A. Hu, F. Zhou, Y. Xing, Y. Yu, G. Li, and L. Peng, "Radio frequency fingerprint identification based on denoising autoencoders", in *2019 International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*, IEEE, 2019, pp. 1–6.

[90] J. Yu, A. Hu, G. Li, and L. Peng, "A robust RF fingerprinting approach using multi-sampling convolutional neural network", *IEEE Internet of Things Journal*, vol. 6, no. 4, pp. 6786–6799, Aug. 2019, ISSN: 2372-2541. DOI: 10.1109/JIOT.2019.2911347.

[91] L. Peng, A. Hu, J. Zhang, Y. Jiang, J. Yu, and Y. Yan, "Design of a hybrid RF fingerprint extraction and device classification scheme", *IEEE Internet of Things Journal*, vol. 6, no. 1, pp. 349–360, Feb. 2019, ISSN: 2372-2541. DOI: 10.1109/JIOT.2018.2838071.

[92] A. Al-Shawabka, F. Restuccia, S. D'Oro, T. Jian, B. C. Rendon, N. Soltani, J. Dy, S. Ioannidis, K. Chowdhury, and T. Melodia, "Exposing the fingerprint: dissecting the impact of the wireless channel on radio fingerprinting", in *IEEE INFOCOM 2020 - IEEE Conference on Computer Communications*, ISSN: 2641-9874, Jul. 2020, pp. 646–655. DOI: 10.1109/INFOCOM41043.2020.9155259.

[93] K. Sankhe, M. Belgiovine, F. Zhou, S. Riyaz, S. Ioannidis, and K. Chowdhury, "Oracle: optimized radio classification through convolutional neural networks", in *IEEE INFOCOM 2019-IEEE Conference on Computer Communications*, IEEE, 2019, pp. 370–378.

[94] K. Sankhe, M. Belgiovine, F. Zhou, L. Angioloni, F. Restuccia, S. D'Oro, T. Melodia, S. Ioannidis, and K. Chowdhury, "No radio left behind: radio fingerprinting through deep learning of physical-layer hardware impairments", *IEEE Transactions on Cognitive Communications and Networking*, vol. 6, no. 1, pp. 165–178, 2019.

[95] F. Restuccia, S. D'Oro, A. Al-Shawabka, M. Belgiovine, L. Angioloni, S. Ioannidis, K. Chowdhury, and T. Melodia, "DeepRadioID: real-time channel-resilient optimization of deep learning-based radio fingerprinting algorithms", in *Proceedings of the Twentieth ACM International Symposium on Mobile Ad Hoc Networking and Computing*, ser. Mobihoc '19, Catania, Italy: Association for Computing Machinery, Jul. 2, 2019, pp. 51–60, ISBN: 978-1-4503-6764-6. DOI: 10.1145/3323679.3326503. [Online]. Available: https://doi.org/10.1145/3323679.3326503 (visited on 02/06/2020).

[96] M. Cekic, S. Gopalakrishnan, and U. Madhow, "Robust wireless fingerprinting: generalizing across space and time", *arXiv:2002.10791 [cs, eess, stat]*, Jul. 20, 2020. arXiv: 2002.10791. [Online]. Available: http://arxiv.org/abs/2002.10791 (visited on 02/24/2021).

[97] N. Soltani, K. Sankhe, J. Dy, S. Ioannidis, and K. Chowdhury, "More is better: data augmentation for channel-resilient RF fingerprinting", *IEEE Communications Magazine*, vol. 58, no. 10, pp. 66–72, Oct. 2020, Conference Name: IEEE Communications Magazine, ISSN: 1558-1896. DOI: 10.1109/MCOM.001.2000180.

[98] A. Gritsenko, Z. Wang, T. Jian, J. Dy, K. Chowdhury, and S. Ioannidis, "Finding a 'new' needle in the haystack: unseen radio detection in large populations using deep learning", in *2019 IEEE International Symposium on Dynamic Spectrum Access Networks (DySPAN)*, 2019, pp. 1–10. DOI: 10.1109/DySPAN.2019.8935862.

[99] S. Hanna, S. Karunaratne, and D. Cabric, "Open set wireless transmitter authorization: deep learning approaches and dataset considerations", *IEEE Transactions on Cognitive Communications and Networking*, pp. 1–1, 2020, Conference Name: IEEE Transactions on Cognitive Communications and Networking, ISSN: 2332-7731. DOI: 10.1109/TCCN.2020.3043332.

[100] T. Oyedare and J.-M. J. Park, "Estimating the required training dataset size for transmitter classification using deep learning", in *2019 IEEE International Symposium on Dynamic Spectrum Access Networks (DySPAN)*, Newark, NJ, USA: IEEE, Nov. 2019, pp. 1–10, ISBN: 978-1-72812-376-9. DOI: 10.1109/DySPAN.2019.8935823. (visited on 01/31/2020).

[101] A. Mouaffo, L. Cardoso, H. Boeglen, G. Villemaud, and R. Vauzelle, "Radio link characterization of the CorteXlab testbed with a large number of software defined radio nodes", in *Antennas and Propagation (EuCAP), 2015 9th European Conference on*, Lisbon, Portugal, Apr. 2015. [Online]. Available: `https://hal.inria.fr/hal-01245107`.

[102] C. Morin. (). Datasets and data gathering code, [Online]. Available: `https://wiki.cortexlab.fr/doku.php?id=tx-id`.

[103] E. Research. (). Octoclock product overview, [Online]. Available: `https://www.ettus.com/wp-content/uploads/2019/01/Octoclock_Spec_Sheet.pdf`.

[104] S. Tomasin, A. Brighente, F. Formaggio, and G. Ruvoletto, "Physical-layer location verification by machine learning", in *Machine Learning for Future Wireless Communications*. John Wiley & Sons, Ltd, 2019, ch. 20, pp. 425–438, ISBN: 9781119562306.

[105] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks", *arXiv:1312.6199 [cs]*, Dec. 20, 2013. arXiv: 1312.6199. [Online]. Available: `http://arxiv.org/abs/1312.6199` (visited on 06/06/2019).

[106] B. Flowers, R. M. Buehrer, and W. C. Headley, "Evaluating adversarial evasion attacks in the context of wireless communications", *IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 1102–1113, 2019.

[107] N. Ye, X. Li, H. Yu, A. Wang, W. Liu, and X. Hou, "Deep learning aided grant-free NOMA toward reliable low-latency access in tactile internet of things", *IEEE Transactions on Industrial Informatics*, vol. 15, no. 5, pp. 2995–3005, May 2019.

[108] 3GPP, *Further NB-IoT enhancements (RP-171428)*, 2017.

[109] Z. Ding, X. Lei, G. K. Karagiannidis, R. Schober, J. Yuan, and V. K. Bhargava, "A survey on non-orthogonal multiple access for 5G networks: research challenges and future trends", *IEEE Journal on Selected Areas in Communications*, vol. 35, no. 10, pp. 2181–2195, 2017.

[110] E. Paolini, C. Stefanovic, G. Liva, and P. Popovski, "Coded random access: applying codes on graphs to design random access protocols", *IEEE Communications Magazine*, vol. 53, no. 6, pp. 144–150, 2015.

[111] D. Duchemin, L. Chetot, J. Gorce, and C. Goursaud, "Coded random access for massive MTC under statistical channel knowledge", in *2019 IEEE 20th International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*, 2019, pp. 1–5.

[112] M. Ke, Z. Gao, Y. Wu, X. Gao, and R. Schober, "Compressive sensing-based adaptive active user detection and channel estimation: massive access meets massive MIMO", *IEEE Transactions on Signal Processing*, vol. 68, pp. 764–779, 2020.

[113] G. Wunder, Č. Stefanović, P. Popovski, and L. Thiele, "Compressive coded random access for massive MTC traffic in 5G systems", in *2015 49th Asilomar Conference on Signals, Systems and Computers*, IEEE, 2015, pp. 13–17.

[114] M. Kim, N. Kim, W. Lee, and D. Cho, "Deep learning-aided SCMA", *IEEE Communications Letters*, vol. 22, no. 4, pp. 720–723, 2018.

[115] M. Liu, T. Song, and G. Gui, "Deep cognitive perspective: resource allocation for NOMA-based heterogeneous IoT with imperfect SIC", *IEEE Internet of Things Journal*, vol. 6, no. 2, pp. 2885–2894, 2019.

[116] Narengerile and J. Thompson, "Deep learning for signal detection in non-orthogonal multiple access wireless systems", in *2019 UK/ China Emerging Technologies (UCET)*, ISSN: null, Aug. 2019, pp. 1–4. DOI: 10.1109/UCET.2019.8881888.

[117] D. Magrin, C. Pielli, C. Stefanovic, and M. Zorzi, *Enabling LTE RACH collision multiplicity detection via machine learning*, 2018. arXiv: 1805.11482 [cs.IT].

[118] A. C. Cirik, N. Mysore Balasubramanya, and L. Lampe, "Multi-user detection using ADMM-based compressive sensing for uplink grant-free NOMA", *IEEE Wireless Communications Letters*, vol. 7, no. 1, pp. 46–49, Feb. 2018.

[119] H. V. Poor, *An introduction to signal detection and estimation*. Springer Science & Business Media, 2013.

[120] 3GPP, "Evolved Universal Terrestrial Radio Access (E-UTRA); Physical layer procedures", 3rd Generation Partnership Project (3GPP), Technical Specification (TS) 36.213, Mar. 2021, Version 16.5.0. [Online]. Available: https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=2427.

[121] S. Dörner, S. Cammerer, J. Hoydis, and S. Ten Brink, "Deep learning based communication over the air", *IEEE Journal of Selected Topics in Signal Processing*, vol. 12, no. 1, pp. 132–143, 2017.

[122] H. Ye, L. Liang, G. Y. Li, and B.-H. Juang, "Deep learning-based end-to-end wireless communication systems with conditional gans as unknown channels", *IEEE Transactions on Wireless Communications*, vol. 19, no. 5, pp. 3133–3143, 2020.

[123] F. A. Aoudia and J. Hoydis, "Model-free training of end-to-end communication systems", *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 11, pp. 2503–2516, 2019.

[124] F. A. Aoudia and J. Hoydis, "End-to-end learning of communications systems without a channel model", in *2018 52nd Asilomar Conference on Signals, Systems, and Computers*, IEEE, 2018, pp. 298–303.

[125] M. Goutay, F. A. Aoudia, and J. Hoydis, "Deep reinforcement learning autoencoder with noisy feedback", in *2019 International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks (WiOPT)*, IEEE, 2019, pp. 1–6.

[126] S. Cammerer, F. A. Aoudia, S. Dörner, M. Stark, J. Hoydis, and S. Ten Brink, "Trainable communication systems: concepts and prototype", *IEEE Transactions on Communications*, vol. 68, no. 9, pp. 5489–5503, 2020.

[127] F. A. Aoudia and J. Hoydis, "End-to-end learning for OFDM: from neural receivers to pilotless communication", *arXiv:2009.05261 [cs, eess, math]*, Sep. 11, 2020. arXiv: 2009.05261. [Online]. Available: http://arxiv.org/abs/2009.05261 (visited on 09/16/2020).

[128] C. Morin, *Gr-learning*. [Online]. Available: https://github.com/Notou/gr-learning.

[129] J. Harshan and B. S. Rajan, "On two-user gaussian multiple access channels with finite input constellations", *IEEE Transactions on Information Theory*, vol. 57, no. 3, pp. 1299–1327, Mar. 2011, Conference Name: IEEE Transactions on Information Theory, ISSN: 1557-9654. DOI: 10.1109/TIT.2011.2104491.

[130] H. Nikopour and H. Baligh, "Sparse code multiple access", in *2013 IEEE 24th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC)*, IEEE, 2013, pp. 332–336.

[131] J. Harshan and B. S. Rajan, "A novel power allocation scheme for two-user GMAC with finite input constellations", *IEEE Transactions on Wireless Communications*, vol. 12, no. 2, pp. 818–827, Feb. 2013, Conference Name: IEEE Transactions on Wireless Communications, ISSN: 1558-2248. DOI: 10.1109/TWC.2012.122212.120324.

[132] Y. Liang, S. Rini, and J. Kliewer, "Joint constellation and code design for the gaussian multiple access channel", in *2017 51st Asilomar Conference on Signals, Systems, and Computers*, ISSN: 2576-2303, Oct. 2017, pp. 1728–1732. DOI: 10.1109/ACSSC.2017.8335656.

[133] A. Balatsoukas-Stimming, S. Rini, and J. Kliewer, "LDPC coded multiuser shaping for the gaussian multiple access channel", in *2019 IEEE International Symposium on Information Theory (ISIT)*, ISSN: 2157-8117, Jul. 2019, pp. 2609–2613. DOI: 10.1109/ISIT.2019.8849785.

[134] H. Liu, S. Shieh, C. Lin, and P. Chen, "A minimum distance criterion based constellation design for uplink NOMA", in *2019 IEEE 90th Vehicular Technology Conference (VTC2019-Fall)*, ISSN: 2577-2465, Sep. 2019, pp. 1–5. DOI: 10.1109/VTCFall.2019.8891372.

[135] J. Lin, S. Feng, Z. Yang, Y. Zhang, and Y. Zhang, "A novel deep neural network based approach for sparse code multiple access", *arXiv:1906.03169 [cs, eess, math, stat]*, Jun. 4, 2019. arXiv: 1906.03169. [Online]. Available: http://arxiv.org/abs/1906.03169 (visited on 06/20/2019).

[136] Z. Si, S. Wen, and B. Dong, "NOMA codebook optimization by batch gradient descent", *IEEE Access*, vol. 7, pp. 117 274–117 281, 2019, Conference Name: IEEE Access, ISSN: 2169-3536. DOI: 10.1109/ACCESS.2019.2936483.

[137] M. Han, H. Seo, A. T. Abebe, and C. G. Kang, "Deep learning-based multi-user multi-dimensional constellation design in code domain non-orthogonal multiple access", in *2020 IEEE International Conference on Communications Workshops (ICC Workshops)*, ISSN: 2474-9133, Jun. 2020, pp. 1–6. DOI: 10.1109/ICCWorkshops49005.2020.9145347.

[138] S. Zhang, K. Xiao, B. Xiao, Z. Chen, B. Xia, D. Chen, and S. Ma, "A capacity-based codebook design method for sparse code multiple access systems", in *2016 8th International Conference on Wireless Communications Signal Processing (WCSP)*, ISSN: 2472-7628, Oct. 2016, pp. 1–5. DOI: 10.1109/WCSP.2016.7752620.

[139] F. Sun, K. Niu, and C. Dong, "Deep learning based joint detection and decoding of non-orthogonal multiple access systems", in *2018 IEEE Globecom Workshops (GC Wkshps)*, Dec. 2018, pp. 1–5. DOI: 10.1109/GLOCOMW.2018.8644090.

[140] K. Xiao, B. Xia, L. Ma, and Y. Wei, "Design and analysis of probabilistic shaping scheme for uplink nonorthogonal multiple access systems", *IEEE Transactions on Communications*, vol. 67, no. 12, pp. 8818–8833, Dec. 2019, Conference Name: IEEE Transactions on Communications, ISSN: 1558-0857. DOI: 10.1109/TCOMM.2019.2946831.

[141] C. Lin, S. Shieh, T. Chi, and P. Chen, "Optimal inter-constellation rotation based on minimum distance criterion for uplink NOMA", *IEEE Transactions on Vehicular Technology*, vol. 68, no. 1, pp. 525–539, Jan. 2019, Conference Name: IEEE Transactions on Vehicular Technology, ISSN: 1939-9359. DOI: 10.1109/TVT.2018.2881683.

[142] C. Jiang and Y. Wang, "An uplink SCMA codebook design combining probabilistic shaping and geometric shaping", *IEEE Access*, vol. 8, pp. 76 726–76 736, 2020, Conference Name: IEEE Access, ISSN: 2169-3536. DOI: 10.1109/ACCESS.2020.2989448.

[143] S. Wang, H. Yu, Y. Yuan, G. Liu, and Z. Fei, "AI-enhanced constellation design for NOMA system: a model driven method", *China Communications*, vol. 17, no. 11, pp. 100–110, Nov. 2020, Conference Name: China Communications, ISSN: 1673-5447. DOI: 10.23919/JCC.2020.11.009.

[144] M. Kim, N. Kim, W. Lee, and D. Cho, "Deep learning-aided SCMA", *IEEE Communications Letters*, vol. 22, no. 4, pp. 720–723, Apr. 2018, Conference Name: IEEE Communications Letters, ISSN: 1558-2558. DOI: 10.1109/LCOMM.2018.2792019.

[145] R. Xin, Z. Ni, L. Kuang, H. Jia, and P. Wang, "Joint active user and data detection in uplink grant-free NOMA by message-passing algorithm", in *2019 15th International Wireless Communications Mobile Computing Conference (IWCMC)*, ISSN: 2376-6492, Jun. 2019, pp. 126–130.

[146] S. Jiang, X. Yuan, X. Wang, C. Xu, and W. Yu, "Joint user identification, channel estimation, and signal detection for grant-free NOMA", *IEEE Transactions on Wireless Communications*, pp. 1–1, 2020, Conference Name: IEEE Transactions on Wireless Communications, ISSN: 1558-2248. DOI: 10.1109/TWC.2020.3007545.

[147] A. Goldsmith, *Wireless communications*. Cambridge university press, 2005.

[148] R. H. Clarke, "A statistical theory of mobile-radio reception", *Bell system technical journal*, vol. 47, no. 6, pp. 957–1000, 1968.

[149] G. Böcherer, "Achievable rates for probabilistic shaping", *arXiv:1707.01134 [cs, math]*, May 22, 2018. arXiv: 1707.01134. [Online]. Available: http://arxiv.org/abs/1707.01134 (visited on 07/20/2020).

[150] T. M. Cover and J. A. Thomas, *Elements of information theory*. John Wiley & Sons, 2006.

[151] N. Papernot, P. McDaniel, I. Goodfellow, S. Jha, Z. B. Celik, and A. Swami, "Practical black-box attacks against machine learning", in *Proceedings of the 2017 ACM on Asia conference on computer and communications security*, 2017, pp. 506–519.

[152] *Gr-bokehgui*. [Online]. Available: https://github.com/gnuradio/gr-bokehgui.

[153] O. Gungor, C. E. Koksal, and H. El Gamal, "An information theoretic approach to rf fingerprinting", in *2013 Asilomar Conference on Signals, Systems and Computers*, IEEE, 2013, pp. 61–65.

[154] P. Hao, "Wireless device authentication techniques using physical-layer device fingerprint", *Electronic Thesis and Dissertation Repository*, Nov. 2015. [Online]. Available: https://ir.lib.uwo.ca/etd/3440.

[155] O. Avatefipour, A. Hafeez, M. Tayyab, and H. Malik, "Linking received packet to the transmitter through physical-fingerprinting of controller area network", in *2017 IEEE Workshop on Information Forensics and Security (WIFS)*, IEEE, 2017, pp. 1–6.

[156] G. Baldini, C. Gentile, R. Giuliani, and G. Steri, "Comparison of techniques for radiometric identification based on deep convolutional neural networks", *Electronics Letters*, vol. 55, no. 2, pp. 90–92, 2019.

[157] Q. Wu, C. Feres, D. Kuzmenko, D. Zhi, Z. Yu, X. Liu, and X. Liu, "Deep learning based RF fingerprinting for device identification and wireless security", *Electronics Letters*, vol. 54, no. 24, pp. 1405–1407, 2018, ISSN: 0013-5194. DOI: 10.1049/el.2018.6404.

[158] K. Sankhe, F. Restuccia, S. D'Oro, T. Jian, Z. Wang, A. Al-Shawabka, J. Dy, T. Melodia, S. Ioannidis, and K. Chowdhury, "Impairment shift keying: covert signaling by deep learning of controlled radio imperfections", p. 6,

[159] S. Karunaratne, E. Krijestorac, and D. Cabric, "Penetrating RF fingerprinting-based authentication with a generative adversarial attack", *arXiv:2011.01538 [cs, eess]*, Nov. 3, 2020. arXiv: 2011.01538. [Online]. Available: http://arxiv.org/abs/2011.01538 (visited on 02/24/2021).

[160] F. Restuccia, S. D'Oro, A. Al-Shawabka, B. C. Rendon, K. Chowdhury, S. Ioannidis, and T. Melodia, "Hacking the waveform: generalized wireless adversarial deep learning", *arXiv:2005.02270 [cs, eess]*, May 5, 2020. arXiv: 2005.02270. [Online]. Available: http://arxiv.org/abs/2005.02270 (visited on 02/24/2021).

[161] C. M. Bishop, *Pattern recognition and machine learning*. springer, 2006.

[162] K. P. Murphy, *Machine learning: a probabilistic perspective*. MIT press, 2012.

[163] H. Yuan, H. Yu, S. Gui, and S. Ji, "Explainability in graph neural networks: a taxonomic survey", *arXiv preprint arXiv:2012.15445*, 2020.

[164] O. O. Oyerinde, "Compressive sensing algorithms for multiuser detection in uplink grant free NOMA systems", in *2019 IEEE 89th Vehicular Technology Conference (VTC2019-Spring)*, ISSN: 1090-3038, Apr. 2019, pp. 1–6.

[165] G. Gui, H. Huang, Y. Song, and H. Sari, "Deep learning for an effective nonorthogonal multiple access scheme", *IEEE Transactions on Vehicular Technology*, vol. 67, no. 9, pp. 8440–8450, Sep. 2018.

[166] F. Zhou, Y. Wu, Y.-C. Liang, Z. Li, Y. Wang, and K.-K. Wong, "State of the art, taxonomy, and open issues on cognitive radio networks with NOMA", *IEEE Wireless Communications*, vol. 25, no. 2, pp. 100–108, Apr. 2018, Focus on Power NOMA only.

[167] Y. Zhang, Q. Guo, Z. Wang, J. Xi, and N. Wu, "Block sparse bayesian learning based joint user activity detection and channel estimation for grant-free NOMA systems", *IEEE Transactions on Vehicular Technology*, vol. 67, no. 10, pp. 9631–9640, Oct. 2018.

[168] K.-H. Ngo, S. Yang, M. Guillaud, and A. Decurninge, "Joint constellation design for the two-user non-coherent multiple-access channel", *arXiv:2001.04970 [cs, math]*, Jan. 16, 2020. arXiv: 2001.04970. [Online]. Available: http://arxiv.org/abs/2001.04970 (visited on 02/11/2020).

[169] M. Taherzadeh, H. Nikopour, A. Bayesteh, and H. Baligh, "SCMA codebook design", *2014 IEEE 80th Vehicular Technology Conference (VTC2014-Fall)*, pp. 1–5, Sep. 2014. DOI: 10.1109/VTCFall.2014.6966170. arXiv: 1408.3653. [Online]. Available: http://arxiv.org/abs/1408.3653 (visited on 07/31/2020).

[170] M. Han, H. Seo, A. T. Abebe, and C. G. Kang, "Deep learning-based codebook design of multi-user multi-dimensional modulation for code-domain NOMA achieving a single-user performance", *IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS*, p. 30, 2020.

FOLIO ADMINISTRATIF

THESE DE L'UNIVERSITE DE LYON OPEREE AU SEIN DE L'INSA LYON

NOM : MORIN

DATE de SOUTENANCE : 22/07/2021

Prénoms : Cyrille

TITRE : Approches d'apprentissage profond pour la détection en couche physique de télécommunications multi-accès

NATURE : Doctorat

Numéro d'ordre : 2021LYSEI049

École doctorale : Électronique, Électrotechnique et Automatique

Spécialité : Traitement du signal et des images

RESUME :
Les tendances actuelles pointent vers une accélération de l'augmentation du nombre d'objets cherchant à accéder au spectre radio, à la fois par la démocratisation des objets grand public, smartphones, ordinateurs, montres connectées,... et par le déploiement d'objets et capteurs connectés.
Des avancées technologiques, protocolaires et législatives augmentent les bandes de fréquence disponibles en ouvrant l'accès à la zone des GHz, mais la densité des objets communicant sur le spectre tend quand même à augmenter.
L'accès multiple à une ressource radio partagée mène à des situations qui sont à la fois complexes à modéliser et à aborder avec les algorithmes actuels, et c'est particulièrement vrai pour les taches de type détection présentes au niveau des couches physiques des communications sans fil.
Les algorithmes d'apprentissage profond sont particulièrement utiles dans ce type de situation, sans modèle ou avec des algorithmes existant peu pratiques, pour peu qu'une grande quantité de données soit disponible pour entrainer les réseaux de neurones.
Cette thèse vise à adapter l'outil de l'apprentissage profond aux problèmes de détection de la couche physique, à différentes étapes de la chaine de décodage.
D'abord par le problème de la détection d'origine d'un paquet reçu, commençant par l'identification de caractéristiques matérielles d'un l'objet émetteur, puis étendant ce scénario à un ensemble d'objets actifs simultanément.
L'étape suivant la détection de l'origine d'un paquet est la détection des bits, pour décoder les messages transmis.
Dans ce cadre, l'apprentissage profond est employé pour apprendre des constellations permettant une détection efficace des bits dans un scénario multi-accès non orthogonal à deux utilisateurs.
Les données servant à l'apprentissage des réseaux de neurones impliqués dans cette thèse sont récoltées soit dans des modèles simulés, soit par des expériences implémentées dans l'équipement de radio logicielle FIT/CorteXlab.

MOTS-CLÉS : Apprentissage profond, traitement du signal, radio logicielle, couche physique, communications sans fil

Laboratoire (s) de recherche : Laboratoire CITI

Directeur de thèse: Jean-Marie GORCE

Président de jury : Christophe MOY

Composition du jury : Marwa CHAFII, Symeon CHATZINOTAS, Catherine DOUILLARD, Jean-Marie GORCE, Jakob HOYDIS, Christophe MOY, Marco DI RENZO, Leonardo SAMPAIO CARDOSO