



Automatic modulation classification: A study in robustness against adversarial attacks

A van der Merwe

 **orcid.org/ 0009-0001-5032-7090**

Dissertation accepted in fulfilment of the requirements for the degree *Master of Engineering in Computer and Electronic Engineering* at the North-West University

Supervisor: Prof ASJ Helberg

Graduation: July 2026

Declaration

I, Arnold van der Merwe, hereby declare that the dissertation entitled “Automatic modulation classification: A study in robustness against adversarial attacks” is my own original work and has not already been submitted to any other university or institution for examination.

A. van der Merwe

Signed on the 27th day of November 2025 at
Potchefstroom.

Acknowledgements

1. I would like to express my sincere gratitude to Prof. Albert Helberg for his supervision, support, and encouragement throughout my Master's studies. I am indebted to him for consistently challenging me to raise my standards and for pushing me to achieve more than I thought possible.
2. Thank you to my family for their support during my studies. Although they did not always immediately comprehend the intricacies of my academic research (neither did I at times), their unwavering faith in my ability to succeed provided the grounding I needed during the long hours.
3. To my fellow students at the North-West University, I extend a heartfelt thank you. The camaraderie we shared over the past two years proved to be an essential source of encouragement and perspective. I will especially miss our Friday lunches at the Drakenstein, where we shared both our wins and our losses.

Abstract

Automatic modulation classification (AMC) is the systematic identification of the modulation scheme used by signals without any prior knowledge of the signal. The AMC process can be rather difficult, depending on the signal channel which a transmitted signal has been subjected to. Historically, AMC has been approached using a feature-based, likelihood-based, or clustering approach, which we consider as non-deep learning (NDL) methods. In more recent research, the focus has shifted to deep learning (DL) methods using deep neural networks (DNNs) for AMC, as they can outperform older NDL methods in more complex signal channels. However, this good performance from the DNNs comes with the vulnerability to adversarial attacks. Adversarial attacks are subtle perturbations added to the input signals, which are visually imperceptible, but cause the DNN to misclassify the modulation type of the input signal. White-box adversarial attacks have complete knowledge of the DNN's architecture and parameters, whereas with black-box attacks this information is not available.

In this study, we investigate the robustness of different AMC methods against white-box adversarial attacks. Our selected NDL methods consist of the quasi-hybrid likelihood ratio test (QHLRT), a clustering classifier, and a k-nearest neighbour (KNN) using higher-order cumulants. These approaches are selected to cover the different NDL approaches to AMC. We compare these NDL methods against each other and against a DL method, namely the parameter estimation and transformation-based CNN-GRU deep neural network (PET-CGDNN), in the presence and absence of adversarial attacks, using a dataset that simulates an additive white Gaussian noise (AWGN) signal channel. The white-box adversarial attacks consists of the fast gradient method (FGM), the projected gradient descent (PGD), and the DeepFool attacks. We use two different approaches to our adversarial attacks, that is a fixed-size attack and a varying-size attack that scales with a perturbation-to-noise ratio (PNR). We generate adversarial attacks using a trained PET-CGDNN model. The adversarial attacks cause a significant deterioration in the classification performance of the DNN. With the FGM and PGD attacks, we find limited transferability from the attacks to the NDL methods, as they do not result in significant changes in classification performance. Against the DeepFool attacks, the classification performance of all our classifiers is significantly deteriorated. Furthermore, we propose the design of a hybrid classifier that combines the

DL and NDL techniques, through a support vector machine (SVM) which acts as a switching mechanism, deciding whether to use the DNN or the NDL method on an input. Our hybrid classifier comprises the PET-CGDNN and the KNN using higher-order cumulants. The hybrid classifier shows comparable performance to the DNN when classifying unperturbed signals. The hybrid classifier shows a clear improvement in robustness against FGM attacks, but only a modest improvement against PGD attacks. Only a marginal improvement in robustness is observed against DeepFool attacks.

Keywords: Automatic modulation classification; Adversarial attacks; Deep neural networks; Likelihood ratio test; Features-based classification

Contents

List of Figures	viii
List of Tables	x
List of Acronyms	xi
List of Symbols	xiii
1 Introduction	1
1.1 Background	1
1.2 Problem statement	3
1.3 Research questions	3
1.4 Research objectives	4
1.5 Scope and assumptions	4
1.6 Dissertation overview	4
1.7 Publications	5
2 Background	7
2.1 Introduction	7
2.2 Wireless systems	7
2.2.1 Modulation of signals	8
2.2.2 Signal channels	11
2.3 Non-deep learning methods	13
2.3.1 Likelihood-based methods	14
2.3.2 Feature-based methods	16
2.3.3 Clustering	18
2.4 Deep learning methods	19
2.4.1 Datasets	19
2.4.2 Types of deep neural networks	20
2.4.3 Notable neural networks	24
2.5 Adversarial attacks	28
2.5.1 Fast gradient sign method white-box attack	30
2.5.2 Carlini & Wagner white-box attack	31
2.5.3 Projected gradient descent white-box attack	31
2.5.4 Universal adversarial perturbation	32

2.5.5	DeepFool	32
2.6	Chapter summary	33
3	Dataset and adversarial attacks	35
3.1	Introduction	35
3.2	Description of dataset	35
3.3	Selection and generation of white-box adversarial attacks	36
3.3.1	Fast gradient method	37
3.3.2	Projected gradient descent	38
3.3.3	DeepFool	38
3.3.4	Scaling adversarial attacks	40
3.4	Visualisation of dataset	41
3.5	Analysis of DeepFool attacks	43
3.6	Chapter summary	44
4	Investigation of DL techniques	46
4.1	Introduction	46
4.2	PET-CGDNN architecture	46
4.3	Verification of implementation	48
4.4	Training process	49
4.4.1	Hyperparameter tuning	50
4.4.2	Training with selected parameters	51
4.5	Results	52
4.5.1	Performance on unperturbed signals	52
4.5.2	Performance on attacked signals	53
4.6	Chapter summary	55
5	Investigation of NDL techniques	57
5.1	Introduction	57
5.2	NDL methods	57
5.2.1	Clustering	57
5.2.2	QHLRT	61
5.2.3	KNN with higher-order cumulants	64
5.3	Verification of implementation	66
5.3.1	QHLRT	67
5.3.2	KNN with higher-order cumulants	68
5.4	Training process of KNN	70
5.5	Results	70
5.5.1	Performance on unperturbed signals	70
5.5.2	Performance on attacked signals	71
5.6	Chapter summary	73

6	Combination of DL and NDL techniques	76
6.1	Introduction	76
6.2	Design of hybrid classifier	76
6.2.1	Neural rejection	77
6.2.2	Support vector machines	78
6.2.3	Configuration	80
6.3	Training process of SVM	81
6.4	Tuning the SVM	82
6.5	Results	85
6.5.1	Performance on unperturbed signals	85
6.5.2	Performance on attacked signals	86
6.6	Chapter summary	88
7	Conclusion	92
7.1	Introduction	92
7.2	Reflection on research objectives	92
7.2.1	Research methodology	93
7.3	Reflection on research questions	94
7.4	Future work	95
7.5	In closing	96
	Bibliography	97
A	Supporting figures for investigation of NDL techniques	104

List of Figures

2.1	Diagram showing simplified wireless communication system.	8
2.2	Constellation diagrams of Phase-shift keying (PSK) and quadrature amplitude modulation (QAM) modulation techniques showing symbol positions.	11
2.3	Constellation diagrams of PSK and QAM modulation techniques with AWGN at a signal-to-noise ratio (SNR) of 20 dB.	13
2.4	Process of likelihood-based classification.	14
2.5	Generic multi-layer perceptron (MLP) architecture.	21
2.6	Generic convolutional neural network (CNN) architecture for a classifier showing convolutional layers, pooling layers and fully connected layers.	22
2.7	Generic recurrent neural network (RNN) architecture showing the folded and unfolded version of the architecture.	23
2.8	General scenario of adversarial attacks in AMC.	29
2.9	Fast gradient sign method (FGSM) attack illustration on DNN gradients.	31
2.10	PGD attack illustration on DNN gradients	32
2.11	DeepFool attack illustrating locally linearised decision boundaries.	33
3.1	8-PSK signal frames with a SNR of 20 dB with adversarial attacks.	41
3.2	16-QAM signal frames with a SNR of 20 dB with adversarial attacks.	42
3.3	Cumulative distribution function of the ϵ of DeepFool attacks.	43
3.4	Bounded DeepFool attacks with $\epsilon \leq 1.5$	44
4.1	PET-CGDNN architecture.	47
4.2	Classification performance of PET-CGDNN on RML2016.10a dataset for verification.	50
4.3	Training of PET-CGDNN using selected hyperparameters.	52
4.4	PET-CGDNN classification performance on unperturbed signals as SNR varies.	53
4.5	PET-CGDNN classification performance on fixed-size attacked signals as SNR varies.	54
4.6	PET-CGDNN classification performance on signals attacked with varying-size attacks for specific SNRs as PNR varies.	55
4.7	3D surface of PET-CGDNN's classification performance on varying-size attacks.	56
5.1	Differentiating between PSK and QAM modulation schemes for clustering classifier.	60
5.2	Process of modulation classification using clustering.	61

5.3	Classification performance of QHLRT classifier as SNR varies for verification.	68
5.4	Classification performance of KNN classifier that uses higher-order cumulants as SNR varies for verification.	69
5.5	Classification performance of NDL classifiers as SNR varies on unperturbed signals.	71
5.6	Confusion matrix of classification performance of QHLRT on signals without adversarial attacks at 22 dB SNR.	72
5.7	Classification performance of NDL classifiers on fixed-size attacks as SNR varies.	73
5.8	Classification performance of NDL classifiers on varying-size attacks as PNR varies for specific SNRs.	74
6.1	Conceptual visualisation of neural rejection system on a feature space with three distinct classes. Bounding circles are drawn to indicate rejection threshold.	78
6.2	Hybrid classifier architecture.	81
6.3	Classification accuracy using rejection thresholds determined using specific percentiles for the hybrid classifier on signals with FGM attacks.	83
6.4	Classification accuracy using thresholds from specific percentiles for hybrid classifier on signals with PGD attacks.	84
6.5	Classification performance of the different configurations of the hybrid classifier as SNR varies on unperturbed signals.	86
6.6	Classification performance of the different configurations of the hybrid classifier on fixed-size attacks as SNR varies.	87
6.7	Classification performance of the different configurations of the hybrid classifier and the PET-CGDNN on signals attacked with varying-size attacks for specific SNRs as PNR varies.	89
6.8	Difference between performance of different configurations of hybrid classifier and PET-CGDNN on signals attacked with varying-size attacks as SNRs and PNR varies.	90
A.1	3D surface of the classification performance of NDL classifiers on varying-size attacks as SNR and PNR varies.	104

List of Tables

2.1	Datasets for training and evaluating DNNs for AMC	20
4.1	Hyperparameter tuning results for PET-CGDNN.	51
5.1	Parameter $b^{(i)}$ for different modulation orders of PSK and QAM.	62
6.1	Hyperparameter tuning results for SVM of hybrid classifier on reduced training set.	82
6.2	F1 scores across SNR levels and rejection threshold percentiles for FGM attacks on the validation set. The highest F1 score for each row is shown in bold.	83
6.3	F1 scores across SNR levels and percentiles for PGD attacks on the validation set. The highest F1 score for each row is shown in bold.	84

List of Acronyms

ALRT	average likelihood ratio test
AMC	automatic modulation classification
ART	adversarial robustness toolbox
ASK	amplitude-shift keying
AWGN	additive white Gaussian noise
BPSK	binary phase-shift keying
CDP	cyclic domain profile
CR	cognitive radio
CW	Carlini & Wagner
CNN	convolutional neural network
CWT	continuous wavelet transform
DBI	Davies-Bouldin index
DL	deep learning
DNN	deep neural network
FGM	fast gradient method
FGSM	fast gradient sign method
FSK	frequency-shift keying
GCN	graph convolutional network
GLRT	generalised likelihood ratio test
GRU	gated recurrent unit
HLRT	hybrid likelihood ratio test
I	in-phase
KNN	k-nearest neighbour
LR	learning rate
LSTM	long short-term memory

ML	maximum likelihood
MLP	multi-layer perceptron
MoM	method-of-moments
NDL	non-deep learning
OOK	on-off keying
PAM	pulse-amplitude modulation
PET-CGDNN	parameter estimation and transformation-based CNN-GRU deep neural network
PGD	projected gradient descent
PNR	perturbation-to-noise ratio
PSR	perturbation-to-signal ratio
PSK	phase-shift keying
Q	quadrature
QAM	quadrature amplitude modulation
QHLRT	quasi-hybrid likelihood ratio test
RBF	radial basis function
ReLU	rectified linear unit
RNN	recurrent neural network
SACAIR	Southern African Conference for Artificial Intelligence Research
SATNAC	Southern Africa Telecommunication Networks and Applications Conference
SCRNN	sequential convolutional recurrent neural network
SCF	spectral correlation function
SNR	signal-to-noise ratio
SVM	support vector machine
S-transform	Stockwell transform
UAP	universal adversarial perturbation

List of Symbols

$A(t)$	Instantaneous amplitude of continuous time signal (A/P format)
$b^{(i)}$	Distinct parameter used for QHLRT
C	Regularisation parameter for SVM
$\mathbf{c}_i$	Centroid of the i -th cluster
C_i	Cluster with index i
c_n	n th-order cumulant (for real-valued variables)
C_t	Cumulant generating function
C_{xy}	(x, y) -th order cumulant (for complex-valued signals)
d	Distance function
$\mathcal{D}$	Dataset containing $\mathbf{x}_i, y_i$ pairs
D_{ij}	Inter-cluster distance between cluster i and cluster j
f_c	Signal carrier frequency
$I(t)$	Amplitude of continuous in-phase signal component
$\mathbb{I}$	Indicator function
J	Cost function
K	Number of signal samples
$\mathcal{L}$	Likelihood function
L	Loss function
$\hat{l}$	Index of the class corresponding to the closest linearised decision boundary
$\mathcal{L}^{(i)}$	Likelihood of the i -th modulation scheme
L_p	p -norm of a vector

$\hat{\mathcal{M}}$	Estimated modulation type
M	Modulation order
$\mathcal{M}^{(i)}$	Modulation type at index i in $\mathbf{R}$
M_{pq}	(p, q) -th order moment
$\hat{M}_{pq}$	Estimated (p, q) -th order moment
N	Signal channel noise variance/Total number of clusters/Number of training examples
$\mathbf{n}$	Additive white Gaussian noise (AWGN) vector
$\mathcal{N}_k$	Set of indices of k nearest neighbours for KNN
$\hat{N}^{(i)}$	Estimated channel noise for modulation type i
$\mathcal{P}$	Probability function
P_{noise}	Power of noise
P_{signal}	Power of signal
$Q(t)$	Amplitude of continuous quadrature signal component
$\mathbf{R}$	Pool of possible modulation schemes
$r[k]$	Received signal symbol at index k
$\bar{R}$	DBI score
$\hat{\mathbf{r}}[k]$	Discrete received signal vector after inverse transformation at index k
$\mathbf{r}$	Discrete received signal vector
$\hat{\mathbf{r}}$	Discrete received signal vector after inverse transformation
R_{ij}	Similarity measure between cluster i and cluster j
$s(t)$	Continuous time signal
$\mathbf{s}$	Discrete modulated signal vector (before transmission)
S_i	Dispersion measure of the i -th cluster
$s_k^{(i)}$	k -th signal constellation point of the i -th modulation scheme
s_m	Modulated signal symbol at position m
T_i	Number of points in the i -th cluster
w_0	Bias term

$\mathbf{w}$	Weight vector
$\mathbf{w}'_y$	Difference between gradient of class y and the gradient of the predicted class $\hat{y}$
X	Real-valued random variable
$\mathbf{x}$	Input feature vector
$\mathbf{x}_i$	i -th input feature vector/feature vector at iteration i
y	True class label
$\hat{y}$	Predicted class label
$\mathcal{Y}$	Class label space for $\mathcal{D}$
y_i	i -th class label
α	Signal channel gain/PGD step size
$\hat{\alpha}^{(i)}$	Estimated channel gain for modulation type index i
β_i	i -th coefficient for the weight vector
$\hat{\delta}$	Final cumulative DeepFool adversarial perturbation
δ_i	Cumulative DeepFool adversarial perturbation at iteration i
ϵ	Scaling factor of adversarial perturbation
γ	Kernel coefficient for a SVM
κ	Kernel function for a SVM
μ	Mean
$\hat{\phi}$	Estimated phase offset
$\phi(t)$	Instantaneous phase of a continuous time signal (A/P format)
ϕ	Signal channel phase response
ϕ_i	Phase shift corresponding to i -th signal symbol
$\hat{\phi}^{(i)}$	Estimated channel phase for modulation type i
σ	Standard deviation of noise/Standard deviation
θ	Deep neural network model parameters

Chapter 1

Introduction

1.1 Background

Wireless communication systems are increasingly becoming important in the modern world, with a focus on reliability, spectral efficiency, and other key metrics. One of the challenges in this domain is to identify the modulation type used by a transmitted signal in varying signal environments. The modulation type of a signal refers to the method that is used to alter the signal to convey information. Automatic modulation classification (AMC) plays a crucial role in wireless communication systems by systematically determining the modulation type of unknown signals.

One of the applications of AMC lies in cognitive radios (CRs). A CR is an intelligent radio system that is capable of changing the modulation type of a signal to maximise the efficient use of the frequency spectrum [1]. It achieves this by sensing its environment and reconfiguring the transmission channel as needed to make use of "spectrum holes" [1]. Spectrum holes are temporal and spatial gaps in the frequency spectrum that are not currently in use. By adjusting its transmission parameters, the CR can make use of these gaps without interfering with licensed users (primary users) of the spectrum, thereby maximising data transmission rates. The transmission parameters that can be changed include factors such as the signal modulation scheme, which the CR can dynamically adjust to fully utilise the spectrum.

The CR uses AMC to assist in spectrum sensing to detect spectrum holes for unlicensed users (secondary users). The method relies on recognising that all primary users use a specific modulation scheme for transmission, so identifying the modulation scheme employed by a primary user indicates the presence of a primary user's signal in the channel [2], [3]. Several different spectrum sensing methods have been developed that has AMC as a core component of their functionality [2], [3]. AMC is used to help differentiate between primary

and secondary users by assisting the CR in its decision-making process.

There are mainly two approaches to AMC, namely deep learning (DL) and non-deep learning (NDL) methods [4], [5]. DL methods, a comparatively recent approach to AMC, utilise deep neural network (DNN) architectures trained on signal patterns across various modulation types. NDL methods for AMC are often grouped into feature-based, likelihood-based, and clustering techniques. Feature-based techniques use the features that become apparent with the different modulation types to classify the modulation type, whereas likelihood-based techniques use statistical tests to determine the modulation type used. Clustering techniques for AMC use signal groupings in a constellation diagram to determine the signal modulation type.

Currently, DL methods are able to vastly outperform NDL methods in classification accuracy and can be used in more challenging signal channels where NDL methods struggle [6], [7]. Generally, sufficiently large DNNs are able to capture more complex signal patterns by automatically extracting the relevant features compared to NDL methods which interact with the signals mainly through previously defined features or processes. Furthermore, DL methods can easily adapt to a different signal channel if they are properly trained on representative signals.

However, the effectiveness of DL methods for AMC comes with a caveat, as they are vulnerable to adversarial attacks. Adversarial attacks are subtle perturbations that are added to the input of a DNN which causes the model to misclassify the input [8]. They are also commonly referred to as evasion attacks. Adversarial attacks are often grouped as white-box, black-box or gray-box attacks depending on the knowledge available of the DNN model and its parameters [8]. White-box attacks have complete access to the DNN model and its parameters, while black-box attacks have no access to the DNN model, and gray-box attacks have partial access to the DNN model. These attacks can severely degrade the performance of DL methods, as evidenced by a significant decrease in the classification accuracy [9]–[11]. Additionally, adversarial attacks are efficient, as their minimal spectrum use enables attacks to have a stealthy and energy-efficient nature [12].

Adversarial defence techniques have been evaluated for the DL methods for AMC which aim to make the method more robust against adversarial attacks [13]. These defence techniques can make the method more resistant to attack, but some vulnerability still remains, especially in the case of white-box attacks [13]. Even when defence techniques are introduced, attacks can still be crafted later, potentially diminishing the effectiveness of the defence techniques.

Limited research has been carried out on the transferability of adversarial attacks generated on a DNN to the NDL methods for AMC. To our knowledge, only Usama *et al.* [14] have investigated the effects that adversarial perturbations have on NDL classifiers. They tested the white-box Carlini & Wagner (C&W) attack against a DNN, k-nearest neighbour (KNN), support vector machine (SVM), Naive Bayes, linear discriminant analysis, decision tree, and

a random forest. Adversarial perturbations were generated on the DNN using a well-known dataset that is commonly used for AMC, that is the RML2016.10a. Their findings indicate all of the classifiers are vulnerable to the C&W attacks [14]. However, their work did not investigate how scaling the attack strength affects the classification performance and what effect other attack methods have on these classifiers. Furthermore, it appears they only used raw signal samples for the NDL methods as they do not explicitly state that they used an specific signal features. To our knowledge, the transferability of adversarial attacks to likelihood-based or clustering-based AMC methods has not been evaluated.

1.2 Problem statement

Automatic modulation classification (AMC) is an important low-level process in wireless communication systems that enables the efficient use of spectrum space by identifying different types of signal modulation as needed. Advanced neural networks like convolutional neural networks (CNNs) and recurrent neural networks (RNNs) have shown considerable effectiveness in AMC [4], [15], [16], achieving remarkable success over traditional approaches such as likelihood-based methods, feature-based methods, and clustering methods. Even though these DNNs are highly effective in AMC, they are susceptible to adversarial attacks that add subtle perturbations to the modulated signal, causing incorrect classification of the modulation type [8], [9]. The robustness of AMC methods against attacks is crucial as wireless communication systems must be reliable. Adversarial defence techniques can improve the robustness of DNNs against adversarial attacks, but the vulnerability still remains [13]. Since adversarial attacks are crafted specifically for the DNNs and how they operate, they might not have the same effectiveness against NDL methods. The transferability of adversarial attacks to NDL methods in AMC has not been extensively investigated in the available literature.

1.3 Research questions

The defined research problem raises the following questions that are focused on:

- Are adversarial attacks transferable to NDL methods for AMC and to what extent are they transferable?
- Can NDL techniques be used in conjunction with DL methods to create a new AMC approach that has improved robustness against adversarial attacks?

1.4 Research objectives

The research problem and questions lead to these key objectives that are deemed essential to effectively explore the problem:

- Select and recreate a DNN for AMC that is well-known and well-regarded. Select and recreate various NDL methods for AMC. Compare these AMC methods with and without adversarial attacks present and evaluate their robustness in classification accuracy in an additive white Gaussian noise (AWGN) channel.
- Develop a new method that utilises both DNN and NDL techniques that demonstrates greater robustness against adversarial attacks and has comparable performance to existing DNNs in AMC.

1.5 Scope and assumptions

The scope of the research is limited as follows to ensure a focused and in-depth analysis of the research problem:

- The research is limited to a simple AWGN channel, assuming channel equalisation has already been performed.
- Only Single Input Single Output (SISO) channels are used.
- Only an existing DNN that has been proposed and well-cited for AMC is used.
- This research is not focused on applying new DL techniques to improve AMC performance against adversarial attacks.

1.6 Dissertation overview

The dissertation has the following structure:

- Chapter 2 gives an overview of wireless signals, NDL methods for AMC, DL methods for AMC, and adversarial attacks. The context in which adversarial attacks are used in AMC is discussed.
- Chapter 3 details the dataset generated, which simulates an AWGN channel, and the adversarial attacks used for the study. All AMC methods in the study are evaluated using this dataset.

- Chapter 4 investigates a DL method for AMC, which is the parameter estimation and transformation-based CNN-GRU deep neural network (PET-CGDNN). The training and hyperparameter tuning of the PET-CGDNN is provided. The robustness of this DL model against unperturbed and perturbed signals is evaluated.
- Chapter 5 investigates NDL methods for AMC. The NDL methods comprise the quasi-hybrid likelihood ratio test (QHLRT), a clustering classifier, and a KNN classifier using higher-order cumulants. The classifiers are compared with each other and with the PET-CGDNN to assess their robustness to adversarial attacks and relative performance.
- Chapter 6 proposes a hybrid classifier that combines the PET-CGDNN and the KNN using higher-order cumulants through a neural rejection system. The hybrid classifier is compared with the other classifiers to determine whether it has improved robustness against the adversarial attacks and if its performance against unperturbed signals is in line with the PET-CGDNN's performance.
- Chapter 7 concludes the study by reflecting back on the research objectives and the research questions to examine how the study addressed the research problem.

1.7 Publications

The research conducted in this study led to the following publications:

- *"Modulation classification accuracy of a clustering and a CNN technique under FGSM adversarial attacks"* [17] presented at Southern Africa Telecommunication Networks and Applications Conference (SATNAC) 2024.
- *"Hybrid Automatic Modulation Classification for Increased Robustness under White-Box Adversarial Attacks"* [18] presented at Southern African Conference for Artificial Intelligence Research (SACAIR) 2025 and published in the Communications in Computer and Information Science.
- *"Comparison of robustness of modulation classifiers against white-box adversarial attacks"* [19] presented at SATNAC 2025.

The results from the first paper were obtained during the initial assessment of whether the study topic was suitable. Chapter 5 contains the clustering technique used in the paper. The second paper presents one of the main parts of the study, which is the design of a hybrid classifier that combines DL and NDL techniques. It compares the different classifiers, excluding the clustering technique, against varying-size adversarial attacks. Chapters 3, 4, 5, and 6 are summarised in the paper. The final paper summarises the work from Chapters 3,

4, and 5, comparing various AMC methods against fixed-size adversarial attacks, excluding the hybrid classifier.

These publications arose from work done for the M.Eng degree between 2024 and 2025. This work has not been submitted, in full or partially, for any other qualification.

Chapter 2

Background

2.1 Introduction

In this chapter, we introduce a simplified wireless system used to investigate AMC, the modulation schemes used to encode information into signals, and the signal channels over which these signals are transmitted. The operation and theory behind the NDL AMC approaches of likelihood-based methods, feature-based methods, and clustering are explored. Furthermore, a similar discussion occurs with the DL approaches, focusing on the available datasets, the different types of DNNs, and the related work conducted for these DL approaches. Adversarial attacks are also introduced by examining the scenario in which they occur and the different methods that can be used to implement these attacks.

2.2 Wireless systems

A typical wireless communication system consists of a transmitter and a receiver, with the goal of transmitting information from one end to the other. The transmitter receives this input information or signal and processes it through signal modulation. Signal modulation involves altering one or more properties of the signal, such as its amplitude, phase, or frequency, to encode information. The resulting modulated signal is transmitted over a signal channel. However, as the signal propagates through the channel, it is subject to numerous negative effects due to the real world environment through which it travels. These channel effects can consist of noise, fading, interference, or other effects. The signal that arrives at the receiver can differ significantly from the transmitted signal depending on the channel.

In the scenario considered in this study, the receiver does not have prior knowledge of which

signal modulation was used by the transmitter. The receiver therefore uses a classification process known as automatic modulation classification (AMC) to determine the modulation type of the received signal. After the classifier has identified the modulation type that is being used, the data can be extracted from the modulated signal by demodulation. Ideally, the output signal of the communication system is identical to the input signal. Figure 2.1 illustrates this system.

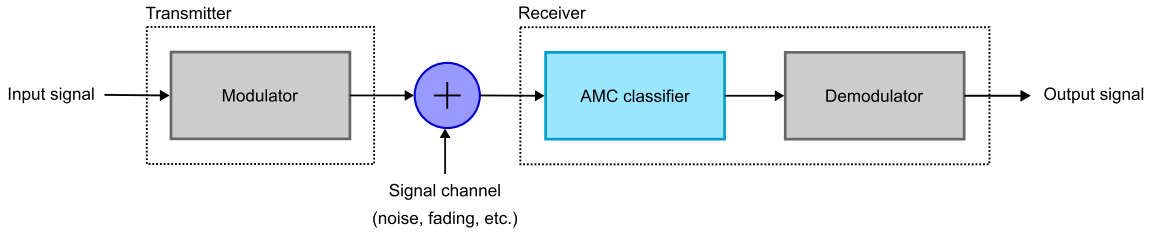


Figure 2.1: Diagram showing simplified wireless communication system.

Note that the described communication system skips many steps that are found in real-world systems, such as channel equalisation, source encoding, channel encoding, and other processes. These processes are not in scope for this study, but it is important to keep them in mind. The described communication system is a simplified system that only consists of the steps that are considered crucial for investigating AMC.

The symbols of the received signal can be mathematically represented as follows [20]. In Equation 2.1, $\mathbf{r}$ is the vector of received signal symbols affected by the channel conditions and noise, such that $\mathbf{r} = [r[1], r[2], \dots, r[K]]^T$ and K is the number of symbols transmitted. Similarly, $\mathbf{s} = [s[1], s[2], \dots, s[K]]^T$ is the vector of transmitted symbols and $\mathbf{n} = [n[1], n[2], \dots, n[K]]^T$ is the vector of noise added to each symbol due to the signal channel. The variables α and ϕ represent the channel amplitude and phase response, respectively. Note that the channel phase also includes the carrier phase offset.

$$\mathbf{r} = \alpha e^{j\phi} \mathbf{s} + \mathbf{n} \quad (2.1)$$

2.2.1 Modulation of signals

There are several different signal modulation techniques that can be used to modulate a wireless signal. Typical digital signal modulation techniques that can be used are amplitude-shift keying (ASK), frequency-shift keying (FSK), phase-shift keying (PSK), and quadrature amplitude modulation (QAM) [21, pp. 420–449, 521–527]. Each of these techniques modifies the signal's amplitude, frequency, or phase to represent the transmitted data. A carrier wave, which is typically a constant high-frequency wave, is used as the base on which the data that need to be transmitted can be superimposed as part of the modulation

technique. Digital modulation techniques are typically used instead of analogue modulation techniques, as signal fidelity can be better controlled [21, p. 27]. Digital modulation techniques enable efficient use of channel bandwidth, can be regenerated in long-distance transmission to help eliminate noise, and are cheaper to implement [21, p. 27].

The modulation order M refers to the number of distinct symbols used by the modulation type, where a higher order typically allows for a higher data rate, because each symbol can represent more bits, but is less resistant to noise and interference in the signal channel [21, p. 548]. M is constrained to be a power of 2, that is M must be a number 2^n , where n is a positive integer greater than 0. The minimum modulation order is $M = 2$ as the data that needs to be transmitted is binary and cannot be represented by a smaller modulation order. The number of bits that can be assigned to each symbol can be determined by the modulation order as $\log_2 M$.

PSK is a modulation technique that changes the phase of the carrier wave to represent information [21, pp. 426–436]. The carrier signal can be expressed as in Equation 2.2, where A is the amplitude (typically 1), f_c is the carrier frequency, and ϕ_i is the phase shift corresponding to the i -th symbol.

$$s(t) = A \cos(2\pi f_c t + \phi_i) \quad (2.2)$$

For example, binary phase-shift keying (BPSK) has two distinct symbols, which corresponds to having a phase shift of 180° or π radians between the symbols. The higher the modulation order for PSK, the smaller the phase shift is between the symbols, which reduces the modulation technique's robustness to noise and other negative channel effects, but allows it to have a higher data rate.

Similar to PSK, ASK changes the amplitude A of the carrier wave in discrete steps to represent information whilst keeping the frequency and phase constant [21, pp. 420–422]. Likewise, FSK changes the frequency f of the carrier wave with a constant amplitude and phase [21, pp. 521–527]. Each of these modulation techniques can be used with a generic modulation order that is given, for example, as M -PSK (in the case of PSK) where M is the modulation order described above.

QAM is a modulation technique that differs from FSK, ASK, and PSK as it changes more than one carrier wave property in the modulation process. Specifically, the signal amplitude and phase are varied at the same time to produce a QAM modulated signal [21, pp. 439–449]. Essentially, QAM combines aspects of ASK and PSK and is widely used today. Symbols are mapped onto a constellation of points that are defined in a two-dimensional in-phase (I) and quadrature (Q) plane. The I component represents the part of the signal that is in-phase with the carrier wave and the Q component represents the part of the signal that is 90° out

of phase of the carrier wave. Typical QAM signals are 16-QAM, 64-QAM, 256-QAM and 1024-QAM.

Transmitted QAM waveforms have the following form as in Equation 2.3, where f_c is the carrier frequency, $I(t)$ is the amplitude value of the in-phase component, and $Q(t)$ is the amplitude value of the quadrature components. The carrier frequency f_c is held constant in the case of QAM.

$$s(t) = I(t) \cos(2\pi f_c t) - Q(t) \sin(2\pi f_c t) \quad (2.3)$$

Generally, PSK and QAM modulated signals can be described as a complex signal with the following form using I and Q components:

$$s(t) = I(t) + jQ(t) \quad (2.4)$$

Constellation diagrams are used to visualise these modulation techniques in the form of a two-dimensional plot of the I and Q components. Note that these diagrams are used primarily in the case of QAM and PSK due to their nature of having the I and Q components as the axes. By examining the constellation diagram, the modulation order and type can be identified. An example of the modulation types 8-PSK and 16-QAM is provided in Figure 2.2, where the different symbols of the modulation type can be seen as well as the bit mapping assigned to each symbol. The constellation diagram displays a normalised modulated signal with unit average power. Normalisation ensures consistent signal power across modulation schemes, improving comparisons and preventing dominance by higher-power signals. Furthermore, it can be seen how the PSK modulation type only alters the phase of the signal while QAM modulates both the phase and amplitude. The distance between symbols indicates the modulation type's resistance to noise, fading, and channel effects. Greater spacing enhances robustness.

An alternative way to represent modulated signals is through their amplitude (A) and phase (P) in an A/P format. This representation is given in Equation 2.5, where $A(t)$ is the instantaneous amplitude and $\phi(t)$ is the instantaneous phase [21, p. 45]. Note that in this study, the I/Q format is used rather than the A/P format.

$$s(t) = A(t)e^{j\phi(t)} \quad (2.5)$$

The A/P representation is equivalent to the I/Q representation as follows [21, p. 45]:

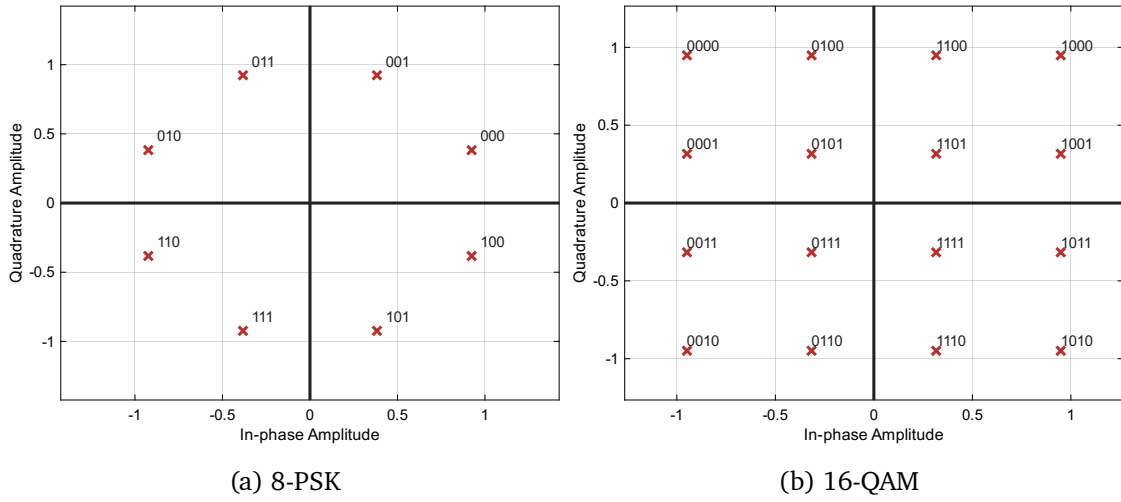


Figure 2.2: Constellation diagrams of PSK and QAM modulation techniques showing symbol positions.

$$A(t) = |s(t)| = \sqrt{I^2(t) + Q^2(t)} \quad (2.6)$$

$$\phi(t) = \arg\{s(t)\} = \arctan\left(\frac{Q(t)}{I(t)}\right) \quad (2.7)$$

2.2.2 Signal channels

When transmitting a wireless signal, it is subjected to various sources of interference and noise, which can affect the reliability and data transmission rate of the wireless communication system. Signal channels are the mediums through which the wireless signal travels and influence the characteristics of the signal.

The different classes of channel models consist of small-scale fading, large-scale fading, slow or fast fading, and frequency-based fading [21, pp. 791–815]. Small-scale fading results from multipath propagation, in which signals arrive at the receiver through multiple paths with different delays and phase offsets. The different signal components are added constructively or destructively, causing fluctuations in the signal amplitude. Large-scale fading comprises path loss, in which the signal power deteriorates as it travels from the transmitter to the receiver, and shadowing, in which obstacles or obstructions lie in the signal's path. Slow fading occurs when the channel changes slowly relative to the signal duration, whereas fast fading occurs when the channel changes quickly. Frequency-based fading is either flat fading, in which the entire signal bandwidth is affected, or frequency-selective fading, in which only a portion of the bandwidth is affected.

Frequently used signal channels are an AWGN channel, a Rayleigh fading channel, and a

Rician fading channel [21, p. 367, 803, 809]. Rayleigh and Rician fading channels are examples of small-scale fading, with Rayleigh having no line-of-sight component and Rician having a line-of-sight component between the transmitter and receiver. The AWGN channel is commonly used as a baseline to compare other channels against.

The AWGN channel is the simplest mathematical model that can be used to obtain the performance metrics of an AMC technique. The channel represents random noise, which generally originates from the thermal noise inherent in all electronic devices, has a constant power spectral density, and follows a normal distribution (also known as the Gaussian distribution). AWGN is represented as the vector n in Equation (2.1).

A received signal makes use of the superposition principle, where the received signal consists of the modulated original signal and unwanted noise. The metric signal-to-noise ratio (SNR) can be used to describe the level of the desired signal to the level of noise that was added when transmitted. It is typically measured in decibels (dB). Equation 2.8 is used to calculate the SNR, where P_{signal} is the power of the signal and P_{noise} is the power of the noise.

$$\text{SNR} = \frac{P_{\text{signal}}}{P_{\text{noise}}} \quad (2.8)$$

To translate the SNR value into a dB value, the following equation can be used:

$$\text{SNR}_{\text{dB}} = 10 \log_{10}(\text{SNR}) \quad (2.9)$$

As an example, Figure 2.3 shows the modulation types 8-PSK and 16-QAM in an AWGN channel with a SNR of 20 dB. With a lower dB value, the constellation points are more spread out from the intended symbol point and vice versa. Furthermore, the constellation diagrams also provides an intuitive explanation of why greater spacing between the symbols translates into better error rates. As the symbols move farther from their expected locations, they drift into the other symbols' expected positions and fall within their decision boundaries, causing errors when demodulating the signal.

Phase offsets can occur where the received signal is shifted in phase relative to the transmitted signal. Oscillator mismatches, synchronisation errors, or propagation effects can cause these phase offsets. In Equation (2.1), the phase offset that is applied to the received signal is represented by ϕ . In an AMC context, phase offsets can complicate the classification process because the classifier has to accommodate for these offsets. This is notably detrimental for phase-sensitive modulations, such as PSK signals, since excessive phase offsets can lead to confusion between received signal symbols and adjacent ones on the constellation diagram.

Two commonly used fading channels are Rician fading and Rayleigh fading. Rayleigh fading

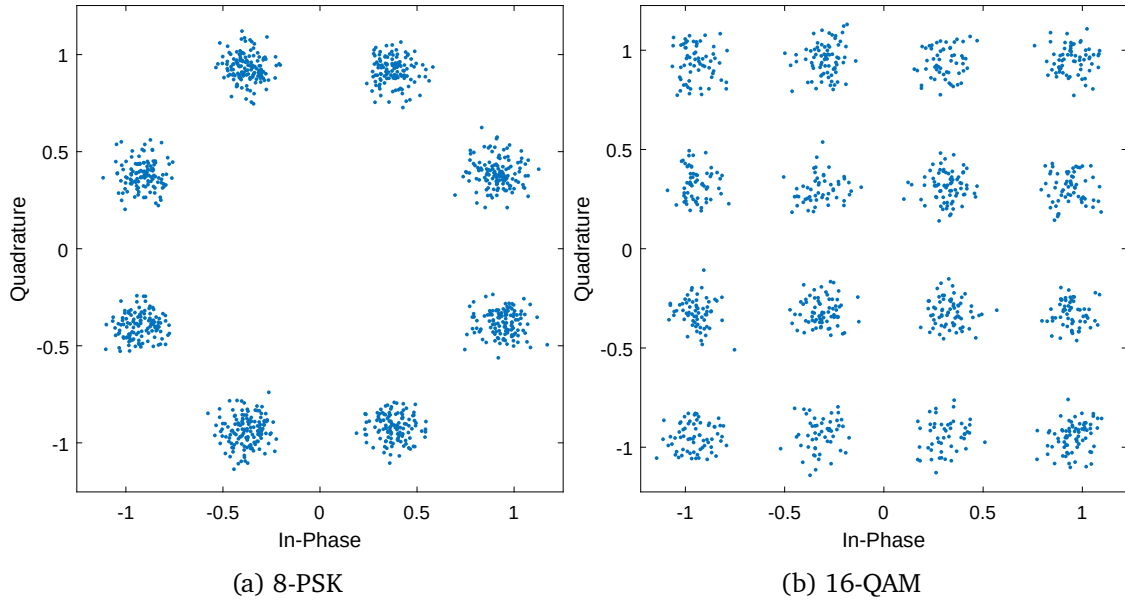


Figure 2.3: Constellation diagrams of PSK and QAM modulation techniques with AWGN at a SNR of 20 dB.

occurs when there is no dominant line-of-sight component between the transmitter and the receiver, where the received signal is the sum of numerous multipath components with random amplitudes and uniformly distributed phases [21, p. 803]. In contrast, Rician fading occurs when a dominant line-of-sight component is present described using the Rician K factor. The K factor is defined as the ratio of power in the line-of-sight component to the power of the scattered components. If the K factor is 0, then Rician fading becomes Rayleigh fading. Compared to an AWGN channel, Rician or Rayleigh fading can introduce significant complexity to the process of classifying modulation types.

2.3 Non-deep learning methods

AMC has been traditionally achieved using classical methods that do not make use of deep learning techniques. These techniques can be mainly categorised as likelihood-based techniques and feature-based techniques [5, pp. 35–46, 65–79]. Feature-based AMC techniques use the inherent characteristics of the different modulation types by extracting distinctive features that identify the specific modulation type. These discriminative features are typically paired with machine learning algorithms such as SVMs, KNNs, Naive Bayes, or decision trees. In contrast, likelihood-based techniques rely on hypothesis testing to determine the modulation type, with each hypothesis representing a different modulation type, but these techniques require almost perfect channel knowledge. Feature-based techniques typically have lower computational complexity than likelihood-based techniques at the cost of generally giving sub-optimal solutions compared to likelihood-based techniques [5, pp. 137–138].

Furthermore, clustering has also been used in AMC through a process of using specific clustering algorithms with a decision-making process to determine the modulation type.

2.3.1 Likelihood-based methods

The likelihood-based modulation classification process consists of multiple steps that can be treated as a hypothesis test. First, if unknown, the channel parameters such as amplitude, phase offset, and noise are estimated. These parameters are then used in a likelihood function for each possible modulation type to determine the likelihood of that modulation type being the one utilised [5, pp. 36–44]. The modulation type that returns the highest likelihood is selected as the identified modulation type. The choice of signal model and computational complexity determines the likelihood function and channel estimation applied.

Figure 2.4 shows the general procedure for a likelihood-based classifier, where the number of possible modulation types are n . Note that pre-processing of the received signal samples can be first applied.

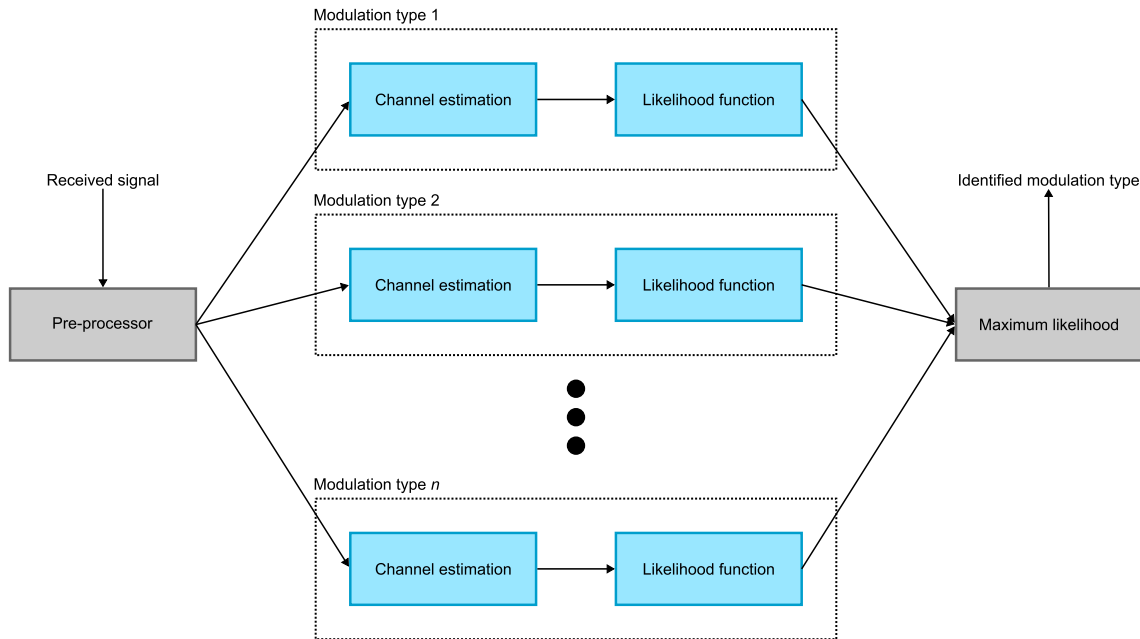


Figure 2.4: Process of likelihood-based classification.

Source: Adapted from [5, p. 40]

The maximum likelihood (ML) is the most general case of likelihood-based classification. The ML classifier follows the procedure mentioned above, with the exception that all channel parameters are already known, that is, it requires perfect channel knowledge [5, pp. 36–44]. Different likelihood functions can be used by the ML classifier for AWGN channels, non-Gaussian noise channels, and fading channels [5, pp. 36–44].

An example of the likelihood-based classification process is provided. Consider an AWGN channel, then the likelihood $\mathcal{L}$ that a received signal sample $r[k]$ belongs to a modulation type $\mathcal{M}^{(i)}$ can be computed as follows in Equation 2.10 [5, p. 36]. Here, the likelihood is equal to the probability $\mathcal{P}$ that the signal sample can be observed in the AWGN channel with that modulation type. The only channel parameter in this scenario is the noise standard deviation represented with σ .

$$\mathcal{L}(r[k] | \mathcal{M}^{(i)}, \sigma) = \mathcal{P}(r[k] | \mathcal{M}^{(i)}, \sigma) \quad (2.10)$$

Using the complex form probability density function for an AWGN channel, the likelihood can be determined as follows where s_m is the expected modulation symbol and M is the modulation order [5, p. 36]:

$$\mathcal{L}(r[k] | \mathcal{M}^{(i)}, \sigma) = \sum_{m=1}^M \frac{1}{M} \frac{1}{2\pi\sigma^2} \exp\left(-\frac{|r[k] - s_m|^2}{2\sigma^2}\right) \quad (2.11)$$

Since the received signal sample can correspond to any of the expected symbols of the modulation type, the likelihood is averaged over all the received signal samples, where K is the number of received samples [5, p. 36]:

$$\mathcal{L}(r[k] | \mathcal{M}^{(i)}, \sigma) = \prod_{k=1}^K \sum_{m=1}^M \frac{1}{M} \frac{1}{2\pi\sigma^2} \exp\left(-\frac{|r[k] - s_m|^2}{2\sigma^2}\right) \quad (2.12)$$

Consider the pool $\mathcal{R}$ of possible modulation types, then the identified modulation type is selected as shown in Equation (2.13), where the hypothesis $H_{\mathcal{M}^{(i)}}$ of each modulation type is determined using the likelihood of that modulation type [5, p. 39]. The modulation type with the highest likelihood is selected as the modulation type of the received signal.

$$\hat{\mathcal{M}} = \arg \max_{\mathcal{M}^{(i)} \in \mathcal{R}} \mathcal{L}(\mathbf{r} | \mathcal{H}_{\mathcal{M}^{(i)}}) \quad (2.13)$$

The general ML classifier's alternatives, the average likelihood ratio test (ALRT), the generalised likelihood ratio test (GLRT), and the hybrid likelihood ratio test (HLRT), can be used in cases where any of the channel parameters are unknown [5, pp. 40–43], [22]. Furthermore, several proposed quasi variants of the ALRT, GLRT, and HLRT are available, aiming to reduce computational complexity while maintaining similar performance [22].

In the ALRT approach, unknown parameters are treated as random variables with specific probability density functions [22]. The parameters are replaced with the integral of all their possible values and probabilities in the probability density functions, giving the averages of the parameters [5, p. 40].

As the number of unknown parameters increases, the ALRT becomes more computationally complex, and the alternative generalised likelihood ratio test (GLRT) becomes more preferable [5, p. 41]. Instead of treating unknown parameters as random variables, the parameters are treated as unknown deterministics [22]. The likelihood function in the GLRT maximises the likelihood of the unknown parameters over a possible range, giving the most likely value for the parameter. This method has less computational complexity than the ALRT, but is often biased toward higher-order modulation types [5, p. 42].

The hybrid likelihood ratio test (HLRT) was created to accommodate the bias in the GLRT for higher-order modulation types. It is a combination of the ALRT and the GLRT, where the likelihood is determined by averaging the signal samples and then maximising the resulting likelihood function [5, p. 43], [22]. Typically for the HLRT the channel parameters are estimated and the received data symbols are averaged.

2.3.2 Feature-based methods

Feature-based modulation classifiers use the characteristics of the signal to classify the modulation type of a signal by selecting specific features of the signal and using a decision-making process to determine the modulation type [5, p. 65], [22]. The features commonly used by feature-based classifiers are signal spectral-based features, wavelets-based features, high-order statistics-based features, and cyclostationary analysis-based features [5, p. 65]. All of these features make use of the amplitude, phase, and frequency of the unknown signal to determine the modulation type used. These features are typically paired with a machine learning algorithm to determine the modulation type, where the classification process can employ multiple features to improve the classification probability [23].

Signal spectral-based features comprise numerous parameters calculated from the signal samples. As mentioned in [5, pp. 66–69], some of the signal spectral-based features that are used are listed below. After these features have been calculated, a classifier such as a decision tree can be used to determine the specific modulation type used by the unknown signal.

- An evaluation of the spectrum symmetry around the carrier frequency.
- The standard deviation of the absolute value of the normalised and centred instantaneous frequency.

- The maximum value of the spectral power density of the normalised and centred instantaneous amplitude.
- The standard deviation of the absolute value of the normalised and centred instantaneous amplitude.
- The standard deviation of the normalised and centred instantaneous amplitude.
- The kurtosis of the normalised and centred instantaneous amplitude.
- The standard deviation of the absolute value of the non-linear component of the instantaneous phase.
- The standard deviation of the non-linear component of the direct instantaneous phase.

Wavelet transform-based features are based on the wavelet transform, which allows changes in the instantaneous amplitude, phase, and frequency of the signal to be localised [22]. Specifically, the continuous wavelet transform (CWT) can be used to determine the characteristics of the signal that differentiate different modulation types [5, pp. 71–74]. The modulation types FSK and PSK can be reliably classified using features obtained through the CWT, but ASK and QAM can not be differentiated using these features [5, pp. 71–74]. A disadvantage of the CWT is that it can be difficult to implement practically due to its computational complexity and the requirement to select the appropriate wavelet. An available alternative is the Stockwell transform (S-transform), which generally performs better than the CWT and is easier to practically implement [23], [24]. Furthermore, the S-transform is capable of successfully distinguishing between the modulation types ASK, FSK, PSK and QAM [23].

High-order statistics-based features consist of moment-based features and cumulant-based features. Moments are statistical measures that describe the specific shape of the probability distribution, where the first moment gives the mean, the second moment gives the variance, the third moment gives the skewness, and the fourth moment gives the kurtosis [25, pp. 164–166]. These moments can be obtained through a moment generating function. When taking the natural logarithm of the moment generating function, the cumulant generating function is obtained [25, p. 173]. The modulation types pulse-amplitude modulation (PAM), PSK, and QAM can be differentiated through the use of cumulants of the second order and fourth order, as each of these modulation types will have different values for these statistical measures [5, pp. 75–76].

Cyclostationary analysis-based features are another set of features that can be used in AMC. The periodicity of the unknown signal is used to model the signal as a cyclostationary signal, allowing for a spectral correlation function (SCF) and cyclic domain profile (CDP) to be determined [5, pp. 76–78], [23]. Both the SCF and CDP can be used to identify the modulation type, but generally struggle with higher-order modulation types, where

cyclic cumulants can be used [5, pp. 76–78]. These features are highly robust against noise, which is useful in an AWGN signal channel and other channels, but they have relatively high computational complexity [23].

Typical machine learning algorithms that are used for AMC are SVMs, KNNs, decision trees, and Naive Bayes [5, pp. 81–87], [26], [27]. SVMs are used due to their ability to find optimal separating hyperplanes using the features of the different modulation types [28, pp. 496–505]. They are effective in high-dimensional feature spaces and can handle non-linear decision boundaries through the use of kernel functions. KNNs classify the modulation type by a majority vote of the training samples nearest to a given input [28, pp. 16–18]. These KNNs compute the distances between samples through feature vectors that describe each sample. Decision trees split the feature space into regions by learning threshold-based rules and are represented as a tree where each decision leads down a branch of the tree [26]. One advantage of decision trees is that they are more interpretable than other machine learning methods, but they can suffer from overfitting. Naive Bayes classifiers assume conditional independence between features and apply Bayes’ theorem to compute posterior probabilities of each modulation type [27].

2.3.3 Clustering

Clustering is an unsupervised machine learning method that groups together data points that are similar and in close proximity. This is achieved through the use of a clustering algorithm such as k-means, fuzzy c-means (FCM), density-based spatial clustering of applications with noise (DBSCAN), Gaussian mixture models (GMMs), and other clustering algorithms [29]. Every clustering technique is based on a different criterion to identify the clusters, including hierarchy, partitioning, fuzzy theory, distribution, density, graph theory, and other criteria [29]. These clustering algorithms are unknown-order or known-order algorithms, where the order refers to the number of clusters. For example, k-means is an unknown-order clustering algorithm, thus a different algorithm or human intervention is needed to determine the ideal number of clusters. Each of these clustering algorithms utilize a distance metric to evaluate the proximity and similarity of data points, such as the standardised Euclidean distance, Minkowski distance, Cosine distance, and other metrics [29].

Clustering has yielded good outcomes in AMC, although it requires an adequate SNR [30]–[32]. In this context, the order of the clustering algorithm refers to the number of symbols in the modulation type. For example, 16-QAM would have sixteen distinct clusters, a cluster per symbol position. Many of the proposed AMC methods that involve clustering have a decision-making component that identifies the exact modulation type based on the determined number and positioning of the clusters, similar to a decision tree. Clustering can also be seen as a feature-based method since it operates on the inherent characteristics of the signal modulation type, but it is not typically grouped with the other feature-based

methods.

2.4 Deep learning methods

DNNs are a subset of machine learning methods that consist of multiple layers of interconnected nodes, where each layer extracts and transforms features from the input data to capture patterns such as signal modulation types. DNNs rely on DL to learn the relationships between model inputs, such as the I/Q stream, and model outputs, such as 16-QAM or other modulation types. DNNs use a loss function, which serves as the objective criterion that quantifies the difference between the predicted outputs and the truth labels during training. When training DNNs, the objective is to minimise this loss function. Large datasets are required for DNNs, which represent the distribution of all expected data occurrences, as DNNs must train on these datasets to learn the features.

DL methods are typically used for either classification or regression problems, where AMC is a classification problem. DNNs have been applied in AMC with significant success compared to features-based and likelihood-based NDL methods [6], [7]. The NDL methods previously described rely on hand-crafted features and processes to determine the modulation type of signals, while DNNs are trained systematically to recognise the modulation type of signals. Furthermore, DNNs can accurately classify the modulation type in more challenging channel conditions, including the effects of fading, pulse shaping, and clock offsets, where NDL methods typically struggle.

2.4.1 Datasets

Numerous datasets have been created for the training and testing of DNNs that aim to recreate real-world effects on the signal such as AWGN, sample rate offset, centre frequency offset, multipath effects, and fading [4]. The available datasets consist of RML2016.04c, RML2016.10a, RML2016.10b, RML2018.01a, HisarMod2019.1, and the newly developed RML22 [4], [33]. The datasets consist of multiple signal frames of different modulation types, analogue and digital, at varying SNRs with the added channel effects. The signal frames are given in the I/Q format. Most of the proposed DNNs for AMC are evaluated on at least one of these datasets, with the majority using the RML2016.10a dataset [4]. Although a dataset that represents a simple AWGN channel can be used for training and evaluating DNNs, it is trivial to the complex patterns in the other channel effects that DNNs are capable of learning.

Table 2.1 gives an overview of the available datasets with the modulation types available in each dataset and their frame length of I/Q symbols. Frame length refers to the number of discrete complex-valued samples in a single segment of the signal to be classified. Most

Table 2.1: Datasets for training and evaluating DNNs for AMC

Dataset	Number of classes	Modulation types	Frame length
RML2016.04c [34]	11	BPSK, QPSK, 8PSK, 16QAM, 64QAM, CPFSK, GFSK, PAM4, AM-DSB, AM-SSB, WBFM	128
RML2016.10a [34]	11	BPSK, QPSK, 8PSK, 16QAM, 64QAM, CPFSK, GFSK, PAM4, AM-DSB, AM-SSB, WBFM	128
RML2016.10b [34]	10	BPSK, QPSK, 8PSK, 16QAM, 64QAM, CPFSK, GFSK, PAM4, AM-DSB, WBFM	128
RML2018.01a [35]	24	BPSK, QPSK, 8PSK, 16PSK, 32PSK, 16QAM, 32QAM, 64QAM, 128QAM, 256QAM, OOK, 4ASK, 8ASK, 16APSK, 32APSK, 64APSK, 128APSK, AM-SSB-WC, AM-SSB-SC, AM-DSB-WC, AM-DSB-SC, FM, GMASK, OQPSK	1024
RML22 [33]	10	BPSK, QPSK, 8PSK, 16QAM, 64QAM, PAM4, WBFM, CPFSK, AM-DSB, GFSK	128
HisarMod2019.1 [36]	26	BPSK, QPSK, 8PSK, 16PSK, 32PSK, 64PSK, 4QAM, 8QAM, 16QAM, 32QAM, 64QAM, 128QAM, 256QAM, AM-DSB, AM-SC, AM-USB, AM-LSB, FM, PM, 2FSK, 4FSK, 8FSK, 16FSK, 4PAM, 8PAM, 16PAM	1024

of these datasets range from -20 to 20 dB SNR in steps of 2 dB, except for RML2018.01a, which goes up to 30 dB SNR.

2.4.2 Types of deep neural networks

Common DNN architectures that are used in AMC are multi-layer perceptrons (MLPs), CNNs, RNNs, and hybrid models that combine the architectures mentioned [4], [37]. Transformers have also been successfully evaluated for AMC with comparable performance to the other DNN architectures [38], [39]. These DL based approaches are discussed in more detail in the following subsections.

Multi-layer perceptrons

Multi-layer perceptrons (MLPs) are one of the most straightforward DL models that are used in AMC. These neural networks consist of an input layer, multiple hidden layers, and an output layer, where the information can only be passed forward through the layers. The feedforward nature of the network is characterised by unidirectional connections between the nodes of the various layers. Hidden layers in the MLP processes inputs from the previous layers through weighted connections and activation functions [40, pp. 41–52]. Activation

functions give MLPs the non-linearity required to learn the patterns in the training data provided to the network. Common activation functions used in MLPs are the logistic sigmoid function and the tangent hyperbolic function [37]. As the number of hidden layers increases, the network becomes deeper and is able to represent more complex relationships between its inputs and outputs [40, p. 51].

In Figure 2.5, a diagram of the typical architecture for a MLP is provided. A MLP can have many hidden layers, providing it with the ability to represent the relationships between the input and output data.

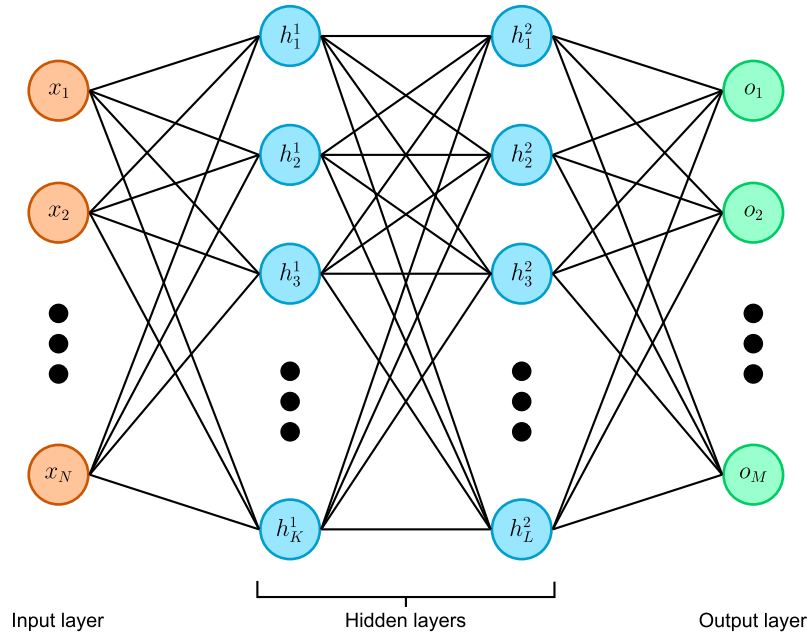


Figure 2.5: Generic MLP architecture.

Source: Adapted from [37]

MLPs have been applied in AMC typically using handcrafted features of the signal as input to the model such as high-order cumulants, kurtosis, skewness, and other statistical features [4], [37]. By using these features, the MLP relies on the mathematical features that are used in the traditional feature-based classifier to differentiate between modulation types. Fewer MLPs are proposed for AMC than CNNs, RNNs, and hybrid models [4], possibly because those models use fewer parameters or offer better classification performance in this domain.

Convolutional neural networks

Convolutional neural networks (CNNs) are commonly used in object detection, object classification, and image segmentation tasks where multiple input channels are used, but they have also been applied in AMC with significant success [15], [16], [41]. The structure of these networks consists of an input layer, multiple convolutional layers with activation

layers, multiple pooling layers, and one or more fully connected layers (also known as dense layers). The typical activation functions used in CNNs are rectified linear unit (ReLU), leaky ReLU, and exponential linear unit [37]. The fully connected layers function similarly to the hidden layers found in a MLP. A convolutional layer operates by sliding small filters (kernels) across the input data to compute weighted sums at each position, producing feature maps that capture local patterns [40, pp. 161–174]. Convolutional layers are well-suited for learning spatial features about the input data. The pooling layers are used to upsample or downsample the input to the layer and are typically placed between the convolutional layers.

A generic diagram of a CNN architecture is provided in Figure 2.6. In this figure, the different layers are shown such as the convolutional layers, activation layers, pooling layers, and fully connected layers. Furthermore, it is shown that before the fully connected layers can be used, the data must be changed from a multi-channel representation to a one-dimensional representation.

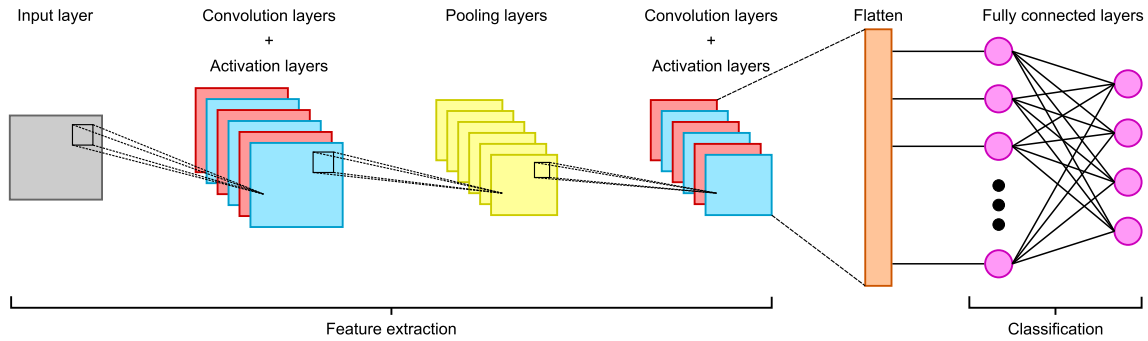


Figure 2.6: Generic CNN architecture for a classifier showing convolutional layers, pooling layers and fully connected layers.

CNNs are frequently applied in AMC with numerous architectures available due to their ability to extract the crucial characteristics that identify the modulation type of the signal [4], [37]. Most of the suggested CNNs are sequential signal-based models that use raw signal samples, specifically I/Q points, as input to the model [4], [37]. Another widely used form of input involves pre-processing the raw signal samples to generate constellation diagram images as input for the model, leveraging the typical role of a CNN in image classification. Furthermore, some approaches for CNNs in AMC make use of classical signal processing methods such as high-order cumulants, constellation density matrices, and other characteristics of the signal [4]. Compared to MLPs used in AMC, CNNs use significantly fewer parameters to define the model, thereby reducing the computational complexity of both training and model usage.

Recurrent neural networks

Recurrent neural networks (RNNs) are primarily used for sequence data because they function similarly to MLPs, with the added ability of leveraging previous inputs to influence the model output. This functionality is achieved through what are known as recurrent edges that can feed information back into previous layers or the same layer [37]. A RNN receives as input a sequence of data that is passed through the model, one sample at a time, allowing the model to compute the current input and a hidden representation of the previous input (through the recurrent edge) that allows its final output to be representative of the whole input sequence [40, p. 233]. The structures long short-term memory networks (LSTMs) and gated recurrent units (GRUs) can be used to help RNNs not forget information that is further back in the sequence [40, p. 233].

For example, a generic diagram of an RNN architecture is provided in Figure 2.7, illustrating how the model's outputs are influenced by the previous sequence of inputs. In this figure, o represents the output of the model, x is the input to the model, and h is a hidden node. The weights of each connection are also shown in the diagram and it is evident that a RNN is able to benefit from parameter sharing as it uses the same parameters for each time step [37]. Note that the diagram illustrates a generic scenario with an input sequence and an output sequence, which is not applicable to classification tasks, such as in AMC. Fully-connected layers are typically used after recurrent layers for classification.

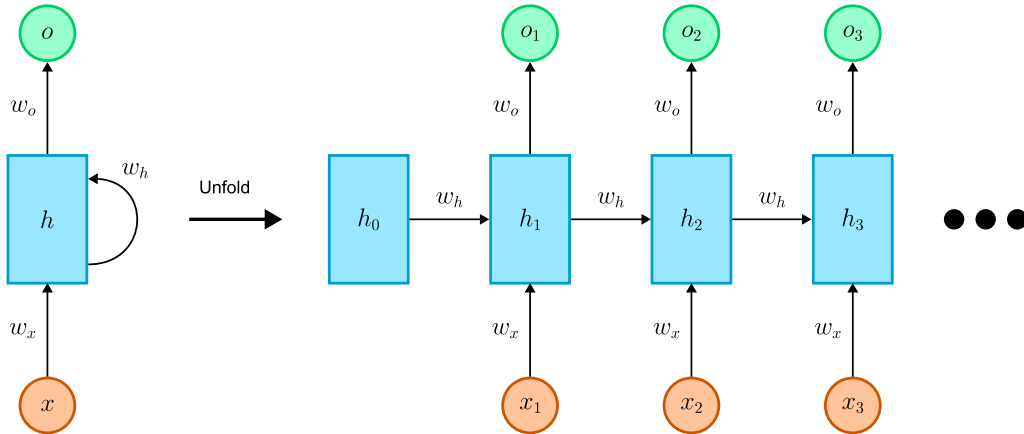


Figure 2.7: Generic RNN architecture showing the folded and unfolded version of the architecture.

Source: Adapted from [37]

RNNs are frequently utilised in AMC due to their ability to extract the temporal characteristics of the unknown signal, which helps the models to identify the modulation type. LSTMs and GRUs are also frequently used by RNNs in AMC to improve their long-term information retention with significant success [4], [37]. These RNNs either use the raw I/Q samples or amplitude and phase information as input to the model [4]. The capability of RNNs to extract temporal features enables them to generally outperform a MLP in AMC [37].

Hybrid neural networks

Some DNNs in AMC combine the previously mentioned architectures RNNs, CNNs and MLPs to create a hybrid model that aims to benefit from the advantages of the different architectures [4], [37]. This combination uses the spatial feature extraction ability of a CNN architecture and the temporal feature extraction ability of a RNN architecture. Typically, hybrid models begin with several convolutional layers followed by one or more LSTM layers, although other variations of this design are also used [4]. Some networks can have bypass connections, where the input skips one or more convolutional layers and is concatenated with the outputs of the convolutional layers before it is passed into a LSTM layer [42]. Other advanced connections and layers are also used in hybrid models to improve the performance and time complexity of the models.

Transformers

Transformers are another type of DL architecture, which was originally developed for natural language processing, but they have also been applied to AMC [38], [39]. Positional encoding and attention, specifically in the form of self-attention, enable transformers to be used effectively for different tasks [40, pp. 207–232], [43]. The transformer utilises positional encoding to learn the importance of element positioning in the sequence by adding a matrix to the network’s input. The matrix contains absolute or relative positional information about each element. Attention is the core mechanism of the transformer, which enables it to dynamically assign importance to different parts of the input sequence when generating an output. Specifically, self-attention involves comparing each part within the input sequence to assess its significance relative to the others [40, pp. 207–232], [43]. Multiple self-attention mechanisms can be used simultaneously through what is known as multi-head self-attention, where each head focuses on different aspects of the input sequence. Transformers improve upon RNNs by retaining long-term information better, as RNNs can forget information from the beginning of the sequence as the length of the sequence increases [44].

2.4.3 Notable neural networks

Significant developments have been made in DNNs that are specifically designed for use in AMC. Certain DNNs are highly influential because they are often used as a benchmark to compare the performance of new DNNs proposed for AMC. These DNNs were evaluated using one of the datasets mentioned in Section 2.4.1. An overview of these DNNs are provided in the following sections.

Convolutional neural networks

Neural networks that only use CNN techniques are among the noteworthy DNNs developed for AMC:

- CNN by O'Shea *et al.* [6],
- MCNet by Huynh-The *et al.* [37],
- Improved CNN-based AMC network (IC-AMCNet) by Hermawan *et al.* [16]

In 2016, O'Shea *et al.* demonstrated the use of CNNs on raw I/Q signal samples for AMC for the first time [6]. The architecture of their CNN follows a fairly standard design, as it uses two convolutional layers with ReLU activation functions, followed by a fully connected layer and an output layer. Two variations of this CNN architecture were used, with one a smaller version with smaller filters, and the other a larger version with larger filters. In this paper, they also developed and released the RML2016.10a dataset, which was later frequently used to benchmark DNNs for AMC. The key result of this paper is the finding that CNNs can process raw I/Q samples directly, learning the required feature representations without any guidance, surpassing traditional approaches that rely on hand-crafted features derived from I/Q samples.

Huynh-The *et al.* developed an efficient and high-performing CNN known as MCNet using the RML2018.01a dataset [15]. The architecture comprises several convolutional blocks with bypass connections, each consisting of distinct convolutional layers. Bypass connections help prevent overfitting and vanishing gradients when training the CNN. Convolutional kernels are arranged in parallel to enable the network to effectively capture multi-scale spatiotemporal signal features while keeping the number of model parameters low. This network utilises six core convolutional blocks, which the authors refer to as M-blocks, consisting of three convolutional layers with varying kernel sizes. Some M-blocks also include max pooling layers. MCNet achieves high classification accuracy while maintaining relatively low computational complexity, while outperforming rival CNNs.

Similarly, Hermawan *et al.* created IC-AMCNet to achieve low latency (below 0.01 ms) for beyond 5G communications, which was evaluated on the RML2016.10b dataset. [16]. The IC-AMCNet architecture is a more straightforward CNN compared to MCNet, where it simply uses multiple convolutional, pooling, and dropout layers to extract the features of the signal. The dropout layers help prevent overfitting by randomly deactivating a fraction of neurons during training. After these layers, the matrix is flattened and fully connected layers are used to classify the modulation type of the signal. One feature that sets IC-AMCNet apart is its use of a Gaussian noise layer to help prevent overfitting. The IC-AMCNet is evaluated against other CNNs for AMC, where it obtains a relatively low computing time compared to the other CNNs in the paper and outperforms the other CNNs.

Hybrid networks combining CNN and RNN architecture

Several significant hybrid DNNs that combine CNN and RNN techniques have also been developed for AMC. The hybrid models discussed are as follows:

- CNN-LSTM model by Zhang *et al.* [45],
- Multitask learning-based deep neural network (MLDNN) by Chang *et al.* [46],
- Multi-channel convolutional long short-term deep neural network (MCLDNN) by Xu *et al.* [47],
- PET-CGDNN by Zhang *et al.* [48]

The hybrid CNN-LSTM model was developed by Zhang *et al.* to focus on the feature interaction of the signal through a dual-stream structure [45]. This was achieved by including a preprocessing step in which the input signal for the network was converted into I/Q components and into the A/P representation, one format for each stream. Each stream uses three convolutional layers and two LSTM layers, with both streams being combined afterwards forming an input for a fully-connected layer which predicts the modulation type. The RML2016.10a dataset was used for experimentation with the network to obtain performance results. Zhang *et al.* reasons that combining the CNN and LSTM techniques allows the model to interact better with the spatial-temporal properties of the signal [45].

Similar work was done by Chang *et al.* in the creation of the MLDNN model [46]. As with the CNN-LSTM model, the MLDNN model also makes use of the I/Q components and A/P representation, but it differs with specifically created blocks that aim to increase the performance of the model. Chang *et al.* argue that in modulation classification, the I/Q components perform better in low SNRs while the A/P format is better in high SNRs [46]. The MLDNN architecture consists of a multitask learning head which is composed of different parts, specifically a part for high SNR predictions using the A/P representation, a part for low SNR predictions using I/Q components, a part for predicting the SNR, and part to merge the predictions. These different parts consist of multiple convolutional and RNN layers, which feed into each other to form the final modulation prediction. The MLDNN model was evaluated using the RML2016.10a and RML2018.01a datasets.

The MCLDNN model developed by Xu *et al.* is another hybrid DNN that differs from these other models by relying only on I/Q components as input to the network [47]. Xu *et al.* argue that using multiple forms of a signal as input to the network is not practical, as it can be a complex task to obtain the different forms in practical applications and can affect the performance of the model [47]. The MCLDNN model accepts three different inputs, namely the full I/Q signal, the I channel, and the Q channel. Separating the I and Q components into different inputs for the network enables it to use the complementary information of

the different channels effectively. Multiple convolutional, LSTM, and fully-connected layers are used by the model. The RML2016.10a and RML2016.10b datasets are used to evaluate the model, revealing that the model achieves some of the best performance in AMC [4].

Zhang *et al.* created the PET-CGDNN model, which has a specific component that specialises in extracting phase offset related information and uses phase parameter transformation to improve model performance [48]. The model aims to have fewer parameters, shorter training times, and shorter testing times than other AMC models while having comparable performance. The network comprises three key parts. Initially, the phase parameter is estimated using a fully-connected layer followed by a linear activation function. Subsequently, the phase parametric inverse transformation is executed in the next part to remove the phase offset in the input signal. In the third and final part of the network, convolutional, GRU, and fully-connected layers are used for modulation classification. The PET-CGDNN model was evaluated on the RML2016.10a, RML2016.10b, and RML2018.01a datasets, achieving performance results comparable to those of other benchmark models.

Transformers

The transformer architecture has also been used for the AMC context. Noteworthy DNNs that make use of the transformer architecture are as follows:

- Multi-scale radio transformer (Ms-RaT) by Zheng *et al.* [38],
- Transformer network (TRN) by Cai *et al.* [39]

Zheng *et al.* created the Ms-RaT model targeting fine-grained modulation classification of challenging and similar modulation types, such as 16-QAM versus 64-QAM [38]. A dual-channel representation of the signals is used, where I/Q symbols are pre-processed to be in the form of amplitude and phase spectra, split into image blocks, and then fed into the multi-scale radio transformer. 2D spectrum patches, embedded with positional encodings, are fed into alternating layers of multi-head attention and multi-layer perceptron blocks. In addition, a multi-scale pyramid module is integrated within the multi-layer perceptron blocks of the transformer, enabling hierarchical feature extraction and tighter decision boundaries. The model was pre-trained using RML2018.01a and then fine-tuned on their own simulated dataset with modulation types of QAM, PSK, ASK, FSK and on-off keying (OOK). They compared the Ms-RaT model to a CNN, RNN using LSTM, and a graph convolutional network (GCN). Their results indicate that the transformer model outperforms these other models, but it has more parameters than the other models.

The TRN model was developed by Cai *et al.*, which was one of the first times that a transformer network was applied in the domain of AMC [39]. Their architecture processes raw

I/Q data by first segmenting and patching the I and Q components, followed by a linear projection layer that transforms the I/Q patches into linear sequences with embedded positional and sequence encoding. These are fed into a transformer encoder stack, which uses a multi-head attention and a multi-layer perceptron block to capture long-range temporal dependencies in the I/Q sequence. Finally, a multi-layer perceptron head performs the classification. For comparison, the TRN model was compared against a CNN-based model, a LSTM-based model, a sequential convolutional recurrent neural network (SCRNN), a spatial transformer network, a KNN, and a SVM. They used the RML2016.04c, RML2016.10a and RML2018.10a datasets to evaluate and compare their model against the other classification models. The TRN model consistently outperformed the other models, with the improvements in performance most significant at low SNRs. Furthermore, the TRN model has less parameters than the other DL models.

2.5 Adversarial attacks

Adversarial attacks exploit vulnerabilities in deep neural networks (DNNs) by introducing subtle perturbations to the input data to the models, causing the DNNs to potentially misclassify the data [8]. There are several different types of adversarial attacks, but evasion attacks are the most common and specifically involve modification of the inputs to the model during the testing and operation of the DNN.

These adversarial attacks can manifest as white-box, gray-box, or black-box attacks [8]. In white-box attacks, an attacker has full knowledge of the model and its inputs, whereas in black-box attacks, this knowledge is absent. In gray-box attacks, an attacker possesses limited knowledge of the model and its inputs. These gray-box attacks are created using a generative model that only has access to the DNN throughout its training phase to generate adversarial samples [49]. Furthermore, attacks can be targeted or untargeted, where targeted attacks pushes the DNN to misclassify the input as a specific class, and untargeted attacks pushes the DNN to just misclassify the input without aiming for a specific class [8].

Adversarial attacks have been evaluated on DNNs used in AMC, showing a considerable decrease in model classification performance due to the attacks [9]–[11]. The adversarial perturbations are crafted using an attack method such as the fast gradient sign method (FGSM) and superimposed on the modulated signal using a certain power budget [9]. These perturbations push the DNN to misclassify the modulated signal as the wrong modulation type.

The adversarial attack strength is commonly measured using the perturbation-to-noise ratio (PNR) metric which indicates the ratio between the power of the adversarial attack and the noise in the signal channel, where typically at low/negative PNRs the attack is less successful and at high/positive PNRs the attack is more successful [9], [50]. Furthermore,

the perturbation-to-signal ratio (PSR) describes the ratio of power between the perturbation and the signal. The relationship between PNR, SNR, and PSR is provided in Equation 2.14.

$$\text{PNR [dB]} = \text{PSR [dB]} + \text{SNR [dB]} \quad (2.14)$$

An adversarial attack is deemed imperceptible if the PNR is 0 dB or lower, as the noise is at an equivalent or higher power level, obscuring the attack.

It has been shown that knowledge of the model is crucial for carrying out effective adversarial attacks in white box scenarios, as such knowledge can significantly impact the success of the attack [9]. Nonetheless, black-box attacks are also capable of diminishing the classification accuracy of the models [50].

In Figure 2.8, a typical attack scenario in AMC is shown where an attacker superimposes adversarial perturbations on transmitted data [9]. In this scenario, there is a DNN that classifies the signal at both the adversarial transceiver and the legitimate receiver. The adversarial transceiver uses an attack method to generate adversarial perturbations that are transmitted to the legitimate receiver, which can cause the legitimate receiver to misclassify the modulation type. Notice that the benign data transmitted to the adversarial transceiver and the legitimate receiver are subject to different channel conditions and delays. Likewise, the adversarial perturbations transmitted by the adversarial transceiver can also be affected by different channel conditions. These signal channel differences can affect the effectiveness of the attacks generated by the adversarial transceiver. Despite the imperfections in the signal channel and the potential sources of error in synchronisation between the transmitter and the attacker, attacks have been shown to still reduce the classification accuracy of the legitimate receiver [9], [51]. In numerous papers, adversarial attacks are examined without considering the signal channels of both the attacker and the legitimate transmitter, assuming that the attack takes place at the legitimate receiver [13], [14], [50].

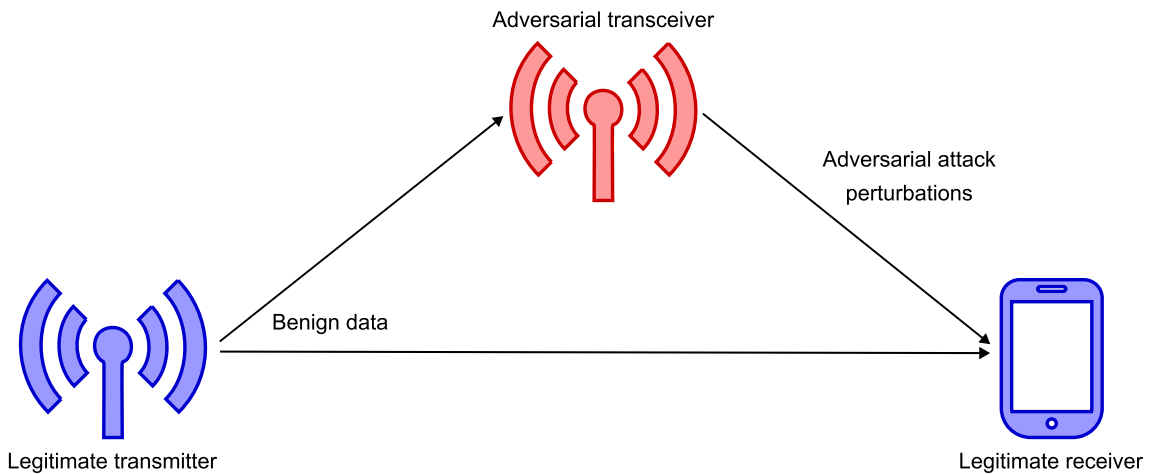


Figure 2.8: General scenario of adversarial attacks in AMC.

Several techniques have been employed to enhance the adversarial defence of DNNs in AMC, thereby making the DNNs more robust against adversarial attacks. In [13], three defence techniques were evaluated on a CNN using the RML2016.10a dataset, namely randomised smoothing, hybrid projected gradient descent adversarial training, and fast adversarial training. These techniques improved the DNN's robustness to adversarial attacks, particularly black-box attacks, which were rendered comparable to a traditional jamming attack. However, the defence techniques were less effective against white-box attacks and the effectiveness diminishes as the power of the perturbation increases. Despite the success of these techniques, the vulnerability to adversarial attacks still remains.

An adversarial attack is commonly constrained to be inside a certain power budget using a L_p norm, which is defined in Equation 2.15 for a complex signal $\mathbf{r}$. The variable p is the norm, where a value of 1 is the Manhattan distance, a value of 2 is the Euclidean distance, and ∞ is the Chebyshev or maximum distance.

$$\|\mathbf{r}\|_p = \left(\sum_{i=1}^K |r_i|^p \right)^{\frac{1}{p}} \quad (2.15)$$

Common adversarial attacks that are tested on DNNs in AMC are the fast gradient sign method (FGSM), Carlini & Wagner (C&W), projected gradient descent (PGD), universal adversarial perturbation (UAP), and DeepFool attacks [4], [9], [14], [50].

2.5.1 Fast gradient sign method white-box attack

The FGSM is a simple fast white-box attack method that operates by adding a perturbation to the input data by taking a step against the direction of the gradient of the DNN's loss function with respect to the input features, scaled with a defined factor [52]. By utilising the gradient, it can determine how to modify the input to increase the loss, thereby driving the model's prediction away from the true classification of the input data. Additionally, the use of the gradient exposes the most sensitive features in the input space that can decrease the classification accuracy of the model [52]. Furthermore, the FGSM method can also be seen as a form of gradient ascent. This approach uses the L_∞ norm, although there is a more general form known as the fast gradient method (FGM) that can also use the norms L_1 or L_2 to create attacks [53].

Figure 2.9 illustrates simply how this attack functions by moving against the gradient of the loss function of the DNN. A DNN's loss landscape is shown in this figure, which is a topographical depiction of the DNN's loss function. Note that usually the loss landscape is shown with respect to the weight space, but in this instance it is with respect to the input space. A clean sample refers to an input without any adversarial perturbation added to the

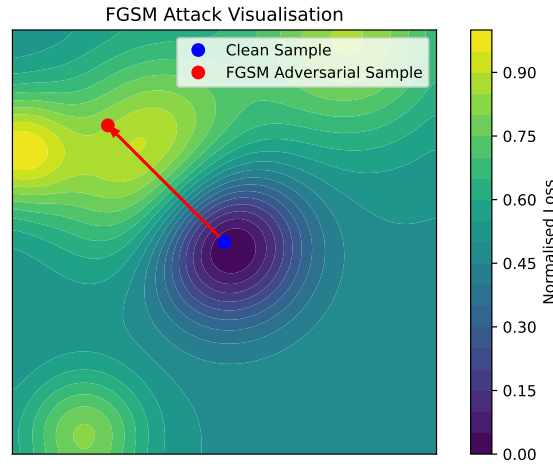


Figure 2.9: FGSM attack illustration on DNN gradients.

sample.

2.5.2 Carlini & Wagner white-box attack

The C&W adversarial attack is a white-box attack that treats the generation of adversarial attacks as an optimisation problem [54]. An objective function is used to find the minimal perturbation that causes a misclassification while keeping the perturbation as imperceptible as possible. The C&W approach iteratively improves adversarial attacks by repeatedly modifying the perturbations according to the objective function through an optimisation algorithm. Furthermore, it is more computationally expensive compared to the simple FGSM. This method can use L_0 , L_2 , or L_∞ norms to create the attack.

2.5.3 Projected gradient descent white-box attack

The PGD method is another iterative white-box attack that uses the loss function gradient to add perturbations to the input data, similar to the FGSM method [55]. Where the FGSM method takes a single step in the direction of the gradient with respect to the input features, the PGD method takes multiple steps in an iterative process using a specific step size. A constraint is used to limit the attack size using either L_1 , L_2 , or L_∞ norm. PGD is widely recognised in the optimisation literature, analogous to gradient descent but constrained, and in this adversarial attack context, it is utilised to create a perturbation.

Figure 2.10 illustrates the steps that the PGD attack takes to create an adversarial perturbation with a L_2 norm constraint. The landscape shown in this figure is with respect to the input space. A PGD attack may be initiated from the input sample, as depicted in the given diagram, or a random point within the perturbation constraint. The PGD method

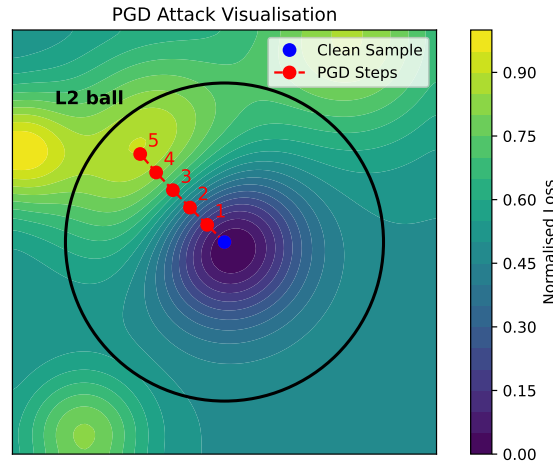


Figure 2.10: PGD attack illustration on DNN gradients with L_2 norm projection.

attempts to maximise the loss of the newly crafted sample within the epsilon-ball, where the epsilon ball is indicated by the black circle on the figure. We elaborate on the PGD method in Section 3.3.2.

2.5.4 Universal adversarial perturbation

The UAP technique is an adversarial attack that can function in both a white-box and a black-box context. It involves creating a single perturbation that can be applied to almost any input, causing the model to incorrectly classify the input [56]. The UAP technique is input-independent, whereas the attack methods discussed above are input-dependent. The attack is generated through an iterative optimisation procedure, where the adversarial perturbation is updated to maximise the loss over a batch of inputs. The minimal perturbation necessary to cause a misclassification is computed at each iterative step and added to the current universal perturbation, building the universal perturbation over the iterations [56].

2.5.5 DeepFool

The DeepFool attack is another white-box attack method that determines the minimal perturbation required for a DNN to misclassify an input [57]. Where the FGM and PGD methods operated on the gradients of the loss function of the model, the DeepFool method operates on the decision boundaries of the model. For a multi-class scenario, it locally linearises the decision boundaries close to the input sample, determines the closest linearised class boundary, and projects the input sample across the linearised boundary. Since the decision boundaries are linearised, it needs to iteratively check whether the input sample has crossed the actual class boundary. Following this process, it creates the minimal perturbation for the specific input sample.

Figure 2.11 provides an example of the DeepFool attack in a multi-class scenario. In the diagram, x_0 denotes the initial unperturbed input sample, with decision boundaries labelled $\mathfrak{F}_1$, $\mathfrak{F}_2$, and $\mathfrak{F}_3$. The initial iteration is provided in this diagram as the projection of the input sample across the linearised decision boundaries would not cause it to be misclassified by the neural network. More information is provided on the DeepFool method in Section 3.3.3.

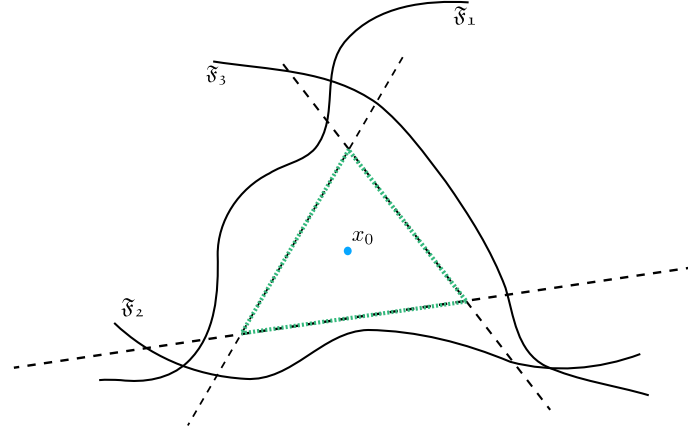


Figure 2.11: DeepFool attack illustrating locally linearised decision boundaries.

Source: Adapted from [57]

2.6 Chapter summary

This chapter discussed the relevant background information for the AMC domain. Basic telecommunications concepts such as the wireless communication system, modulation types, and signal channels were introduced. A simplified wireless communication system consisting of a transmitter and receiver was described, with the process of signal modulation at the transmitter side, transmission over a signal channel, and demodulation at the receiver side. Basic signal modulation types were discussed, such as PSK that changes the phase of a carrier wave to represent information and QAM that changes the amplitude and phase of a carrier wave to represent information.

The NDL approaches to AMC of feature-based methods, likelihood-based methods, clustering were examined. Feature-based AMC methods use the inherent characteristics of signals to determine modulation type through the use of features such as signal spectral-based features, wavelets-based features, higher-order statistics-based feature, and cyclostationary analysis-based features. Likelihood-based methods conduct a statistical hypothesis test to classify the modulation type. Clustering methods consist of using a clustering algorithm to group the data points for a specific modulation order and determining if the clusters are a good fit. The positioning of the clusters and number of clusters is used by clustering methods to classify the modulation type.

Additionally, the DL discussion covered the different types of DNNs used in AMC, which is MLPs, CNNs, RNNs, hybrid neural networks, and transformers. The differences between the architectures of the DNNs were mentioned and their applications in AMC. The datasets used by these DNNs were also provided. Notable neural networks that are highly cited were discussed to show the related work in the field.

The concept of adversarial attacks in the form of white-box, gray-box, and black-box attacks was explored. Adversarial attacks such as the FGSM, the C&W, the PGD, the UAP, and DeepFool were mentioned.

Chapter 3

Dataset and adversarial attacks

3.1 Introduction

We introduce the simulated dataset and the adversarial attacks applied to it in this chapter. A thorough description of the dataset is given by discussing the signal channel, modulation schemes, and signal frames. Furthermore, the creation of training, validation and test set are discussed. We explain the selected adversarial attacks of the FGM, PGD, and DeepFool and discuss how they are applied to the dataset. Furthermore, we introduce the scaling of these attacks using PNR. Constellation diagrams are provided to visualise the signal frames under the different attacks. An analysis of the DeepFool attacks is conducted to investigate their adversarial perturbation strength.

3.2 Description of dataset

MATLAB R2023b [58] is used to create a synthetic dataset simulating an AWGN channel for training and testing the various chosen AMC classifiers. Only an AWGN channel is considered, as it is assumed that some form of channel equalisation has taken place at the receiver before modulation classification occurs. Furthermore, using only an AWGN channel focuses the comparison of the classifiers on their robustness to adversarial attacks rather than their ability to classify under significantly adverse channel conditions.

The modulation types of BPSK, QPSK, 8-PSK, 8-QAM, 16-QAM, 32-QAM, 64-QAM, and 128-QAM are selected for use in the dataset. These modulation schemes are selected to cover a wide range of modulation orders and to cover different processes of modulating a signal.

Signal frames are generated for each of the modulation types selected, where unit average

power is used for the QAM modulated signal frames. The SNR for the signal frames varies from -30 to 30 dB in increments of 2 dB, where for each modulation type there are 2000 frames at each SNR. Each signal frame has a length of 2048 I/Q symbols, which is a sufficiently large frame length for the investigated AMC methods. The total dataset size is 496,000 input frames, where each input frame has the shape of (2048×2) with a column per I/Q symbol. Due to the large number of data points, the I/Q symbols are stored with the single data format. The signal frame length is selected so that the AMC methods are limited by the classification capabilities instead of the frame length. A long frame length is crucial for clustering AMC methods, as insufficient frame lengths can significantly impair performance [30].

The frames are divided into training, validation, and test sets in a 6:2:2 ratio, ensuring an equal distribution among modulation types for class balance. A reduced training and test set is selected, comprising a third of each respective original set, for the computationally expensive AMC methods. The reduced test set is used to evaluate the clustering classifier in Chapter 5, whereas the reduced training set is used to train the SVM of the hybrid classifier proposed in Chapter 6.

3.3 Selection and generation of white-box adversarial attacks

Adversarial attacks are required to test the robustness of the different AMC methods when adversarial perturbations have been added to the inputs. We choose the FGM, PGD, and DeepFool white-box attacks to examine varied adversarial attack methods and their impact on the classifiers. The gradient of the DL model's loss function with respect to input features is used by the FGM and PGD methods, while the decision boundaries of the different classes are used by the DeepFool method. The adversarial robustness toolbox (ART) [59] is used to generate white-box attacks, where the authors have verified the attack methods included in their toolbox. We assume that the adversarial attacks occur at the receiver, which is thus that the adversarial perturbations do not experience any detrimental signal channel effects, as done in numerous papers in the research field [13], [14], [50]. When generating the attacks, the entire signal frame (the 2048×2 matrix) is attacked jointly instead of attacking each element individually. A brief overview with graphical illustrations of the attack methods is provided in Chapter 2.5, but a more mathematical discussion of the methods is given here.

Using these selected attack methods, fixed-size and varying-size attacks are generated on the validation and test sets. For training purposes, the training and validation sets are used without any adversarial perturbations present. The attacked validation set is used only in Chapter 6 to evaluate the SVM used in the hybrid classifier combining DL and NDL techniques. For testing the performance of the AMC methods, both the attacked and the

unperturbed test sets are used. The fixed-size adversarial attack occurs when a specific attack size is selected regardless of any properties of the input signal, whereas in the varying-size attacks, a specific PNR is selected and the adversarial perturbation is generated to achieve this PNR. Note that in the case of the reduced test set, the same adversarial perturbation that is applied to the signal in the full test set is also applied to the corresponding signal in the reduced set for both the fixed-size and varying-size attacks. A discussion of how adversarial perturbations are scaled to achieve a specific PNR is given later in Section 3.3.4.

For white-box attacks, the adversarial perturbations are generated with full knowledge of the DNN model. In our case, the PET-CGDNN model is selected as our DNN model with the reasoning provided in Section 4.2. A PET-CGDNN model trained on the unperturbed training set and validation set described in Section 3.2 is used to generate the attacks.

3.3.1 Fast gradient method

The FGM method uses the gradient of the loss function of the DNN with respect to the input features to generate an adversarial perturbation [53]. A step is taken against this gradient, increasing the model's loss function. The FGM is a variant of the FGSM method that allows the attack to use norms different from only L_∞ . Misclassification when adding the adversarial perturbation is not guaranteed since the attack method uses the gradient to generate the attack and does not know how certain the model is about the classification of the input.

This study uses the L_2 norm variant, defined by Equation 3.1, where $\mathbf{x}$ is the input, ϵ is a scaling factor, L denotes the loss function of the DNN, θ are the DNN parameters, and y is the true label [53]. As can be seen in the equation, a higher ϵ creates a stronger attack that has a greater chance of causing misclassification. Furthermore, an adversarial perturbation generated with a higher ϵ is more visible when displaying the signal in a constellation diagram.

$$\mathbf{x}_{adversarial} = \mathbf{x} + \epsilon \cdot \frac{\nabla_{\mathbf{x}} L(\theta, \mathbf{x}, y)}{\|\nabla_{\mathbf{x}} L(\theta, \mathbf{x}, y)\|_2} \quad (3.1)$$

For fixed-size attacks on the dataset, an ϵ of 1.5 is selected for the FGM attack. This value is selected for multiple reasons:

- Attacks generated using this ϵ value causes a difference in classification performance when evaluating the DNN, without making the DNN practically ineffective.
- An adversarial perturbation generated using a very high ϵ would be obvious when added to the signal. This ϵ value is not easily perceptible when showing the signal

frame in a constellation diagram.

- This ϵ value allows us to see the difference between attacks generated using FGM and attacks generated using PGD with the same ϵ value.

3.3.2 Projected gradient descent

The PGD method also uses the gradient of the loss function of the DNN with respect to the input features, but it takes multiple steps against the gradient, whereas the FGM method takes only one step [55]. The adversarial perturbation constructed by the PGD attack is constrained within a predefined norm ball. With each iteration, the attacked sample moves in the gradient direction that most rapidly increases the loss, then projects the sample back to the allowed set if the step would leave it.

Mathematically, PGD can be described as follows when using the L_2 norm:

$$\mathbf{x}_{i+1} = \text{Proj}_{\mathbf{x}+S_\epsilon} \left(\mathbf{x}_i + \alpha \cdot \frac{\nabla_{\mathbf{x}} L(\theta, \mathbf{x}, y)}{\|\nabla_{\mathbf{x}} L(\theta, \mathbf{x}, y)\|_2} \right) \quad (3.2)$$

The variable $\mathbf{x}_{i+1}$ is the updated sample with added adversarial perturbation, $\mathbf{x}_i$ is the current sample, possibly already perturbed. The projection operator $\text{Proj}_{\mathbf{x}+S_\epsilon}$ limits the attack's strength to be within the norm ball. Attack step size is defined with α . Furthermore, S_ϵ is the perturbation constraint set, that is the set of allowed adversarial changes of size at most ϵ . The remaining variables are the same as those in Equation 3.1. Visualise the projection operator as a L_2 norm ball, centred on $\mathbf{x}$ and with radius ϵ . Typically, the number of steps taken is predefined.

For both fixed-size and varying-size attacks, we selected the number of iterations that are used by the PGD method as 100 with no restarts. As with the FGM attack, an ϵ of 1.5 with an α of 0.1 is selected for the fixed-size attacks on the dataset. The α value is selected to be small enough to allow the PGD method to find the local maxima in the loss landscape with respect to the input space, without being too small to cause a large number of unnecessary iterations. The loss landscape is a topographical depiction of the DNN's loss function, as shown in Figure 2.10. Loss landscapes are typically given with respect to the weight space, but in Figure 2.10 it is with respect to the input space.

3.3.3 DeepFool

The DeepFool attack uses the decision boundaries of the DNN to generate an adversarial perturbation, which is a different avenue compared to FGM and PGD [57]. The method locally approximates a DNN's decision boundaries as linear near a given input. By iteratively

linearising the classifier around the current input and projecting the input onto the closest decision boundary, DeepFool can compute a perturbation that pushes the input across the boundary.

The following algorithm generates a L_2 norm DeepFool attack, based on the algorithm in [57].

Setup: Consider a differentiable classifier $f : \mathbb{R}^n \rightarrow \mathbb{R}^c$ with the output $f(\mathbf{x})$ where $f_y(\mathbf{x})$ refers to the y^{th} class. For the classifier, n is the input dimension and c is the number of classes. The predicted label given by the classifier is defined as follows:

$$\hat{y}(\mathbf{x}) = \arg \max_y f_y(\mathbf{x}) \quad (3.3)$$

The goal of the attack is to find a small perturbation δ such that $\mathbf{x} + \delta$ is assigned a different label. The inputs to the DeepFool algorithm are the original unperturbed input sample $\mathbf{x}_0$ and the classifier f . The output is the final perturbation $\hat{\delta}$.

1. **Initialisation:** Set $\mathbf{x}_0 \leftarrow \mathbf{x}$ and the cumulative adversarial perturbation $\hat{\delta} \leftarrow 0$. Let i be the index of iterations, starting at 0. Stop when the predicted label changes, that is $\hat{y}(\mathbf{x}_i) \neq \hat{y}(\mathbf{x}_0)$.
2. **Linearise decision boundaries:** For every class $y \neq \hat{y}(\mathbf{x}_0)$ form the first-order approximation of the decision boundary between class y and the original class $\hat{y}$ at the current point $\mathbf{x}_i$:

$$\mathbf{w}'_y = \nabla f_y(\mathbf{x}_i) - \nabla f_{\hat{y}(\mathbf{x}_0)}(\mathbf{x}_i), \quad f'_y = f_y(\mathbf{x}_i) - f_{\hat{y}(\mathbf{x}_0)}(\mathbf{x}_i) \quad (3.4)$$

These equations define hyperplanes approximating where $f_y = f_{\hat{y}}$. The variable $\mathbf{w}'_y$ is defined as the difference between the gradient of the classifier for a specific class y and the gradient of the function for the predicted class $\hat{y}$.

3. **Find closest linear boundary:** For each $y \neq \hat{y}(\mathbf{x}_0)$ determine the approximate distance from $\mathbf{x}_i$ to the closest hyperplane:

$$\frac{|f'_y|}{\|\mathbf{w}'_y\|_2} \quad (3.5)$$

Choose the index $\hat{l}$ giving the smallest value (closest hyperplane):

$$\hat{l} = \arg \min_{y \neq \hat{y}(\mathbf{x}_0)} \frac{|f'_y|}{\|\mathbf{w}'_y\|_2}. \quad (3.6)$$

4. **Compute minimal step to cross boundary:** The perturbation that projects $\mathbf{x}_i$ onto the selected hyperplane is:

$$\delta_i = \frac{|f'_i|}{\|\mathbf{w}'_i\|_2^2} \mathbf{w}'_i \quad (3.7)$$

Update the cumulative perturbation and the current point:

$$\hat{\delta} \leftarrow \hat{\delta} + \delta_i, \quad \mathbf{x}_{i+1} \leftarrow \mathbf{x}_i + \delta_i \quad (3.8)$$

5. **Repeat until misclassification:** Repeat the steps 2–4 until $\hat{y}(x_{i+1}) \neq \hat{y}(x_0)$. Multiple iterations are required, as projecting the sample onto the linearised decision boundary may not cause it to cross the actual decision boundary. Typically, DeepFool converges in a few iterations. Return the adversarial perturbation $\hat{\delta}$.

Due to the DeepFool method operating directly on the decision boundaries, a fixed or varying-size attack cannot be constructed. The size of the attack is dependent on the decision boundaries of the classifier, as shown in the algorithm.

3.3.4 Scaling adversarial attacks

The PNR metric is used to relate adversarial attacks to the modulation classification process. The FGM and PGD attacks use ϵ to scale the size of the generated attack, where the relationship between ϵ and PNR is as follows [60]:

$$\epsilon = \sqrt{\text{PNR} \cdot \|\mathbf{x}\|_2^2 / (\text{SNR} + 1)} \quad (3.9)$$

The variable $\mathbf{x}$ is the received complex signal of I/Q points. Note that this relationship is only valid if the FGM and PGD attacks are generated using the L_2 norm. The required perturbation generated using ϵ scales directly with the input $\mathbf{x}$. Relatively small changes in $\mathbf{x}$ will be reflected with relatively small changes in ϵ . Intuitively, a longer signal frame length requires a larger adversarial perturbation to keep the same PNR, where this perturbation is spread out over more signal samples, thus the overall attack strength remains the same.

Varying-size adversarial attacks are generated on the validation and test sets using FGM and PGD, with a PNR ranging from -40 to 0 dB in 2 dB increments for all SNR values. The PNR is limited to be below 0 dB as exceeding this value would cause the adversarial perturbation power to be greater than the noise power, and it would not be considered imperceptible [50]. For each PNR, the required ϵ for each signal frame is determined using Equation 3.9. For the PGD method, the step size is calculated to be a tenth of the required ϵ .

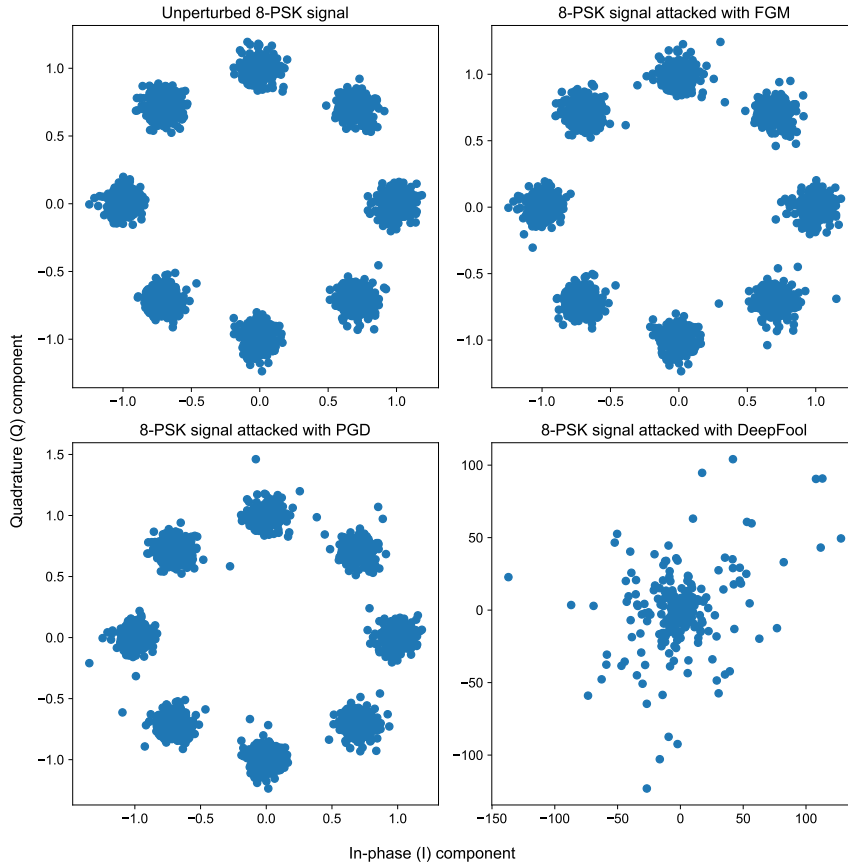


Figure 3.1: 8-PSK signal frames with a SNR of 20 dB with adversarial attacks.

3.4 Visualisation of dataset

The synthetic dataset is generated, and fixed-size and varying-size adversarial attacks are applied to the validation and test sets of the dataset. Constellation diagrams of the dataset are plotted to facilitate easier visual understanding of how the signals now appear. These diagrams can also provide an understanding of the features that are important for the trained PET-CGDNN model and are used to differentiate between modulation types.

Figure 3.1 shows an 8-PSK signal at a SNR of 20 dB when it is unperturbed and when it has been attacked through the FGM, PGD and DeepFool attack methods. Note that this figure is only for visualisation purposes and further details are provided later as it shows adversarial attacks on the PET-CGDNN model that is only trained in the following Chapter 4. In these constellation diagrams, the FGM and PGD attacks have a fixed size with a ϵ of 1.5. It is visible that with the unperturbed signal that the only adverse effect present is the AWGN. The FGM and PGD attacked signals appear similar to the unperturbed signal, showcasing that the adversarial attack is subtle. With the FGM and PGD attacked signals, some of the I/Q points are moved away from the centre of each symbol position, indicating that the trained DNN's loss function may operate on how close the I/Q points are to these

centres. Surprisingly, the perturbation applied by the DeepFool attack is not subtle at all, as the I/Q points are significantly shifted, and the original signal is no longer recognisable. Possibly the original signal is significantly altered by the DeepFool attack due to the decision boundaries of the PET-CGDNN. The DeepFool attack in this case cannot be considered to be imperceptible.

Constellation diagrams of a 16-QAM signal at a SNR of 20 dB are provided in Figure 3.2. A fixed-size FGM, fixed-size PGD, and DeepFool attack are applied to the signal. Note again that these diagrams show adversarial attacks applied to the PET-CGDNN model that will be presented later in Chapter 4, and are included here solely for visualisation purposes. The fixed-size attacks have an ϵ of 1.5. The same observations made with the 8-PSK signal are also relevant to the 16-QAM signal. The FGM and PGD attacked signal is visually similar to the original unperturbed signal, while the DeepFool attacked signal has been significantly altered.

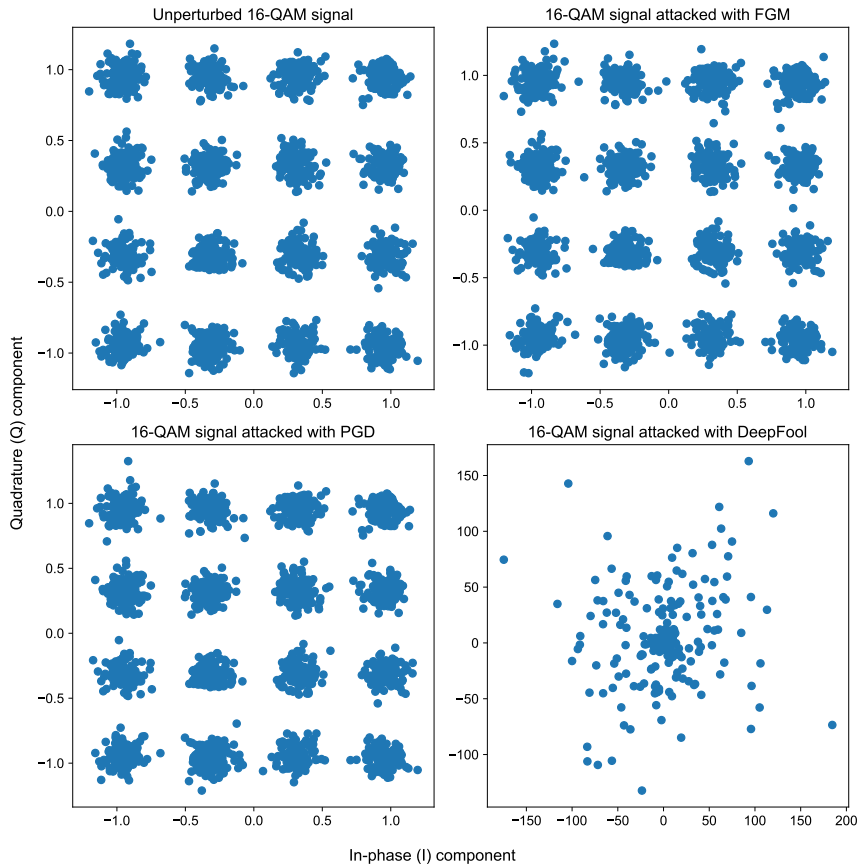


Figure 3.2: 16-QAM signal frames with a SNR of 20 dB with adversarial attacks.

3.5 Analysis of DeepFool attacks

The constellation diagrams highlight that the DeepFool attacks affect the original signals significantly more compared to the FGM and PGD attacks. It is necessary to investigate how the DeepFool attacks differ to the FGM and PGD attacks so that it can be considered when comparing and analysing the results of the different AMC methods on these attacks. The FGM and PGD fixed-size attacks were crafted using a specific ϵ , thus they can be considered to be bounded, while the DeepFool attacks iterate until a misclassification occurs. DeepFool attacks can therefore be considered to be unbounded.

To investigate the strength of adversarial perturbations crafted by the DeepFool method, the following Equation 3.10 can be used to determine the ϵ applied per signal sample. In this equation, $\mathbf{x}_{adv,i}$ is the adversarial signal and $\mathbf{x}_i$ is the original unperturbed signal. This equation uses the L_2 norm to be comparable with the ϵ used by the FGM and PGD attacks.

$$\epsilon_i = \|\mathbf{x}_{adv,i} - \mathbf{x}_i\|_2 \quad (3.10)$$

After the ϵ values have been calculated, a cumulative distribution function plot can be drawn as shown in Figure 3.3. This plot quantifies the proportion of DeepFool attacks that were crafted at or below any given perturbation magnitude on the simulated dataset. As shown by the distribution, the unbounded nature of DeepFool results in significantly large ϵ values being used. Specifically, only 23% of the DeepFool attacks are within the same budget ($\epsilon \leq 1.5$) as was given for the fixed-size attacks of FGM and PGD.

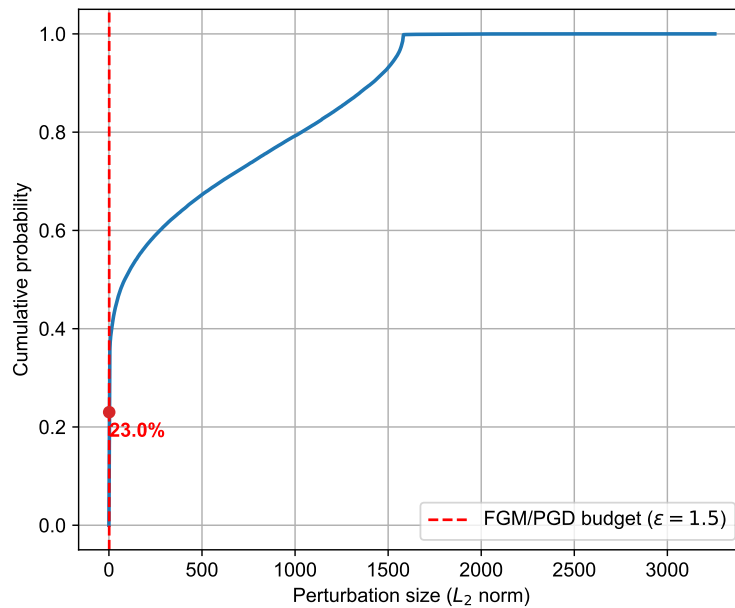


Figure 3.3: Cumulative distribution function of the ϵ of DeepFool attacks.

The DeepFool attacks that are within the same perturbation budget given for the fixed-size

FGM and PGD attacks are further analysed. This subset of the DeepFool attacks can be considered to be the bounded subset. The diagrams in Figure 3.4 highlight significant observations that can be made of this bounded subset. Figure 3.4a shows the distribution of modulation types in the bounded subset. It can be observed that all modulation types are approximately equally represented in the dataset. This is a surprising result as it could be expected that the higher-order modulation types, such as 128-QAM would require less of an ϵ to cause a misclassification compared to BPSK. Figure 3.4b can provide an explanation for why the modulation types are approximately equally represented. In this diagram, the proportion of the bounded subset per SNR is provided. It is apparent that all DeepFool attacks constrained to the same ϵ budget as the FGM and PGD attacks fall within the negative SNRs range. This is possibly due to the extreme noise that allows the DeepFool method to use only a small perturbation budget to create its attacks.

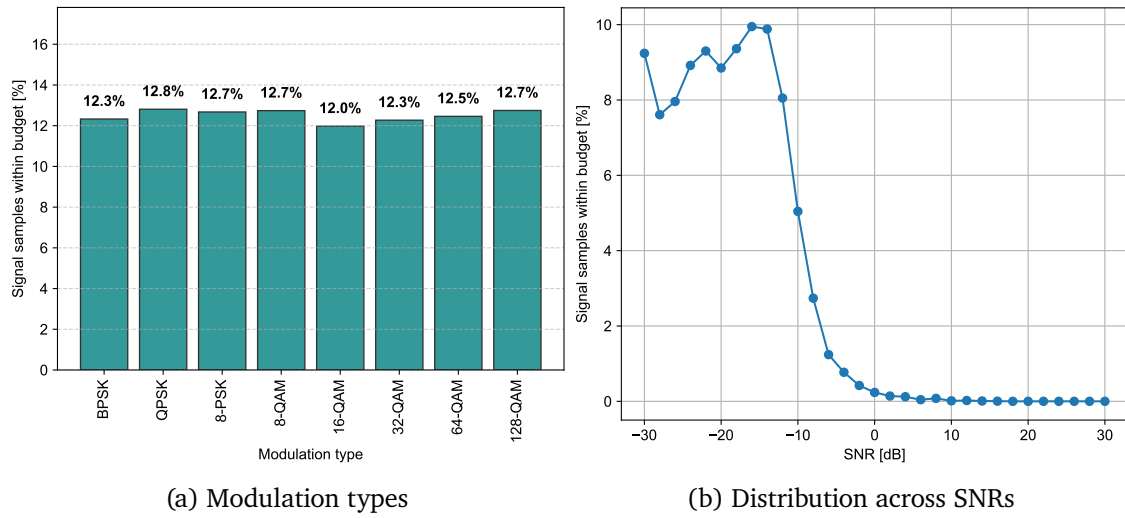


Figure 3.4: Bounded DeepFool attacks with $\epsilon \leq 1.5$.

This analysis highlights that the DeepFool attacks are the worst-case scenario for any AMC models due to their unbounded nature. The DeepFool attacks cannot be considered to be imperceptible due to these large ϵ values. Typically, DeepFool attacks result in small perturbations in the literature and thus this is a surprising result. The extremeness of the DeepFool attacks in our work must be kept in mind for all the next chapters where AMC methods are evaluated and compared.

3.6 Chapter summary

We discussed the generation of the simulated dataset and the reasoning behind using only AWGN as the signal channel. Furthermore, mathematical descriptions of the white-box adversarial attacks of FGM, PGD and DeepFool were provided. The generation of fixed-size and varying-size attacks was discussed. For the fixed-size attacks of FGM and PGD, a specific

ϵ was used to generate the attacks, while for the varying attacks the ϵ was scaled for a specific PNR. Additionally, the dataset was visualised by showing constellation diagrams of an 8-PSK and 16-QAM signal frame when they are unperturbed and when they are under attack using the different attack methods. It could be seen that the DeepFool attacks are significantly more adverse compared to the FGM and PGD attacks. An analysis of the DeepFool attacks was conducted and it was found that only 23% of the DeepFool attacks were within the same perturbation budget given to the fixed-size FGM and PGD attacks. It was concluded that the DeepFool attacks serve as the worst-case scenario for any AMC method.

This dataset is now available for use when investigating the different approaches to AMC. Overall 496,000 signal frames were generated with a SNR varying from -30 dB to 30 dB in increments of 2 dB with each signal frame having a shape of (2048×2) . The dataset contains an unperturbed training and validation set for training our classifiers for AMC. An attacked validation set is also created for Chapter 6, where we tune a hybrid classifier on the attacked validation set. A test set is also included in the dataset, with one configuration containing unperturbed signals and another containing attacked signals. Attacked signals are available as fixed-size attacks (where the ϵ is set) and varying-size attacks (where the PNR is set). For the varying-size attacks, the PNR ranged from -40 to 0 dB in increments of 2 dB. The test set is used to evaluate the classification performance of our classifiers.

Chapter 4

Investigation of DL techniques

4.1 Introduction

In this chapter, a DL approach to AMC, that is the PET-CGDNN, is selected and investigated. The reasoning for why the PET-CGDNN is selected is discussed and the architecture behind this model is thoroughly explained. We verify the implementation of the PET-CGDNN by following the same process that the original authors used and comparing the results we obtain with those reported in the paper. Furthermore, the training process on our simulated dataset, which includes hyperparameter tuning, is provided. The PET-CGDNN is evaluated on our dataset, both with and without adversarial attacks present, and the robustness of the method against these attacks is discussed.

4.2 PET-CGDNN architecture

The PET-CGDNN model [48] is selected as the DL approach to AMC due to its low model parameters, which makes it lightweight, and due to its relatively good classification performance [48]. It is a hybrid DNN that utilises CNN and GRU layers, while also incorporating unique predefined layers specific to the domain of modulation classification. The unique layers are specifically designed to correct any phase offset that the input signal could have been subjected to. These phase offsets are unintentional rotations of the complex-valued signal due to the signal channel, as described earlier in Section 2.2.2. The CNN layers give the model the ability to extract spatial features, whereas the GRU layers enable the model to extract temporal features for classification. Furthermore, the PET-CGDNN is selected because it forms a reasonable basis for a hybrid classifier that combines DL and NDL approaches to AMC due to its simplicity. Additionally, the PET-CGDNN has been cited numerous times and used as a comparison for other DNNs for AMC, indicating that it is

highly regarded. A summary of this DNN has been provided earlier in Section 2.4.3.

The PET-CGDNN architecture comprises three distinct stages: parameter estimation, parameter transformation, and a hybrid neural network [48]. Figure 4.1 shows the overall architecture of the PET-CGDNN.

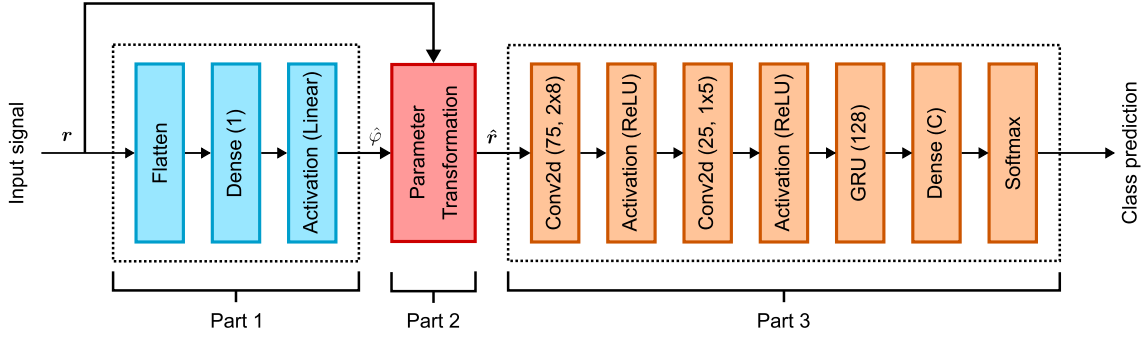


Figure 4.1: PET-CGDNN architecture.

Source: Adapted from [48]

The first stage of the model is the parameter estimator, designated as Part 1 in Figure 4.1 [48]. A received input signal r is arranged as a $2 \times L$ matrix containing I components in the first row and Q components in the second row, where L is the length of the signal frame. The input signal is often distorted by impairments such as phase offsets, frequency shifts, and AWGN, depending on the signal channel. The goal of this stage is to learn and estimate the phase offset parameter $\hat{\phi}$ that the signal could have been subjected to. This is achieved through a layer that flattens the input matrix, followed by a fully connected layer with a linear activation function. During the PET-CGDNN model training, the parameter estimator stage is also trained to predict a phase offset parameter, which describes the phase offset present in the received signal.

The second stage, Part 2, contains the parameter transformer [48]. This stage applies an inverse transformation to the input signal $\mathbf{r}$ using the estimated phase offset $\hat{\phi}$ of the first stage. Mathematically, the module rotates each I/Q point $r[k]$ in the input signal by multiplying it by $e^{-j\hat{\phi}}$, producing a corrected signal $\hat{r}[k]$. The inverse transformation is provided in matrix form in Equation 4.1, where $\Re$ designates the real I component and $\Im$ designates the imaginary Q component of the complex signal.

$$\hat{r}[k] = r[k]e^{-j\hat{\phi}} = \begin{bmatrix} \Re\{r[k]\} \cos \hat{\phi} + \Im\{r[k]\} \sin \hat{\phi} \\ \Im\{r[k]\} \cos \hat{\phi} - \Re\{r[k]\} \sin \hat{\phi} \end{bmatrix} \quad (4.1)$$

Part 3 contains the final stage of a hybrid neural network for feature extraction and classification [48]. The first two stages "cleaned" the input signal in terms of phase offset, allowing efficient use of CNN and GRU layers. The CNN component consists of two 2D convolutional

layers, where the first layer has 75 filters and a kernel size of (2×8) , and a second layer with 25 filters and a kernel size (1×5) . Both convolutional layers are followed by a ReLU activation function. After the CNN component, the signal is passed to the GRU component, which consists of a GRU layer with a size of 128. Final modulation classification is performed by a fully connected layer with hidden units C , where C is the number of modulation types in the dataset, and a final softmax layer.

Our dataset simulates an AWGN channel that does not include any phase offsets to which the signals could be subjected. Parts 1 and 2 of the PET-CGDNN architecture are thus not fully utilised. However, adversarial attacks applied to signals can potentially cause a phase offset in the signal. The unperturbed signals in the dataset are complex-valued signals in the I/Q format. Any adversarial perturbation is represented as a vector that is added to the original signal. The addition of the perturbation can change the angle of the symbol vectors, leading to a phase offset. The phase offset estimator and inverse transformer can potentially correct the phase offset applied by this adversarial perturbation.

4.3 Verification of implementation

We must verify our implementation of the PET-CGDNN model to ensure that it is implemented as intended by the original authors, Zhang *et al.* [48]. The verification process involves implementing the model, training it on the same dataset, and evaluating its classification performance. If the performance is the same as what Zhang *et al.* achieved and presented in their paper, then we can state that we have successfully verified our implementation and can use it for our investigations. Verifying the model lends credibility to our investigations.

The simulation code for the PET-CGDNN was made openly available on GitHub by the authors, who utilise the Keras and TensorFlow frameworks to develop the model. We use this code and make significant adaptations to suit our specific use case. The versions of Keras and TensorFlow that were used are outdated, thus we updated the frameworks to newer versions. Their Keras version was 2.2.4 and their TensorFlow version was 1.14.0, whereas our Keras and TensorFlow versions are 2.15.0. Therefore, it is necessary to verify that the method remains the same as the one the authors had developed.

In the paper, they used the RML2016.10a, RML2016.10b, and RML2018.01a datasets to train and evaluate the PET-CGDNN model. We select the RML2016.10a dataset for verification, which contains the modulation types of BPSK, QPSK, 8-PSK, 16-QAM, 64-QAM, CPFSK, GFSK, PAM4, AM-DSB, AM-SSB, and WBFM, with the SNR ranging from -20 to 20 dB in increments of 2 dB [6]. Each signal frame of each modulation type has a dimension of 2×128 , that is 128 I and Q components in separate rows. The RML2016.10a dataset simulates a harsh signal channel using the GNU Radio framework, which consists of AWGN,

time varying multipath fading, oscillator drift, and sample rate offsets [6]. The dataset is divided into training, validation, and test sets using the ratio of 6:2:2, as in Zhang *et al.*'s method.

An Adam optimiser with cross-entropy loss function is used to train the PET-CGDNN model on the training set and its performance is evaluated on the validation set. The learning rate (LR) is reduced by a factor of 0.5 if a plateau occurs in the reported validation loss after 5 epochs. Furthermore, early stopping is implemented, where training is halted if the model fails to improve its validation loss after 50 epochs. The maximum epoch is set very high at 10000, so that only early stopping is used to stop training. After training is stopped, the epoch with the lowest validation loss is saved as the model. The kernels of the convolutional layers are initialised using Glorot uniform, also known as Xavier uniform, to help prevent vanishing or exploding gradients during training. This is the same training configuration that the authors of the PET-CGDNN model used.

After the PET-CGDNN model is trained, it is evaluated on the test set from the RML2016.10a dataset. Figure 4.2 presents the classification performance on the test set, showing the classification accuracy at each SNR for both our implementation and the reported values by Zhang *et al.*. Observe that the SNR is capped at 18 dB instead of 20 dB, in line with Zhang *et al.*'s approach for comparable analysis. The classification performance is slightly lower than what they reported in their paper [48], especially at SNRs exceeding 0 dB, but the overall curve remains the same. The two curves differ by a mean squared error of 14.035, have a maximum difference of 7.318, and have a correlation coefficient of 0.999. Multiple initialisation seeds were used to train and evaluate the model PET-CGDNN without significant differences observed between the different seeds.

A difference in classification accuracies can be expected due to a difference in simulation hardware, versions of DL frameworks, and the seed used for training the PET-CGDNN.

From this process, it is concluded that the PET-CGDNN model has been successfully implemented as similar performance is achieved on the same dataset that the original authors had used. With verification complete of the model configuration, framework version update, data preparation, and training process, the PET-CGDNN is used in our studies.

4.4 Training process

The PET-CGDNN is trained and evaluated on the simulated dataset that was described in Chapter 3. Specifically, it is trained on the training set and its training is assessed using the validation set. Both the training set and the validation set contains no adversarial attacks. If the model performs well on the validation set, then it should generalise to the test set.

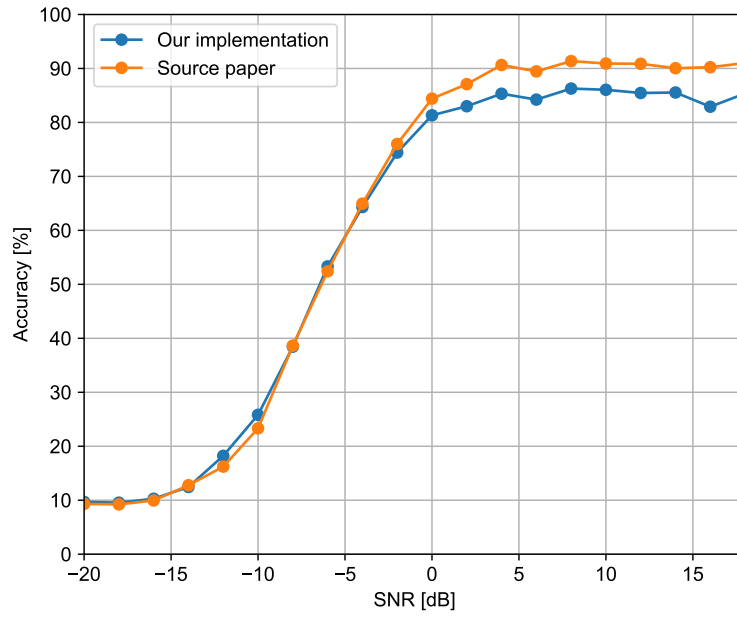


Figure 4.2: Classification performance of PET-CGDNN on RML2016.10a dataset for verification.

The DL frameworks Keras and TensorFlow are again used for our implementation of the PET-CGDNN. A similar training configuration used in the verification process is employed for training the PET-CGDNN on the simulated dataset. The only differences are that the maximum epoch is set to 500 and the patience for early stopping is changed to 30. The rest of the configuration remains the same.

Hyperparameter tuning is used with this training configuration to achieve a good-performing version of the PET-CGDNN.

4.4.1 Hyperparameter tuning

The performance of a DL model is greatly dependent on its hyperparameters, such as the LR, number of layers, or the regularisation strength. Hyperparameter tuning is required to find good values for these hyperparameters so that a good version of the DNN is used. Tuning is performed by defining a search space for these hyperparameters, selecting a search method such as grid search or random search, and evaluating the validation loss returned by the different combinations of hyperparameters. The combination that provides the lowest validation loss is selected as the best parameters for the model.

We use Optuna, a hyperparameter optimisation framework, to tune the PET-CGDNN model [61]. This framework uses the tree-structured Parzen estimator algorithm, which is a Bayesian optimisation algorithm, to find the best values for the hyperparameters [61]. Using this optimisation algorithm is theoretically more efficient than the grid search or

random search methods.

Hyperparameter tuning is performed for the initial LR, the batch size, the LR scheduler's patience, and the LR scheduler factor. We use the same training configuration as discussed above for the hyperparameter tuning, with the only difference being that we limited the number of epochs to 150 instead of 500 to reduce computational time. The results of the hyperparameter tuning after 40 trials have been completed are provided in Table 4.1, which also provides the search space boundaries that are used for the hyperparameters.

After every 10 trials, the tuning process is restarted so that a different seed is used, which makes the best parameters found more independent of the seed. The Optuna hyperparameter tuning process is stopped after every 10 trials, a different seed is selected for Tensorflow, and the study is resumed. This approach allows us to ensure the robustness of the selected hyperparameters. If a specific set of hyperparameters consistently yields high performance across different seeds, it provides confidence that the model is learning the necessary features to classify the signals, rather than simply performing well due to fortunate weight initialisations.

Table 4.1: Hyperparameter tuning results for PET-CGDNN.

Hyperparameter	Search space boundaries	Optimised value
Initial LR	$[1 \times 10^{-4}, 1 \times 10^{-1}]$	3.04×10^{-4}
Batch size	[32, 512]	438
Reduce LR patience	[3, 20]	8
Reduce LR factor	[0.01, 0.9]	0.8074

4.4.2 Training with selected parameters

With hyperparameter tuning complete, the PET-CGDNN is trained on the training set using the best parameters found. Figure 4.3 shows the accuracy and loss during training of the PET-CGDNN with these chosen parameters. As can be seen from the training curves, both training and validation loss decrease steadily while the training and validation accuracy increases, indicating that the model is learning useful representations. The losses and accuracies converge to relatively close values, showing good generalisation. The gap between the training and validation curves suggests that mild overfitting may be occurring, but it is not severe. Furthermore, training stops at 250 epochs when early stopping is triggered. Training any further after this point would not significantly reduce the validation loss, as the training has mostly converged.

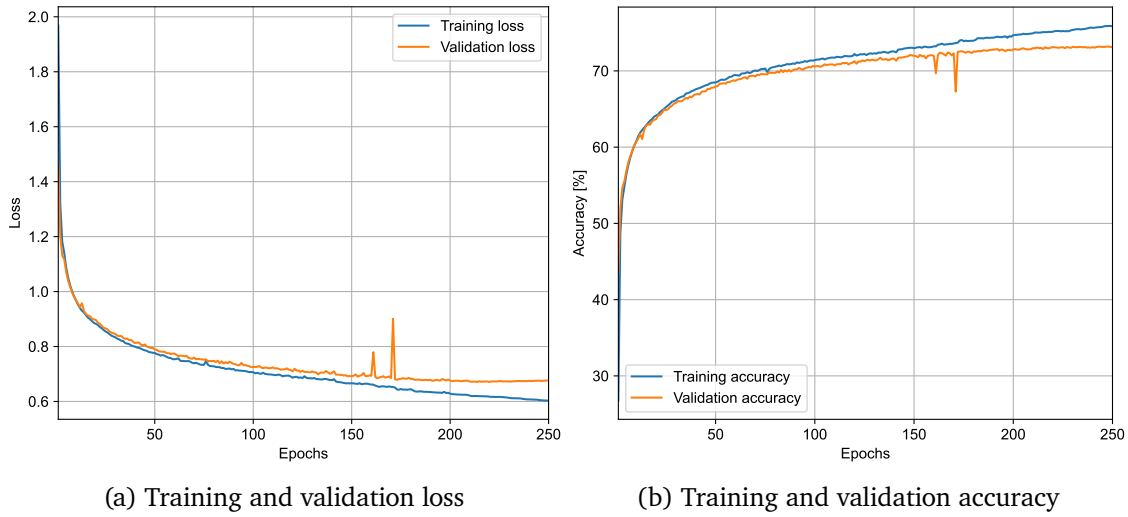


Figure 4.3: Training of PET-CGDNN using selected hyperparameters.

4.5 Results

The trained PET-CGDNN is evaluated on the test sets from the simulated dataset. First, unperturbed signals, that is signals without any attacks present, are used to evaluate the DNN. At each SNR, the percentage of accurately classified signal frames is determined as the classification accuracy. Afterwards, the PET-CGDNN is tested on fixed-size and varying-size attacks, where the classification accuracy is obtained at each SNR and PNR. Please note that despite being already mentioned and discussed in Chapter 3, that these attacks were crafted using the PET-CGDNN model and are fully white-box.

4.5.1 Performance on unperturbed signals

In Figure 4.4, the classification performance of the PET-CGDNN on unperturbed signals is shown. Figure 4.4a illustrates the overall accuracy when the modulation types are combined. Since only a AWGN channel is used in the dataset, the PET-CGDNN can easily learn the relationships between the input signals and the modulation schemes. Separating the classification accuracy per modulation type, as provided in Figure 4.4b, provides mostly the same information that is shown with the overall curve. However, it can be observed that at very low SNRs, the PET-CGDNN tends to classify unknown signals as 32-QAM, as evidenced by its relatively high classification accuracy.

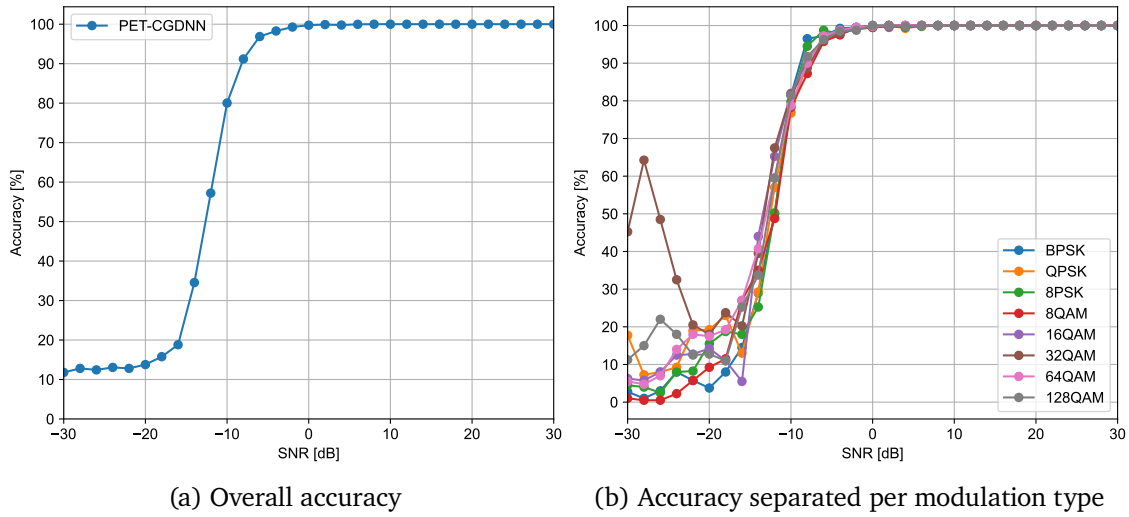


Figure 4.4: PET-CGDNN classification performance on unperturbed signals as SNR varies.

4.5.2 Performance on attacked signals

The PET-CGDNN is tested on the white-box adversarial attacks, which comprises of FGM, PGD, and DeepFool. The FGM and PGD attacks both have a fixed-size and varying-size configuration, as discussed in Chapter 3. Note that the DeepFool attacks are not categorised as fixed-size or varying-size, as discussed in Section 3.3.3, but are shown with the fixed-size attacks of FGM and PGD.

Fixed-size attacks

The classification accuracy of the PET-CGDNN on the DeepFool attacks and the fixed-size attacks of FGM and PGD is given in Figure 4.5. The performance of the PET-CGDNN on unperturbed signals is also included for easy comparison. Each adversarial attack significantly affects the accuracy of the PET-CGDNN, with the impact becoming more pronounced as the SNR increases. This is potentially due to the noise initially obscuring the adversarial perturbation, making it less effective. The FGM and PGD attacks are of equal size using the same ϵ , yet PGD perturbations have a greater impact than those from the FGM method. PGD’s multi-step nature in generating adversarial perturbations leads it to create stronger attacks. However, the most impactful perturbations come from the DeepFool method, which causes the PET-CGDNN to be unusable even from low SNRs. Again, this is due to DeepFool’s nature in generating perturbations, as it is designed to cause the samples to cross the decision boundaries around them.

Please note the analysis of the DeepFool attacks that was done in Section 3.5, when examining Figure 4.5. The DeepFool attacks can be considered to be the worst-case scenario for the PET-CGDNN due to its unbounded nature. It highlights the vulnerability that any model

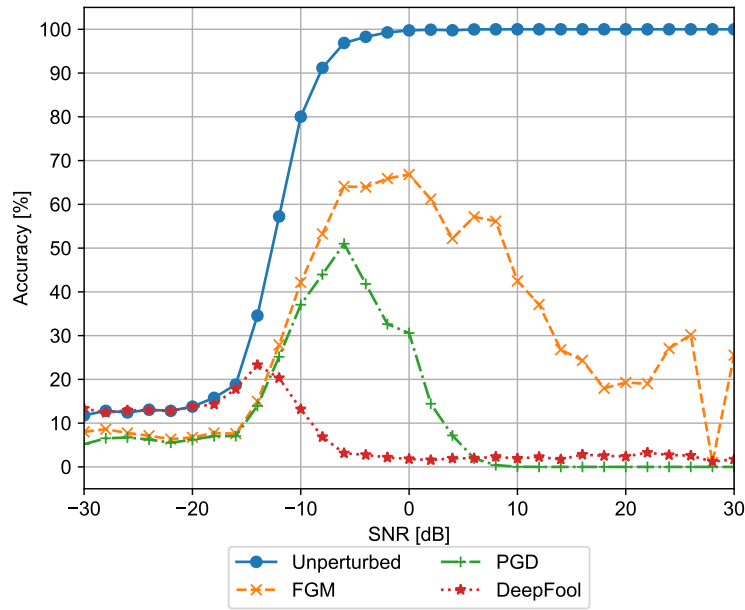


Figure 4.5: PET-CGDNN classification performance on fixed-size attacked signals as SNR varies.

has to white-box attacks that have full access to the model information.

Varying-size attacks

Evaluating the PET-CGDNN for varying-size attacks entails determining its classification accuracy at each SNR and PNR. Since there are now three dimensions, which are accuracy, PNR, and SNR, the results must be presented differently compared to the fixed-size attacks.

Figure 4.6 shows the classification accuracy of the PET-CGDNN as the PNR varies with specific SNRs, with Figure 4.6a showing the results on the FGM attacks and Figure 4.6b showing the results on the PGD attacks. For comparison, the unperturbed performance equivalent at the SNR of -10 dB is 80% and for the other SNRs it is 100%. From both diagrams, it can be seen that as the PNR increases the accuracy decreases as expected, since the perturbation is becoming greater compared to the noise. As seen with the fixed-size attacks, the perturbations generated by the PGD method have a greater impact on the performance of the PET-CGDNN, as indicated by the faster decrease in the curves. Furthermore, as the SNR increases, the point at which attacks start to have an effect occurs at higher PNRs. From previous discussions, the PNR does not exceed 0 dB as the attack cannot be considered imperceptible past that point. It can be deduced from these graphs that if the method has greater robustness against adversarial attacks, the curves would shift to the right or have a more gradual slope.

Since the previous diagrams only provided the performance of the PET-CGDNN at certain

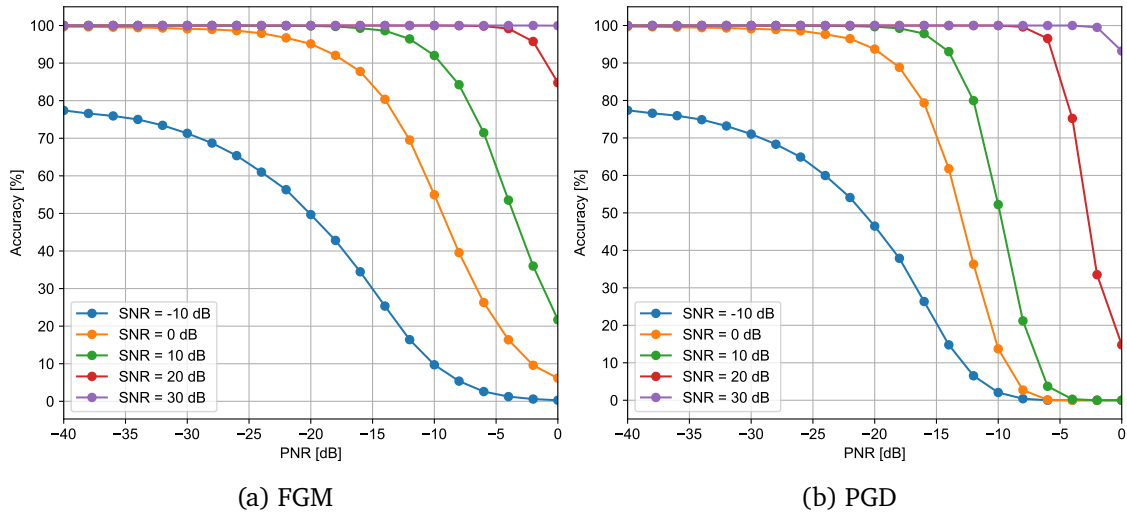


Figure 4.6: PET-CGDNN classification performance on signals attacked with varying-size attacks for specific SNRs as PNR varies.

SNRs, 3D surface plots of the classification accuracy are given in Figure 4.7 where all three dimensions are shown at once. Examining these surface plots yields the same conclusion that as the PNR increases, the attacks become more impactful by decreasing the classification accuracy. Furthermore, the PGD attacks are more effective compared to the FGM attacks.

4.6 Chapter summary

This chapter presented the PET-CGDNN as the DL approach to AMC. We discussed the architecture behind the PET-CGDNN by examining the different components of the architecture and their respective functionalities. Specifically, the first part of the PET-CGDNN is the phase offset parameter estimator, which estimates the phase offset applied to a signal. The second part of the PET-CGDNN contains the parameter transformer, which attempts to reverse the phase offset applied to a signal. Lastly, the third part of the PET-CGDNN is used to classify the modulation type of the signal after it has been corrected. The verification process used for the PET-CGDNN was also provided, demonstrating that the PET-CGDNN was correctly implemented, as shown by the similarity of the performance achieved to what the original authors had reported. Verifying the PET-CGDNN consisted of training the model on the RML2016.10a dataset and evaluating it on the test set of the dataset as the SNR varies. Furthermore, the training process of the PET-CGDNN on our dataset was described, and the hyperparameter tuning results were provided.

The resulting performance of the trained PET-CGDNN was evaluated on both unperturbed and perturbed signals. It was shown that the PET-CGDNN is not robust against the attacks, as they significantly decrease the classification performance of the DNN. The DeepFool attack

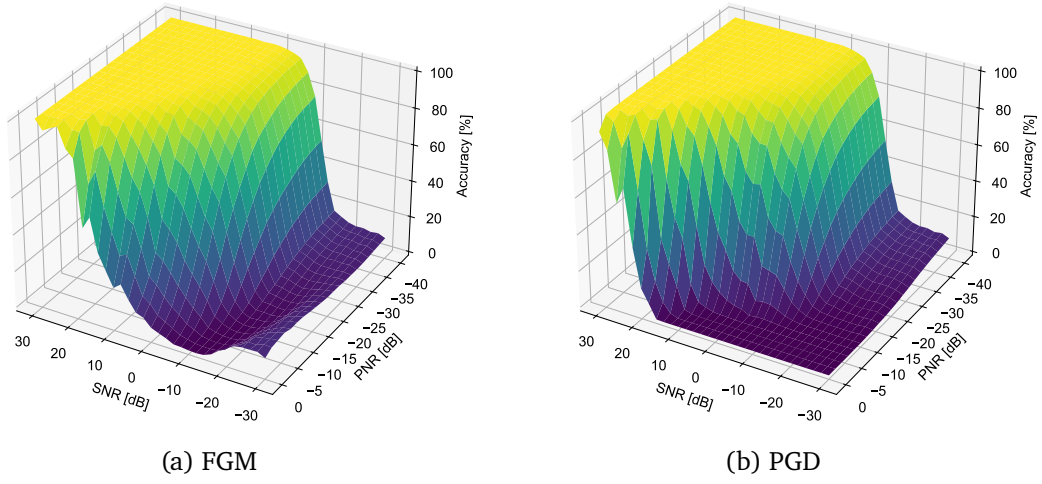


Figure 4.7: 3D surface of PET-CGDNN's classification performance on varying-size attacks.

was the most effective attack, followed by the PGD method, and then the FGM method. It was shown that, even when the attack is considered imperceptible, it has a significant impact, as found with varying-size attacks that scaled with PNR. With the varying-size attacks, it was found that the classification performance of the PET-CGDNN drops off more significantly at higher SNRs compared to lower SNRs.

Chapter 5

Investigation of NDL techniques

5.1 Introduction

This chapter covers the selected NDL approaches to AMC, that is a clustering classifier, the QHLRT, and a KNN that uses higher-order cumulants. The theory and classification process behind each NDL method is discussed in detail. We verify the implementation of the QHLRT and KNN by following the same or a similar approach used by the original authors and comparing our results with theirs. All NDL classifiers are evaluated on our dataset to investigate their robustness to adversarial attacks.

5.2 NDL methods

The selected NDL methods consist of a clustering classifier, the QHLRT, and a KNN that utilises higher-order cumulants. Each of these classifiers is chosen to cover different NDL approaches to AMC in order to investigate their robustness to adversarial attacks. We create the clustering classifier without basing it on any specific previous research in the field, whereas the QHLRT and KNN are based on work done by other authors.

In this section, the theory behind each NDL method and its classification process is discussed.

5.2.1 Clustering

A clustering classifier in AMC uses the grouping of the received I/Q symbols around pre-defined symbol positions to classify the modulation type. The clustering classifier process starts by using a clustering algorithm to group the received I/Q symbols. Clustering is performed for all possible modulation orders that a received signal could have, where the

modulation order is equivalent to the expected number of clusters. For example, 8-PSK would be expected to have 8 clusters. After clustering is performed for each possible number of clusters, the quality of the clustering fit is evaluated using a scoring metric. Using knowledge of the possible modulation schemes that the received signal can use, the classifier can derive the modulation type using the best-scoring number of clusters. A decision-making process is also included to differentiate between PSK and QAM when there is overlap in modulation orders.

We select the well-known unsupervised k-means clustering algorithm [62] for use in our clustering classifier, due to its low time complexity and high computing efficiency [29]. Computing efficiency is especially critical since clustering must be performed for each possible modulation order in the dataset, resulting in 7 clustering instances per received signal for our dataset. Clustering is performed for the number of clusters 2, 4, 8, 16, 32, 64, and 128.

The k-means algorithm is a known-order clustering algorithm based on partitioning; thus, the number of clusters k it uses for grouping is provided before clustering is performed [29]. It begins by randomly selecting k initial centroids, then iteratively assigns each data point to the nearest centroid [62]. Afterwards, it updates the centroids to be the mean of the points assigned to them. This process continues until the centroids converge or the maximum number of iterations is reached. The goal of the k-means algorithm is to produce compact, well-separated clusters by minimising the within-cluster sum of squared distances between each point and the cluster centroid [62].

The Davies-Bouldin index (DBI) [63] is selected for the scoring metric to evaluate the fit for each clustering instance. The general idea of the DBI is to measure, for each cluster, how similar it is to the most similar (i.e. closest) other cluster. Clusters that are compact and far apart from each other return a lower DBI, indicating better clustering. To determine the DBI for an instance of clustering, the dispersion measure S_i , inter-cluster distance D_{ij} , and cluster similarity measure R_{ij} are used [63]. The subscript i and j are used to index the different clusters.

The dispersion measure S_i for each cluster C_i with the number of points T_i is defined in Equation 5.1, where $\mathbf{c}_i$ is the centroid of the cluster (in our case obtained from the k-means clustering algorithm) and q controls how dispersion is measured [63]. For $q = 1$, S_i is the average Euclidean distance of points in C_i to centroid $\mathbf{c}_i$; for $q = 2$, S_i is the standard deviation of these distances. The variable $\mathbf{x}_j$ is a specific point in the cluster C_i . The dispersion measure is determined for each cluster, where a more compact cluster will return a smaller S_i . We use $q = 1$ for the dispersion measure of the DBI.

$$S_i = \left(\frac{1}{T_i} \sum_{j=1}^{T_i} |\mathbf{x}_j - \mathbf{c}_i|^q \right)^{1/q} \quad (5.1)$$

The distance between clusters C_i and C_j is known as the inter-cluster distance D_{ij} , defined in Equation 5.2 [63]. Here, $\mathbf{c}_i$ is the centroid of C_i , and p controls the distance metric used. For $p = 1$, the Manhattan distance is used; for $p = 2$, the Euclidean distance is used. This distance is determined between each pair of clusters. We use the Euclidean distance, thus $p = 2$, to score the clusters. Equation 5.2 is equivalent to the formula for the Minkowski distance between two points [63].

$$D_{ij} = \|\mathbf{c}_i - \mathbf{c}_j\|_p \quad (5.2)$$

The cluster similarity measure R_{ij} in Equation 5.3 gives the similarity between clusters C_i and C_j , using the earlier defined measures and distances [63]. This metric increases as the two clusters become closer or more dispersed, representing greater similarity.

$$R_{ij} = \frac{S_i + S_j}{D_{ij}} \quad (5.3)$$

For each cluster C_i , the maximum similarity to any other cluster is:

$$R_i = \max_{j \neq i} R_{ij} \quad (5.4)$$

The overall measure $\bar{R}$ (i.e. final DBI score) is obtained by averaging these maximum similarities, where N is the total number of clusters [63]. A lower $\bar{R}$ indicates a better fit of clustering, where each cluster is compact and well separated from other clusters.

$$\text{DBI} = \bar{R} = \frac{1}{N} \sum_{i=1}^N R_i \quad (5.5)$$

For our clustering classifier, the clustering instance that returns the lowest DBI is accepted as the modulation order of the received signal.

With the best modulation order of the received signal identified, the modulation scheme of the signal can usually be derived depending on the dataset. For our dataset, if the modulation order is greater than 8, the signal is identified as QAM; if it is below 8, the signal is identified as PSK. However, a differentiating process between PSK and QAM is required, as there is some overlap, as both 8-PSK and 8-QAM are present in the dataset. These two modulation schemes can be differentiated by examining the expected distribution of symbol

positions in the constellation diagram. A PSK signal forms a circle around the origin, where each symbol has an equal distance to the origin, whereas a QAM signal forms a grid, where each symbol does not have an equal distance to the origin. The distances between the origin and the closest cluster centroid and the farthest cluster centroid are used to distinguish between PSK and QAM. If the difference between these two distances is below a threshold, then the signal is PSK, otherwise it is QAM.

Figure 5.1 provides visual clarification of how 8-PSK and 8-QAM can be differentiated, where on each constellation diagram the green arrow points to the closest centroid and the red arrow points to the furthest centroid. Clustering has already been performed on these constellation diagrams with each centroid marked as a black X.

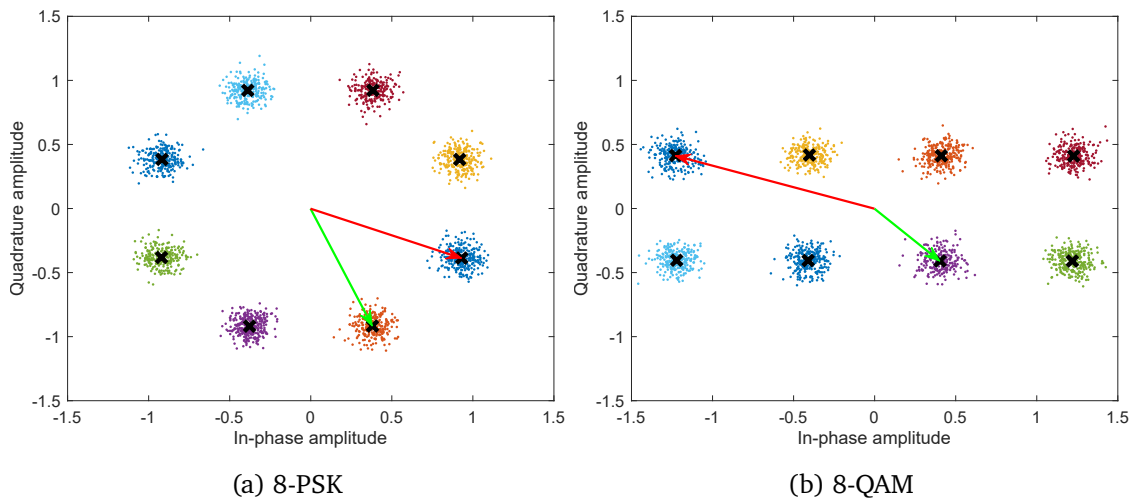


Figure 5.1: Differentiating between PSK and QAM modulation schemes for clustering classifier.

The differentiating threshold is set at 0.3568. This threshold is determined by analysing an 8-QAM constellation diagram with unit average power. We measure the Euclidean distance of the closest and farthest symbol position from the origin in the constellation. The difference between these two distances is computed to obtain the midpoint distance, which is approximately the distance to a point between the symbols. The midpoint distance is halved and set to 97.27% to avoid misclassification, yielding a value of 0.3568. When classifying a received signal, if the difference in distances between the closest and furthest cluster centroids is below this threshold, then the signal is identified as PSK, otherwise the distance is too large and it is identified as QAM.

A flowchart of the overall classification process using clustering is given in 5.2.

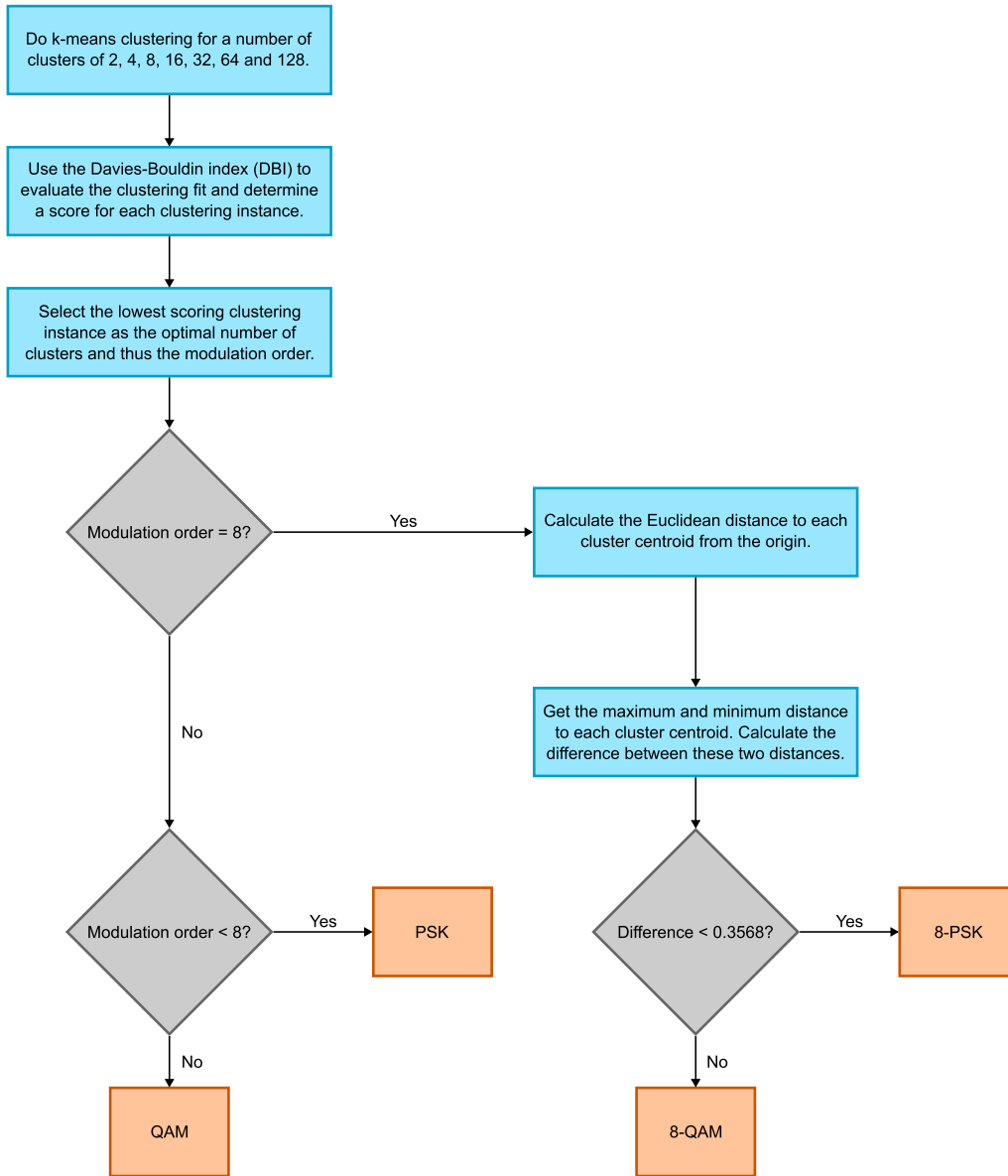


Figure 5.2: Process of modulation classification using clustering.

5.2.2 QHLRT

To cover the likelihood-based approach to AMC, the QHLRT is selected. The QHLRT is a variant of the HLRT that uses method-of-moments (MoM) estimators to estimate channel parameters such as noise, phase offset, and amplitude change [20], [64]. Using these estimators MoM enables the QHLRT to have significantly reduced computational complexity compared to HLRT, while maintaining similar classification performance. In QHLRT, the likelihood of each modulation type is calculated by averaging the unknown constellation points and inserting the MoM estimates for the unknown channel parameters.

MoM estimators rely on moments to estimate the unknown channel parameters [20], [64].

Table 5.1: Parameter $b^{(i)}$ for different modulation orders of PSK and QAM.

Modulation type	Value for $b^{(i)}$
M -PSK	1.000
8-QAM	1.444
16-QAM	1.320
32-QAM	1.310
64-QAM	1.381
128-QAM	1.342

These moments have been introduced previously in Section 2.3.2 as part of the features that can be used for AMC. Moments can be calculated as follows, where M_{pq} denotes the (p, q) -th order moment, $\mathbf{r}$ is the received complex signal of I/Q points, $\mathbf{r}^*$ is the complex conjugate of $\mathbf{r}$, and $E[\cdot]$ is the statistical expected or average value [65].

$$M_{pq} = E[\mathbf{r}^{p-q} \mathbf{r}^{*q}] \quad (5.6)$$

The QHLRT uses the parameter $b^{(i)}$ as part of its MoM estimates of the channel parameters, where each QAM modulation scheme has a unique value and the PSK modulation schemes have the same value [20]. The superscript (i) indicates that this parameter is for a specific modulation type that is indexed with i . Equation 5.7 provides the calculation required to determine the $b^{(i)}$ for a specific modulation type, where $s_k^{(i)}$ are the noiseless signal constellation points of that modulation type [20].

$$b^{(i)} = \frac{E[|s_k^{(i)}|^4]}{\left(E[|s_k^{(i)}|^2]\right)^2} \quad (5.7)$$

Table 5.1 provides the parameter $b^{(i)}$ for all the different configurations of PSK and QAM that are present in our dataset. Note that these values are dependent on the shape of the distribution of symbol positions. Different configurations of the same order QAM modulated signal can return different values for the parameter $b^{(i)}$. The modulation order does not affect the parameter $b^{(i)}$ for PSK modulated signals.

The channel parameters of amplitude α , noise N , and phase ϕ are the possible unknown parameters for our signal model. Estimates of these channel parameters are carried out for each modulation type in the pool of possible modulation types, as provided in Equation 5.8 to Equation 5.11 [20]. A hat symbol $\hat{\cdot}$ placed over a parameter indicates that it is an estimate of the corresponding true value of that parameter. Theoretically, using the true modulation type of the received signal results in the most precise estimates, as it best represents the

signal's symbol distribution. As shown, the equation for estimating the channel phase differs between PSK and QAM modulation types.

$$\hat{\alpha}^{(i)} = \left(\frac{\hat{M}_{42} - 2\hat{M}_{21}^2}{b^{(i)} - 2} \right)^{1/4} \left(E[|s_k^{(i)}|^2] \right)^{-1/2} \quad (5.8)$$

$$\hat{N}^{(i)} = \hat{M}_{21} - \left(\frac{\hat{M}_{42} - 2\hat{M}_{21}^2}{b^{(i)} - 2} \right)^{1/2} \quad (5.9)$$

$$\hat{\phi}_{\text{M-PSK}}^{(i)} = M_i^{-1} \arg \left(\sum_{k=1}^K r[k]^{M_i} \right) \quad (5.10)$$

$$\hat{\phi}_{\text{M-QAM}}^{(i)} = 4^{-1} \arg \left(- \sum_{k=1}^K r[k]^4 \right) \quad (5.11)$$

After the channel parameters have been successfully estimated for a specific modulation type, the likelihood $\mathcal{L}^{(i)}$ of the received signal using that modulation type can be determined in Equation 5.12 [20]. In this equation, the absolute difference between the signal constellation points $s_m^{(i)}[k]$ of the possible modulation type and the received signal symbol $r[k]$ is calculated for all K received signal symbols. Since we do not know which received signal symbol matches which expected signal constellation point, the difference is also calculated for each possible signal constellation point M_i equivalent to the order of the modulation type. The higher the order of the modulation type, the more computationally expensive it is to calculate the likelihood function.

$$\mathcal{L}^{(i)}(\mathbf{r}) = \prod_{k=1}^K \frac{1}{M_i} \sum_{m=1}^{M_i} \frac{1}{\pi \hat{N}^{(i)}} \exp \left(- \frac{1}{\hat{N}^{(i)}} \left| r[k] - \hat{\alpha}^{(i)} e^{j\hat{\phi}^{(i)}} s_m^{(i)}[k] \right|^2 \right) \quad (5.12)$$

The log-likelihood format of the likelihood equation is used for computational efficiency and numerical stability when implementing the QHLRT classifier. The product operator in the likelihood equation can cause instability when working with small values for the I/Q points, especially for a large signal frame length. Equation 5.13 gives the log-likelihood format of Equation 5.12. Taking the log of the equation does not affect our goal of maximising the likelihood equation (i.e. finding the modulation type that returns the highest likelihood) as logarithms are strictly increasing functions and the same result can be obtained from both equations.

$$\log \left(\mathcal{L}^{(i)}(\mathbf{r}) \right) = \sum_{k=1}^K \log \left(\sum_{m=1}^{M_i} \exp \left(-\frac{1}{\hat{N}^{(i)}} \left| r[k] - \hat{\alpha}^{(i)} e^{j\hat{\phi}^{(i)}} s_m^{(i)}[k] \right|^2 \right) \right) - \log \left(M_i \pi \hat{N}^{(i)} \right) \quad (5.13)$$

In our study, we determine only the channel noise parameter N and fix the other channel parameters to their true values. Our generated dataset consists of a simulated AWGN channel which has a true channel amplitude of 1 and a true channel phase offset of 0° . In our implementation, only the noise parameter is estimated to ensure a fair comparison with the other AMC methods.

5.2.3 KNN with higher-order cumulants

The features-based approach to AMC consists of a KNN using higher-order cumulants to differentiate modulation types.

The KNN algorithm is a straightforward machine learning method used for classification and regression tasks [28, pp. 16–18]. It is a non-parametric method which does not presume any specific functional form for the underlying data distribution, relying instead on training data for predictions. The classification process begins by examining the k points in the training set that are closest to the input $\mathbf{x}$ using a specific distance metric, typically Euclidean distance. After the neighbours have been identified, a majority vote is carried out, and the class that occurs the most often among the neighbours is selected as the class of the input. Instead of only conducting a majority vote, fractional estimates can be determined using the closest neighbours to determine the probability that the input is of a specific class [28, pp. 16–18]. In our case, we are only concerned with the majority vote, as it determines the input's classification.

Formally, a labelled training set from a dataset can be defined as in Equation 5.14, where $\mathbf{x}_i \in \mathbb{R}^D$ denotes an input D -dimensional feature vector, $y_i \in \mathcal{Y}$ denotes the corresponding class label in the label space $\mathcal{Y}$, and N is the number of training examples [28, pp. 16–18].

$$\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^N, \quad (5.14)$$

To classify a new input $\mathbf{x}$, the KNN algorithm calculates the distance between $\mathbf{x}$ and every training sample in $\mathcal{D}$ using Euclidean distance as shown in Equation 5.15. Other distance metrics, such as Manhattan, can also be used.

$$d(\mathbf{x}, \mathbf{x}_i) = \|\mathbf{x} - \mathbf{x}_i\|_2. \quad (5.15)$$

The k closest samples are designated as the k nearest neighbours. The predicted class label $\hat{y}$ is determined by a majority vote among the nearest neighbours as given in Equation 5.16, where $\mathcal{N}_k(\mathbf{x})$ is the set of indices of the k nearest neighbours. $\mathbb{I}(\cdot)$ is the indicator function, where $\mathbb{I}(A) = 1$ if A is true, 0 otherwise [28, pp. 16–18].

$$\hat{y} = \arg \max_{c \in \mathcal{Y}} \sum_{i \in \mathcal{N}_k(\mathbf{x})} \mathbb{I}(y_i = c), \quad (5.16)$$

For the KNN to work in an AMC context, it requires features that can be used to discriminate between modulation schemes. We select higher-order cumulants to be used as these discriminative features, where the cumulants are arranged as a feature vector for the KNN. The cumulants of a probability distribution of a random variable are higher-order statistics derived from the moments of that random variable. The cumulant generating function given in Equation 5.17 is the logarithm of the moment generating function, where X represents a real-valued random variable and E is a function that returns the expected or mean value [66].

$$C_t(X) = \log E[e^{tX}] \quad (5.17)$$

The n -th order cumulant c_n is determined from the power series expansion of the cumulant generating function as provided in Equation 5.18 [66]. The n th-order cumulant is obtained by differentiating the expansion n times at $t = 0$.

$$C_t(X) = \sum c_n \frac{t^n}{n!} \quad (5.18)$$

For a normal distribution, its cumulant generating function is given in Equation 5.19, where X is the random variable, μ is the mean, and σ is the standard deviation [66]. Notice that taking the 3rd or higher order derivative of this function with respect to t will result in zero. This highlights an important property of cumulants, which is that third and higher-order cumulants of a normal distribution, such as AWGN in a signal context, are zero. This property is ideal for using cumulants for AMC, as we are interested in the features of the original modulated signal. Cumulants have the cumulative property where $c_n(X + Y) = c_n(X) + c_n(Y)$, such that the zero cumulant of AWGN does not affect the cumulant of the random process we are interested in. As shown in Equation 2.1, the noise vector from an AWGN channel can be viewed as an additive effect on the transmitted signal. When calculating the cumulants of a received signal, which has the noise vector added, we know that the third and higher-order cumulants of the noise from the AWGN channel will be zero.

$$C_t(X) = \mu t + \sigma^2 \frac{t^2}{2} \quad (5.19)$$

Cumulants are rather difficult to compute using the cumulant generating function for received signals, especially if the probability distribution is unknown, and moments can be used to simplify the calculation if the probability distribution for the moments and cumulants is the same. Since the moments are calculated from the received signal symbols using Equation 5.6, the cumulants use the same probability distribution. Furthermore, the above equations are for a real-valued random variable, whereas in the AMC context we operate on complex-valued signals. From [65], the equations for the higher order cumulants C_{40} , C_{41} , C_{42} , C_{60} , C_{61} , C_{62} , and C_{63} are provided, where C_{xy} is the (x, y) -th order cumulant for complex-valued signals:

$$C_{40} = M_{40} - 3M_{20}^2 \quad (5.20)$$

$$C_{41} = M_{40} - 3M_{20}M_{21} \quad (5.21)$$

$$C_{42} = M_{42} - 2M_{21} - |M_{21}|^2 \quad (5.22)$$

$$C_{60} = M_{60} + 30M_{20}^3 - 15M_{20}M_{40} \quad (5.23)$$

$$C_{61} = M_{61} + 30M_{20}^2M_{21} - 10M_{20}M_{41} \quad (5.24)$$

$$C_{62} = M_{62} - 6M_{20}M_{42} + 24M_{21}^2M_{22} - 8M_{21}M_{41} + 6M_{20}^2M_{22} - M_{22}M_{40} \quad (5.25)$$

$$C_{63} = M_{63} + 18M_{20}M_{21}M_{22} + 12M_{21}^3 - 9M_{21}M_{42} - 3M_{20}M_{43} - 3M_{22}M_{41} \quad (5.26)$$

5.3 Verification of implementation

The QHLRT and KNN classifiers are based on previous research done in the field. Before using these methods in our study, we verify their implementations. We implement the methods as described, evaluate them on the same inputs, and verify that we obtain the

same results as those provided in the original sources. If the results are similar, we can conclude that the method has been verified and correctly implemented.

5.3.1 QHLRT

For our verification of the QHLRT classifier, we follow the same approach used by Hameed *et al.* [64]. We follow the experimental setup and examine if we obtain the same results as reported in the source paper.

An equivalent simulation setup to that of Hameed *et al.* is created to test the QHLRT in identifying BPSK and QPSK signals [64]. They used 10,000 Monte Carlo trials to randomly generate modulated signal frames with SNRs ranging from -8 to 10 dB in 1 dB increments, using BPSK and QPSK. For each signal frame, the signal amplitude is set to 1 and the phase offset is uniformly distributed over the $[-\pi/4, \pi/4)$ range. The signal amplitude and phase are fixed over the entire signal frame, which has a length of 100 symbols. For each Monte Carlo trial, the QHLRT is used to classify the modulation type of the received signal, where the QHLRT has the pool of possible modulation types for the received signal as only BPSK and QPSK. We used the same number of Monte Carlo trials for our verification process. Furthermore, we use the log-likelihood format of the likelihood function as provided in Equation 5.13.

We determine the average probability of correct classification of the BPSK and QPSK signals over the range of SNRs using the QHLRT. The result of this test is provided in Figure 5.3, where the probability of correct classification is plotted against the SNR. As in the graph configuration used by Hameed *et al.* to show their results, we focus on the upper end of the curve, where the probability of correct classification is between 0.8 and 1. The classification results obtained by Hameed *et al.* is included for easy comparison. A similar curve is obtained to that achieved in the original paper, but our probability values are higher than those obtained by the original authors. Focusing on the line points shown in the figure (SNR is equal to or greater than -2 dB), the two curves differ by a mean squared error of 0.0003 and a maximum deviation of 0.0504. The correlation coefficient is 0.9997, indicating a very high similarity.

The resulting classification performance of the QHLRT that we implemented is mostly consistent with what is reported in the original paper. We conclude that the QHLRT is correctly verified, as an equivalent classification curve is obtained.

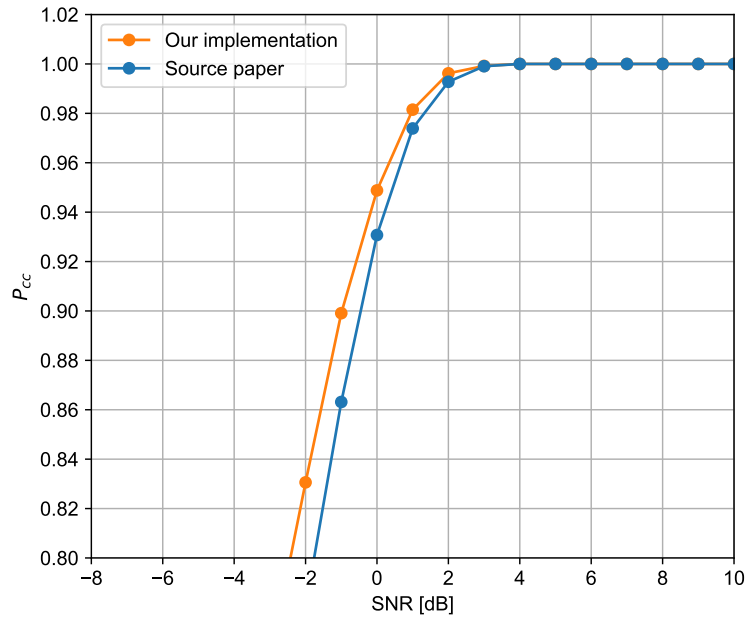


Figure 5.3: Classification performance of QHLRT classifier as SNR varies for verification.

5.3.2 KNN with higher-order cumulants

Two distinct sources are used to verify our implementation of the KNN using higher-order cumulants. In [5, p. 114], Zhu and Nandi describe the implementation and evaluation of a KNN using higher-order cumulants, but they do not provide the equations for all of the cumulants that they use. We use the equations provided in [65], where a KNN with higher-order cumulants is also implemented. The same general approach used by Zhu and Nandi in [5, pp. 110–120] is followed for verification, using the cumulant equations in [65].

The same dataset that was used in [5, p. 110] for testing is recreated. The synthetic dataset consists of the modulation types BPSK, QPSK, 8-PSK, 4-QAM, 16-QAM, 64-QAM, 2-PAM, 4-PAM, and 8-PAM in an AWGN channel. Unit average power is used for the QAM signals. Specifically in [5, p. 110], the dataset is not fully described, but they do provide MATLAB code that provides the frame length and the number of signal frames used. Signal frames are generated with a SNR of -20 to 20 dB in increments of 1 dB, where at each SNR there are 1030 frames with a frame length of 1024 for each modulation type. The dataset is divided into a training set comprising 30 (2.91%) of the signal frames and a test set comprising 1000 (97.08%) of the signal frames across all SNRs. Class balance is maintained in both the training and test sets. No validation set is created because no hyperparameter tuning is performed.

The cumulants C_{40} , C_{41} , C_{42} , C_{60} , C_{61} , C_{62} , and C_{63} are used to differentiate the modulation types. A feature space is constructed for the KNN using cumulants calculated from the signals in the training set. The number of neighbours used by the KNN is not specified in [5, p. 114], so we repeat the verification for the values of 1 to 10. Euclidean distance is

used to determine the closest neighbours as Zhu and Nandi did not explicitly mention the distance metric that they used.

As in the approach used by [5, p. 114], we construct a feature space for the KNN using the training set for a specific SNR before evaluating it on the test set at that SNR. Figure 5.4 compares the classification performance of our implementation of the KNN against the implementation in [5, p. 120]. Each number of neighbours (k) is given a distinct curve on the graph, although some of the curves are excluded from the graph to not overcrowd it. The number of neighbours of 5 performed best overall, whereas the number of neighbours of 7 showed the greatest similarity to the curve in the source paper. The classification performance of our implementation is similar to that reported in the source paper, but there are differences between the two sets of results. Examining the number of neighbours ($k=5$) curve and the source paper result, the two curves differ with a mean squared error of 26.915 and a maximum deviation of 9.885, while the correlation coefficient is 0.998. This large deviation occurs at low SNRs, where our implementation achieves higher accuracies than those reported in the source paper. Our KNN is also more unstable as seen with the spikes in the curve compared to the overall smoother curve in [5, p. 120].

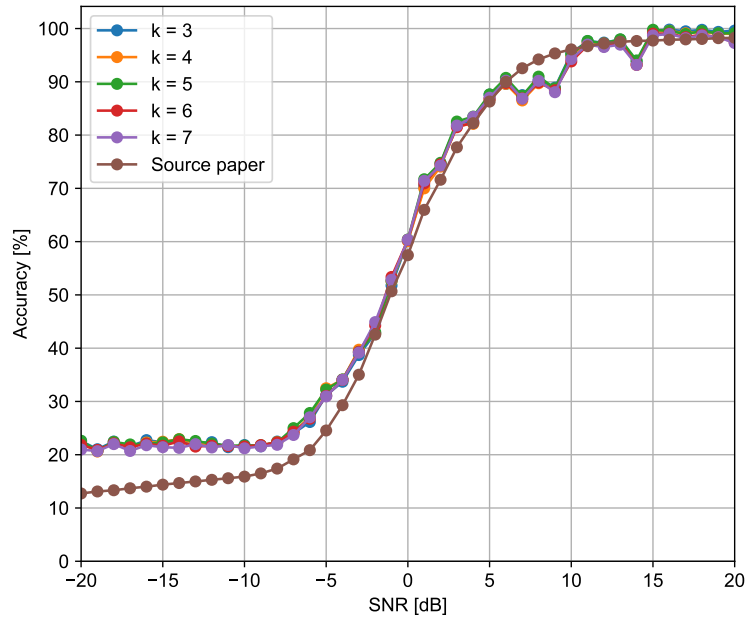


Figure 5.4: Classification performance of KNN classifier that uses higher-order cumulants as SNR varies for verification.

Since there are uncertainties regarding the approach Zhu and Nandi used in their implementation of a KNN using higher-order cumulants, there are many potential reasons for the differences in behaviour between the two KNNs. Potential mismatches are differences in the equations used for the cumulants, a different number of neighbours for the KNN, and possible differences in the dataset.

Despite these changes in classification performance between the two curves, we conclude

that the behaviour of our implementation of the KNN is consistent with the behaviour of the KNN in [5, pp. 110–120]. The KNN classifier has been successfully verified.

5.4 Training process of KNN

Before the KNN using higher-order cumulants can be evaluated for classification performance on unperturbed and attacked signals, its feature space must be constructed. The other NDL methods do not require any training or preparation before classification, as they are specifically designed for AMC.

The KNN is set up to use the cumulants of C_{40} , C_{41} , C_{42} , C_{60} , C_{61} , C_{62} , and C_{63} as used in the verification of the method. Unlike the process used to verify the KNN, a single feature space is constructed for the KNN, which includes the full training set from the dataset described in Section 3.2, regardless of SNR. A heuristic grid search was conducted for the number of neighbours for the KNN classifier from 1 to 10. The average accuracy on the validation set over all the SNRs was used to determine the best number of neighbours. For our simulated dataset, the number of neighbours of 5 resulted in the best average accuracy, although all of the values between 5 and 10 were close in average accuracy. Number of neighbours for the KNN classifier is set as 5 ($k = 5$), using Euclidean distance for determining the closest training sample to an input.

5.5 Results

The clustering, QHLRT, and KNN classifiers are tested on the simulated dataset to determine their classification performance, following the same evaluation process that was used for the PET-CGDNN classifier, as given in Section 4.5. The KNN and QHLRT classifiers are evaluated on the full test set without adversarial attacks, with fixed-size attacks, and with varying-size attacks. We use the reduced test sets without adversarial attacks, with fixed-size attacks, and with varying-size attacks for the clustering classifier due to its computationally demanding classification process.

5.5.1 Performance on unperturbed signals

Figure 5.5 shows the performance of the NDL classifiers on unperturbed signals. The result achieved in the previous chapter with PET-CGDNN is also included for easy comparison. The clustering classifier is evaluated on a reduced test set and is thus shown with a different line style compared to the other classifiers. As shown in the figure, the NDL classifiers do not perform as well as the DL classifier, requiring a higher SNR to accurately classify the

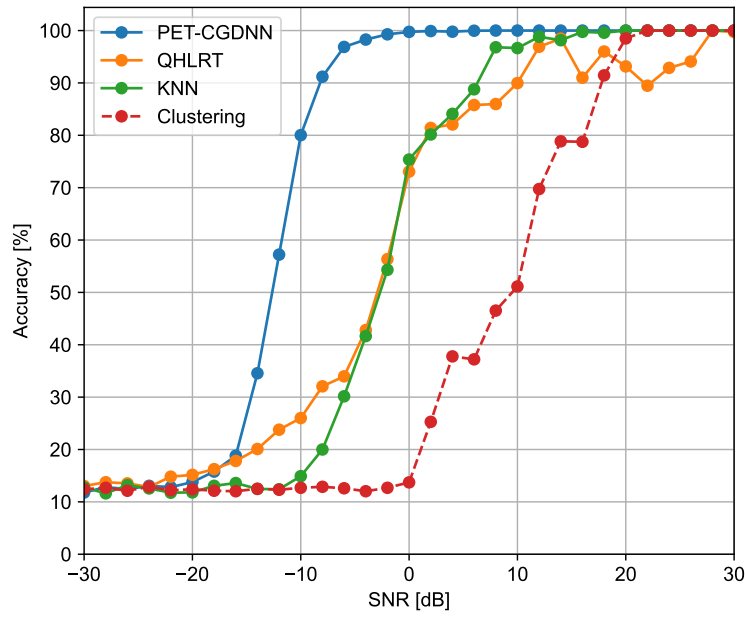


Figure 5.5: Classification performance of NDL classifiers as SNR varies on unperturbed signals.

signals. The QHLRT and KNN classifiers outperform the clustering classifier as a higher SNR is required for well-defined clusters to be formed enabling the clustering classifier to accurately identify the modulation scheme. At high SNRs the performance of the QHLRT classifier is unstable as evidenced by the dips in classification accuracy.

A confusion matrix is plotted to show the performance of the QHLRT at a SNR of 22 dB to investigate the dips in classification accuracy. The confusion matrix shows how each class is correctly and incorrectly classified, enabling us to identify which classes are misclassified. Figure 5.6 provides the confusion matrix, where each value in each cell is normalised to be in the range of $[0, 1]$ with a 1 indicating a classification accuracy of 100%. As shown in the figure, the instability is due to the QHLRT struggling to identify nested QAM constellations, particularly between 16-QAM and 64-QAM. It has been observed that the GLRT variant has a problem in identifying nested constellations at high SNRs, as discussed in [67], which is supposed to be addressed with the HLRT variant. The QHLRT is based on the HLRT and is thus also supposed to be more capable in identifying nested constellations [20], [64], but it appears that this issue still occurs.

5.5.2 Performance on attacked signals

The clustering, QHLRT, and KNN classifiers are evaluated on the fixed-size and varying-size white-box adversarial attacks of FGM, PGD, and DeepFool.

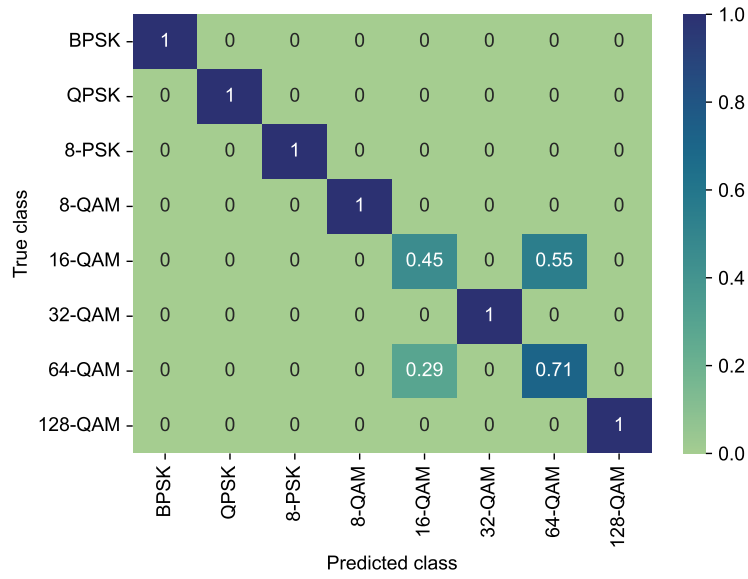


Figure 5.6: Confusion matrix of classification performance of QHLRT on signals without adversarial attacks at 22 dB SNR.

Fixed-size attacks

The NDL classifiers are assessed on the test set using fixed-size attacks from FGM and PGD, and attacks with DeepFool, using the same evaluation method as for the PET-CGDNN. As discussed previously, the DeepFool attack is not considered a fixed-size attack, but it is included here with the other fixed-size attacks because its strength cannot be easily scaled.

The classification performance of the NDL classifiers on the fixed-size adversarial attacks is given in Figure 5.7. Each of the NDL classifiers is largely unaffected by the FGM and PGD attacks, with the QHLRT classifier showing the highest vulnerability to these attacks at high SNRs. The vulnerability may stem from the issue mentioned earlier of identifying nested constellations, which could make the QHLRT more susceptible to attacks in general. Fixed-size attacks generated on the PET-CGDNN using FGM and PGD do not seem to target the same features that the NDL classifiers use and thus have limited transferability. However, the DeepFool attacks are potent enough to also significantly deteriorate the performance of the NDL classifiers, as the DeepFool attacks did with the PET-CGDNN. Section 3.4 presented constellation diagrams showing how DeepFool attacks significantly altered the distribution of the I/Q symbols. The alteration is significant enough that the NDL classifiers also struggle to identify the modulation type of the signal.

Varying-size attacks

The NDL classifiers are also tested on the varying-size attacks with a range of PNRs using FGM and PGD.

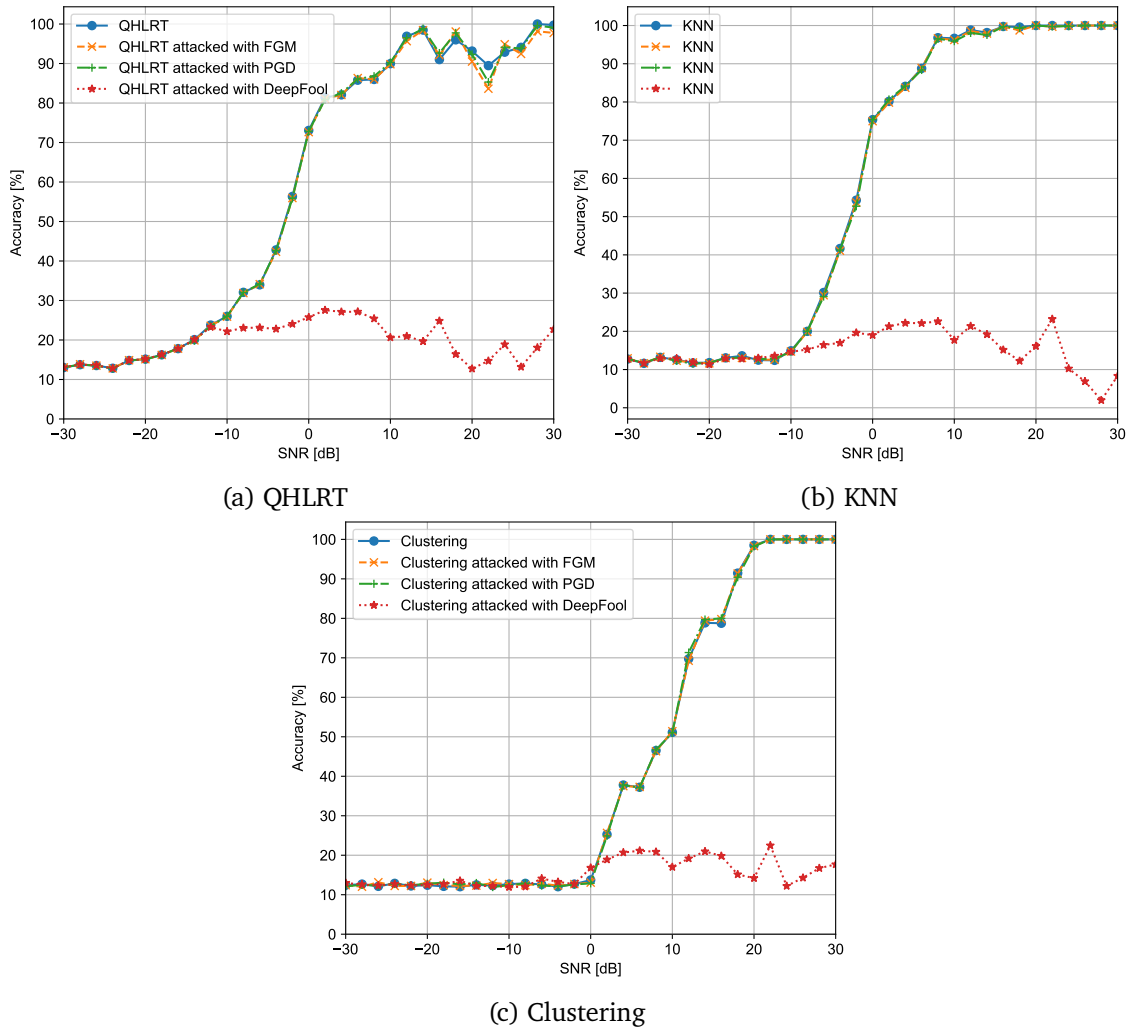


Figure 5.7: Classification performance of NDL classifiers on fixed-size attacks as SNR varies.

In Figure 5.8, the classification performance of the clustering classifier, the QHLRT, and the KNN with higher-order cumulants are shown as the PNR varies for specific SNR values. The performance of the classifiers is largely unaffected by the attacks, with only slight dips in performance for the QHLRT and KNN at high PNRs at 0 dB SNR. This result supports the observation that the FGM and PGD attacks have limited transferability to NDL classifiers, as seen with the fixed-size attacks. Comparing these graphs to the equivalent graphs obtained for the PET-CGDNN in Section 4.5.2, shows that the PET-CGDNN is the only classifier significantly affected by these attacks.

5.6 Chapter summary

To summarise the chapter, a clustering classifier, the QHLRT, and a KNN was investigated as NDL approaches to AMC. The clustering classifier classification process starts with using

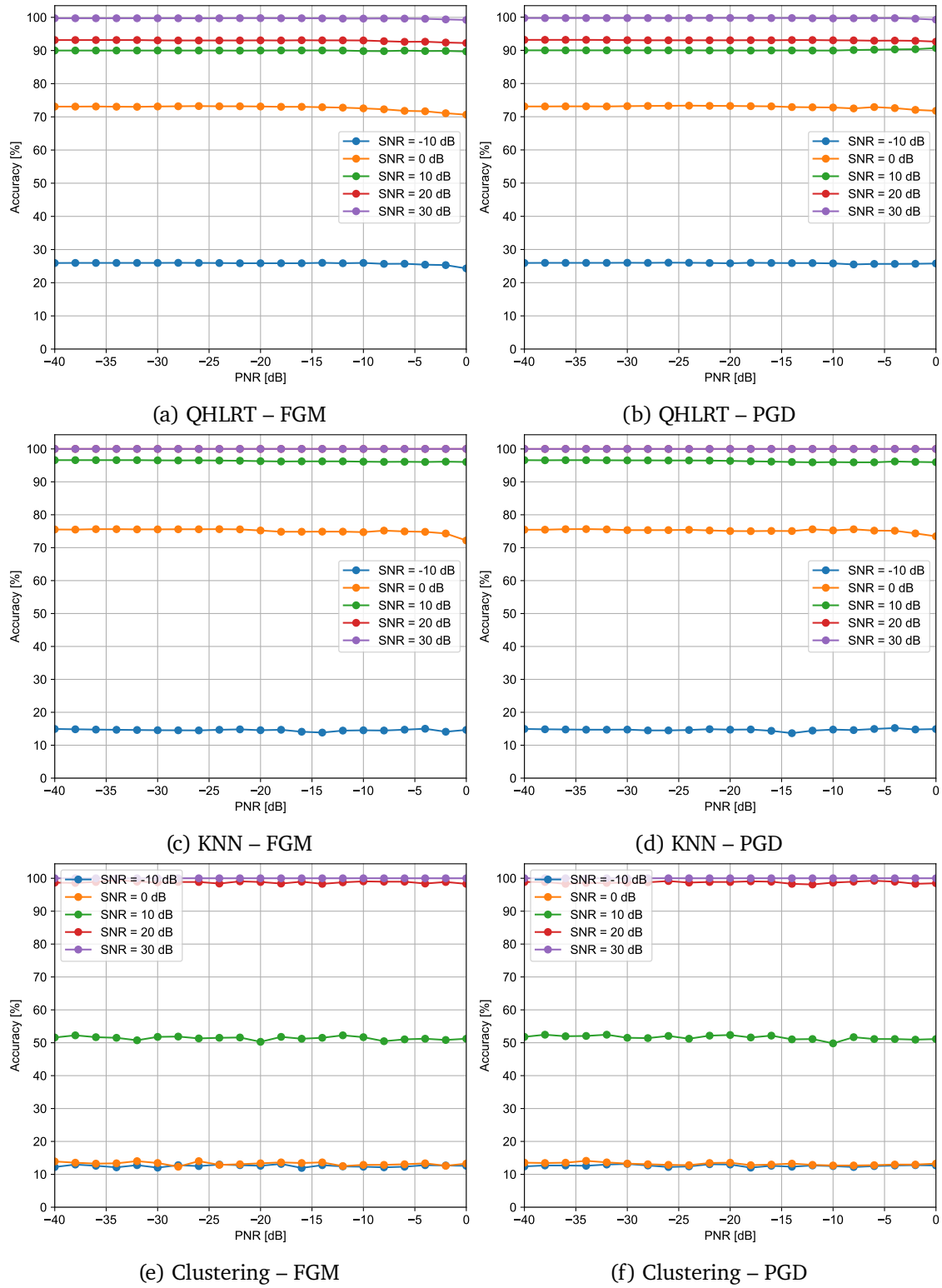


Figure 5.8: Classification performance of NDL classifiers on varying-size attacks as PNR varies for specific SNRs.

the k-means clustering algorithm to fit a different number of clusters for multiple instances. Afterwards, the DBI is used to score the quality of the fit of the clusters and the best-scoring

number of clusters was accepted as the modulation order. A final decision-making process was used to differentiate between PSK and QAM. The QHLRT classifier is a variant of the HLRT method that uses MoM estimators to estimate channel parameters. After the parameters have been estimated, the likelihood of each modulation type in the pool of possible modulation types are calculated. The modulation type that results in the highest likelihood result is the classified modulation type. Lastly, the KNN classifier uses higher-order statistics, specifically in the form of higher-order cumulants, that is calculated on the signals to differentiate between modulation types. The QHLRT and KNN were verified by comparing their implementations to the papers on which they are based. A feature space was constructed for KNN to enable classification on our dataset.

Evaluating the selected NDL classifiers on the dataset both with and without adversarial attacks present, revealed the strong robustness that the classifiers have to the FGM and PGD attacks. These attacks were shown to have limited transferability to the features utilised by the NDL classifiers. The DeepFool attacks were significant enough to cause the NDL classifiers to become unreliable for modulation classification. On the unperturbed signals, it was shown that the DL classifier greatly outperforms the NDL classifiers, highlighting why recent research in AMC has focused on DNNs.

Chapter 6

Combination of DL and NDL techniques

6.1 Introduction

In this chapter, we investigate the combination of DL and NDL techniques. We first discuss the approach of the neural rejection technique that acts through a SVM, which is the basis of our hybrid classifier that combines DL and NDL techniques. Furthermore, background information for the required understanding of SVMs is provided. Our configuration of the hybrid classifier, which combines the PET-CGDNN and the KNN that uses higher-order cumulants with a SVM acting as a switching mechanism, is explored. The training and hyperparameter tuning of the SVM is given. Different configurations of the SVM are tested to determine a good configuration for the hybrid classifier. Finally, the hybrid classifier is evaluated to determine whether improvements in robustness against adversarial attacks are achieved, while maintaining the good classification performance of the PET-CGDNN on unperturbed signals.

6.2 Design of hybrid classifier

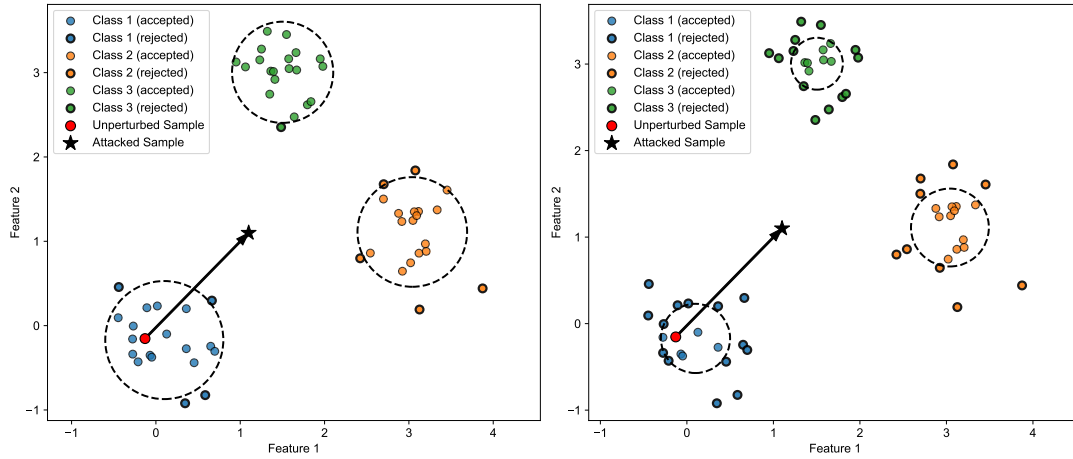
In previous chapters, DL and NDL classifiers were investigated to examine their robustness against adversarial attacks. The NDL classifiers were found to have greater robustness against attacks, but they did not have the same good performance against unperturbed signals as the DL classifier. This observation leads us to hypothesise whether it is possible to combine the two approaches to modulation classification, leveraging the advantages of each approach.

6.2.1 Neural rejection

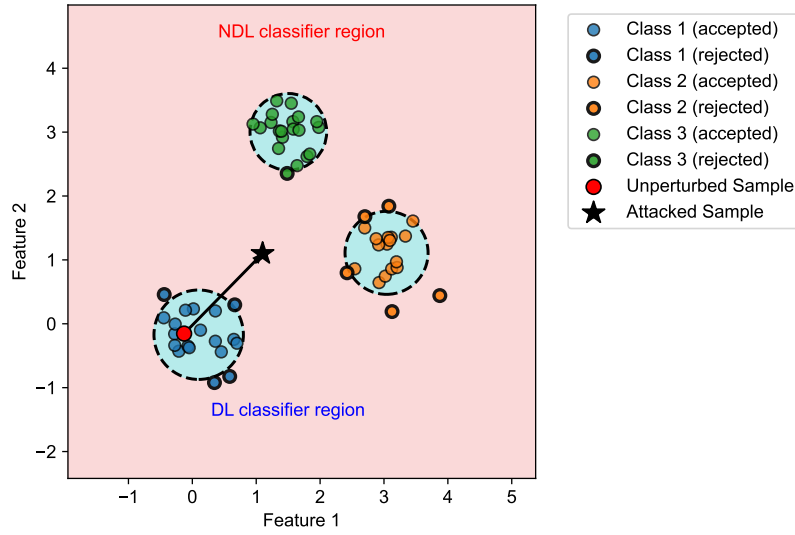
Zhang *et al.* used the neural rejection system for AMC to determine whether the input to a DL model was adversarial or not [60]. The neural rejection system is based on the idea of rejecting samples depending on their distance from the typical classification regions in the feature space of the DNN [68], [69]. An adversarial attack on a sample will move it further away from the region occupied by the rest of the samples. A SVM uses the network layers of a pre-trained DNN to detect anomalous samples by comparing them with the rest of the training samples [68]. For the neural rejection system, the last feature layer of the DNN is used, but with deep neural rejection, multiple network layers are used [68]. After receiving the feature layer as input, the SVM generates decision scores for all possible classes. If the maximum decision score is below a specified threshold θ , the input is considered adversarial and rejected. This neural rejection approach does not require any adversarial examples in advance and is thus less computationally demanding [68].

We want to use the neural rejection system as a switching mechanism to decide whether to use the DNN or the NDL classifier to identify the modulation type of a signal, in the possible presence of adversarial attacks. Instead of only rejecting input samples if they are adversarial or not, the hybrid classifier redirects the sample to the NDL classifier, as it has more robustness against attacks, and if the sample is not rejected then the DL classifier is used.

Figure 6.1 explains the concept of the neural rejection system visually. The feature space of a trained DNN for a specific network layer, where there are three possible classes, are shown. The feature space of a DNN is not to be confused with the I/Q constellation diagram, as the feature space is a representation of the input data as transformed by the neural network's nodes at some layer. The SVM's rejection threshold is shown by a bounding circle, where samples that fall outside of the circle are seen as adversarial and samples inside the circle are not seen as adversarial. Additionally, an adversarial sample is demonstrated by an unperturbed sample moving away from the usual classification region, as shown by an arrow. The higher the PNR is, the more the sample would be moved away from the typical region of where that class lies. Figure 6.1a uses a small rejection threshold, evident by the large bounding circles, while Figure 6.1b shows a large rejection threshold. A balance between rejecting adversarial samples and unperturbed samples must be achieved to achieve a good rejection ratio. Figure 6.1c illustrates the concept of the hybrid classifier, which uses the SVM as a switching mechanism. Samples that fall outside of the bounding circles for the classes are given to the NDL classifier as they have been identified as adversarial, while samples inside the bounding circles are identified using the DL classifier.



(a) Rejection of samples with a small rejection threshold (b) Rejection of samples with a large rejection threshold



(c) Hybrid classifier concept

Figure 6.1: Conceptual visualisation of neural rejection system on a feature space with three distinct classes. Bounding circles are drawn to indicate rejection threshold.

6.2.2 Support vector machines

One of the core components of this neural rejection method is the SVM. SVMs are supervised machine learning algorithms designed to find the hyperplane that best separates data points belonging to different classes [28, pp. 496–505]. The hyperplane is a decision boundary separating classes, and the SVM aims to maximise the margin, which is the gap between the hyperplane and the nearest data points. While maximising the margin, the SVM also aims to minimise classification errors. The nearest data points to the hyperplane are known as the support vectors, which are used to define the hyperplane.

Consider a linear case in a binary classification context using the labelled training set $(\mathbf{x}_i, y_i)$

(as defined in the previous chapter in Equation 5.14), a SVM is trying to determine a decision function $f(\mathbf{x})$ as given in Equation 6.1 [28, p. 496]. The variables $\mathbf{w}$ and w_0 define the hyperplane, where $\mathbf{w}$ is a weight vector and the variable w_0 is the bias term. Furthermore, the variable $\mathbf{x}$ is the input vector. This linear decision function must maximise the margin while minimising the classification errors, leading to an optimisation problem.

$$f(\mathbf{x}) = \mathbf{w}^T \mathbf{x} + w_0 \quad (6.1)$$

The optimisation problem using the hinge loss is defined in Equation 6.2, where $(1 - y_i f(\mathbf{x}_i))_+$ is the hinge loss and C is a regularisation parameter that controls the trade-off between maximising the margin and minimising classification errors [28, p. 504]. The hinge loss function is widely used for training classifiers to maximise the margin, penalising incorrect predictions and those that are correct but not sufficiently confident. Note that the notation $(\cdot)_+$ indicates that the positive part of the value is taken.

$$J(\mathbf{w}) = \min_{\mathbf{w}, w_0} \frac{1}{2} \|\mathbf{w}\|_2^2 + C \sum_{i=1}^N (1 - y_i f(\mathbf{x}_i))_+ \quad (6.2)$$

Through the use of Equation 6.2, the optimal weight vector can be determined. The optimal weight vector $\mathbf{w}$ can be expressed as a linear combination of the training points as in Equation 6.3 [28, p. 499]. The coefficients β_i are only nonzero for the support vectors, which makes the SVM more computationally efficient at inference time.

$$\mathbf{w} = \sum_{i=1}^N \beta_i \mathbf{x}_i \quad (6.3)$$

The linear decision function in Equation 6.1 does not work well if the data is not clearly linearly separable. Therefore, the SVM uses a kernel function which allows it to map data into a higher-dimensional feature space without directly computing the transformation. Multiple different kernel functions can be used, each with its own properties. Equation 6.4 shows the decision function of the SVM with the addition of the kernel function $\kappa(\mathbf{x}, \mathbf{x}_i)$, allowing for feature mapping [28, p. 492].

$$f(\mathbf{x}) = \sum_{i=1}^N \beta_i \kappa(\mathbf{x}, \mathbf{x}_i) + w_0 \quad (6.4)$$

Common kernel functions include the linear, polynomial, and radial basis function (RBF) kernels. We are particularly interested in the RBF kernel, which is defined in Equation 6.5 [28, p. 480]. In this equation, $\mathbf{x}'$ is the reference vector that we compare to the input vector $\mathbf{x}$. The variable σ^2 controls the width of the kernel and how strongly the similarity between two points decays with their distance. Small σ values produce narrow and highly localised kernels (leading to possible overfitting), while large σ values produce smoother decision boundaries.

$$\kappa(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{\|\mathbf{x} - \mathbf{x}'\|_2^2}{2\sigma^2}\right) \quad (6.5)$$

SVMs are typically designed for binary classification, but they can be used for multi-class classification through a one-vs-all setup. This setup involves training several SVMs, each doing binary classification by focusing on its target class and categorising all other data points into another class. When predicting a new sample, the sample is assigned to the class with the highest decision score.

6.2.3 Configuration

In the neural rejection system, a specific threshold θ must be used to reject samples. In [60], they select a specific percentage of the training signal samples to be rejected. We follow the same approach by choosing a percentile of the training samples to reject. The percentile that yields the best performance for the hybrid classifier is unknown, so tests are required to determine it. Our process in determining this percentile will be discussed in Section 6.4.

Our hybrid classifier comprises of the PET-CGDNN as the DL classifier and the KNN using higher-order cumulants as the NDL classifier. The KNN is selected because it performs well and is more stable across high SNRs than the QHLRT. The clustering classifier has the worst performance of the NDL classifiers and is therefore not considered. As input for the SVM of the hybrid classifier, we use the last feature layer of the PET-CGDNN, which is the GRU layer as illustrated in the network architecture (Figure 4.1). A one-vs-all SVM is used with the RBF kernel, as in [60].

Figure 6.2 shows the architecture of the proposed hybrid classifier, where the SVM controls the switching between the PET-CGDNN and the KNN. The SVM uses the output of the last feature layer of the PET-CGDNN, that is the GRU layer, as input. A pre-processor calculates the higher-order cumulants of the received signal $\mathbf{r}$, before providing the cumulants to the KNN classifier.

Our proposed implementation of the hybrid classifier differs from that of [60] in several aspects. We use the SVM as a switching mechanism rather than only rejecting adversarial

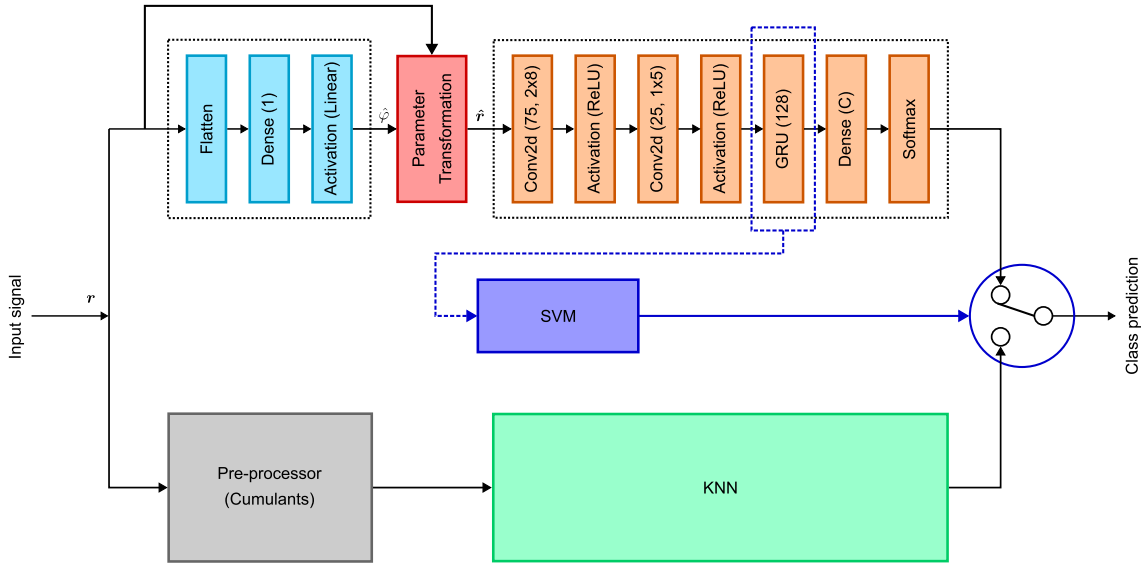


Figure 6.2: Hybrid classifier architecture.

signals, to improve the robustness of the DL classifier against attacks. In [60], the adversarial attack is modified to incorporate the SVM in the attack's loss function. We do not follow this approach and assume that the SVM is unknown to the adversarial attacks. In a real-world application of AMC, we believe that any DNN can be used and that it is not necessarily the PET-CGDNN that is being used. The SVM which is added to the DNN is seen as a retrofit or augmentation of the classifier to improve classification performance and robustness to attacks. Attackers would not be aware if a SVM is being used or not. It can be argued that this means our approach does not fully leverage white-box attacks, but we believe the white-box component applies only to the DNN, not to the SVM we add to it. In this case the hybrid classifier cannot be considered to be attacked with full white-box attacks.

6.3 Training process of SVM

As in the approach with the PET-CGDNN, we need to determine good parameters for the one-vs-all SVM classifier through hyperparameter tuning. Specifically, the C and γ parameters must be determined. As mentioned earlier in the discussion of SVMs, the C parameter controls the trade-off between maximising the margin of the SVM while minimising classification errors. The γ parameter is used in the RBF kernel as provided in Equation 6.5 and is defined as $\gamma = 1/2\sigma^2$. Different values for these parameters must be tested to determine a good version of the SVM.

Grid search with three-fold cross-validation is used to find the best parameters on the reduced training set, as described in Section 3.2. This is the same approach used by [60], where they use the neural rejection approach to reject adversarial signals. The reduced

Table 6.1: Hyperparameter tuning results for SVM of hybrid classifier on reduced training set.

Hyperparameter	Search space	Optimised value
C	0.1, 1, 10, 100	10
γ	scale, 0.001, 0.01, 0.1, 1	0.01

training set contains no adversarial perturbations. The SVM receives as input the output of the GRU(128) layer from the PET-CGDNN and the labels are the modulation types similar to the other AMC methods. The results of the hyperparameter tuning are provided in Table 6.1. Note that the value of "scale" for the γ parameter means that it was automatically calculated as $1/(\text{number of features} \times \text{variance of input})$.

After good parameters have been determined, the SVM is trained using the reduced training set using the selected parameters for C and γ . Multiple rejection thresholds are computed for the SVM using different percentiles of training samples rejected. The percentiles are selected to range from 10% to 90% in increments of 10%, where at low percentiles the DL classifier is more often used, and at high percentiles the NDL classifier is more often used. We use these different rejection thresholds to determine which rejection threshold to use for the hybrid classifier.

6.4 Tuning the SVM

With the SVM of the hybrid classifier trained, the different rejection thresholds from the different percentiles can be evaluated. The percentiles for the rejection threshold of the SVM are evaluated on the attacked validation set to determine if there is a good overall percentile. The validation set is used to test the percentiles to prevent overfitting on the test set, and thus overly optimistic results. Ideally, the hybrid classifier maintains the good performance of the PET-CGDNN on unperturbed signals while leveraging the robustness of the KNN on attacked signals.

The different percentiles for the rejection threshold of the SVM are tested with varying-size attacks using FGM and PGD on the validation set. Only the SNRs of -20, -10, 0, 10, and 20 dB are evaluated as these cover a wide range of SNRs. Choosing a good percentile for the SVM can be seen as "tuning" the hybrid classifier to use the PET-CGDNN or KNN more.

Figure 6.3 shows the performance of the hybrid classifier using different rejection thresholds on the FGM attacked signals, where the attacks have varying sizes using PNR. The SNRs of -10 and 0 dB are shown because they provide the greatest visual spread between the percentiles. The PET-CGDNN and KNN's classification performance on these attacked signals are also provided as a baseline. The curve of 10% percentile is not easily visible on the

graphs, as it is obscured by the 20% percentile curve. As can be seen from the plots, as the percentile increases the curves approach the KNN curve, because more samples are being classified by the KNN. Examining the plot for SNR = 0 dB, a good percentile that appears to balance performance and robustness on the FGM attacks is 30%, however we require a quantitative result for a thorough analysis.

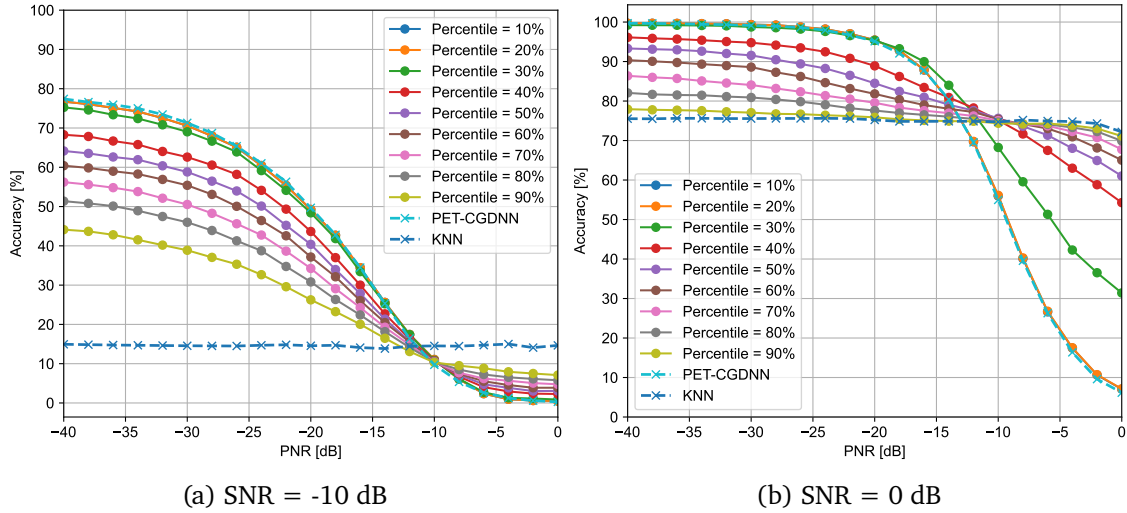


Figure 6.3: Classification accuracy using rejection thresholds determined using specific percentiles for the hybrid classifier on signals with FGM attacks.

Table 6.2: F1 scores across SNR levels and rejection threshold percentiles for FGM attacks on the validation set. The highest F1 score for each row is shown in bold.

SNR (dB)	Percentile								
	10	20	30	40	50	60	70	80	90
-20	0.0690	0.1004	0.1221	0.1221	0.1221	0.1221	0.1221	0.1221	0.1221
-10	0.4214	0.4214	0.4136	0.3791	0.3566	0.3377	0.3143	0.2904	0.2553
0	0.7498	0.7502	0.8163	0.8343	0.8235	0.8102	0.7912	0.7749	0.7564
10	0.8799	0.8801	0.9397	0.9763	0.9803	0.9798	0.9756	0.9714	0.9666
20	0.9906	0.9906	0.9997	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
Average	0.6221	0.6285	0.6583	0.6624	0.6565	0.6500	0.6406	0.6318	0.6200

F1 scores are computed to further analyse the results, where these scores describe the model's accuracy using precision and recall. These F1 scores on the FGM attacks are provided in Table 6.2. The percentiles that yielded the highest F1 score per SNR are shown in bold, and the average across all SNRs for each percentile is provided. From the table, it is evident that as the SNR changes, the percentile that results in the top performance also changes. At very low SNRs, the percentile is not as important as both the KNN and PET-CGDNN struggle. As the SNR increases, the percentile that results in the highest score also increases. Analysing the averaged results, we can see that the rejection threshold using the 40% percentile results in the highest score, with those from the 30% and 50% percentiles close.

The same process is repeated for the PGD attacks. Figure 6.4 shows the hybrid classifier with the different percentiles on PGD attacked signals in the validation set. Again, the 10% percentile curve is obscured by the 20% percentile curve on the graph. Compared to the FGM attacks at $\text{SNR} = 0$ dB, there is no clear percentile that appears to offer a good trade-off between classification performance and robustness against the attacks.

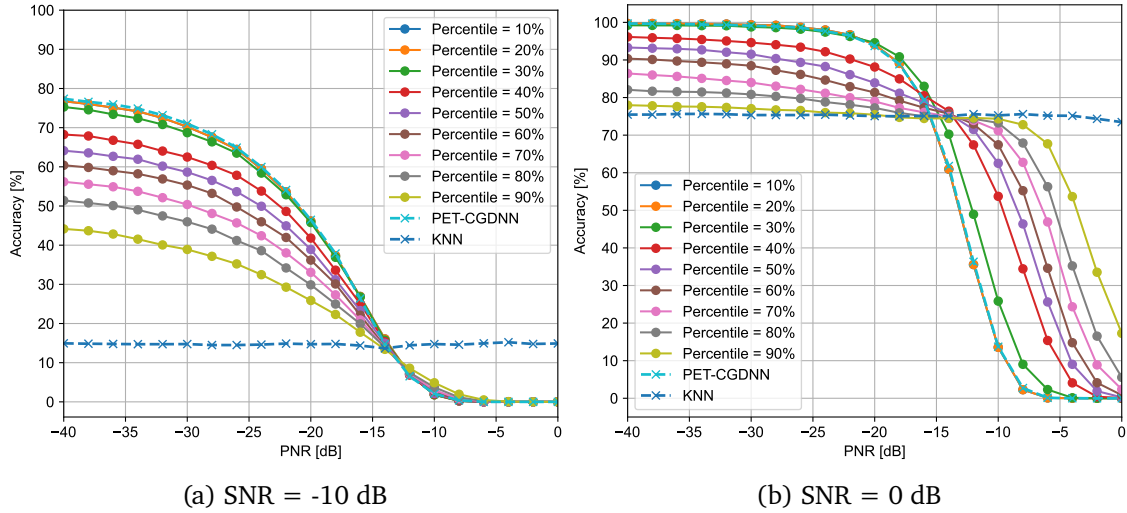


Figure 6.4: Classification accuracy using thresholds from specific percentiles for hybrid classifier on signals with PGD attacks.

Table 6.3: F1 scores across SNR levels and percentiles for PGD attacks on the validation set. The highest F1 score for each row is shown in bold.

SNR (dB)	Percentile								
	10	20	30	40	50	60	70	80	90
-20	0.0537	0.0816	0.1214	0.1226	0.1227	0.1227	0.1227	0.1227	0.1227
-10	0.3930	0.3930	0.3860	0.3527	0.3306	0.3112	0.2874	0.2638	0.2289
0	0.6499	0.6499	0.6714	0.6897	0.6887	0.6864	0.6836	0.6846	0.6969
10	0.7356	0.7356	0.7703	0.8405	0.8697	0.8901	0.9093	0.9253	0.9459
20	0.9142	0.9142	0.9660	0.9990	0.9999	1.0000	1.0000	1.0000	1.0000
Average	0.5493	0.5549	0.5830	0.6009	0.6023	0.6021	0.6006	0.5993	0.5989

F1 scores are again determined to quantitatively analyse the results on the PGD attacked signals. Table 6.3 presents the scores for the different percentiles on the PGD attacked signals as the SNR varies. In scenarios with high SNRs and PGD attacks, higher percentiles, such as 90% perform better, especially at the SNRs of 0, 10 and 20 dB. The PGD attacks are more sophisticated than the FGM attacks due to their multi-step nature, thus the hybrid classifier will overall perform better if it only occasionally uses the PET-CGDNN to identify the modulation type of the unknown signal. From the averaged F1 scores, it is evident that the 50% percentile has the highest score. The rejection thresholds determined using the 40% and 60% percentile also score highly.

With these results, it is clear that there is no overall good percentile that offers a good trade

off between classification performance and robustness for both the FGM and the PGD attacks. The rejection threshold determined using the 40% percentile scores the highest against the FGM attacks, whereas the rejection threshold determined using the 50% percentile scores the highest against the PGD attacks. Selecting one of these percentiles will bias the hybrid classifier in favour of one of the attack methods. It was thus decided to use the 30%, 40%, and 50% percentiles for the evaluation of the hybrid classifier. The 30% percentile is also included as it can be argued that, depending on the environment in which AMC occurs, adversarial attacks may be rare, and thus it is more important to achieve good performance with unperturbed signals.

With the rejection thresholds selected, the different configurations of the hybrid classifier can be evaluated against unperturbed signals, signals with fixed-size attacks, and signals with varying-size attacks.

6.5 Results

The hybrid classifier is evaluated on unperturbed signals, signals with fixed-size attacks, and signals with varying-size attacks to determine whether there is any improvement in robustness against the attacks. Different rejection thresholds are used for the hybrid classifier to evaluate different configurations of the hybrid classifier.

6.5.1 Performance on unperturbed signals

Figure 6.5 shows the performance of all the AMC methods in this study against unperturbed signals in the AWGN channel simulated in the dataset. Please note that the clustering classifier is evaluated on a reduced test set as discussed in Chapter 5, which is why its line style is different compared to the other classifiers in the plot. The important addition to this plot is the hybrid classifier, which comprises the PET-CGDNN and KNN. Different configurations of the hybrid classifier are shown using different percentiles that were used to determine the rejection threshold. The percentile used by each hybrid classifier configuration is shown in parentheses in the graph's legend. Against unperturbed signals, all configurations of the hybrid classifier perform similarly to the PET-CGDNN, with the 30% percentile configuration being the closest to the PET-CGDNN. This similarity is due to many samples still being classified by the PET-CGDNN at these different percentiles. As the percentile increases the curves of the different configurations of the hybrid classifier approach the curve of the KNN. Ideally for the hybrid classifier, the samples that are difficult to identify for the PET-CGDNN are being successfully identified by the KNN.

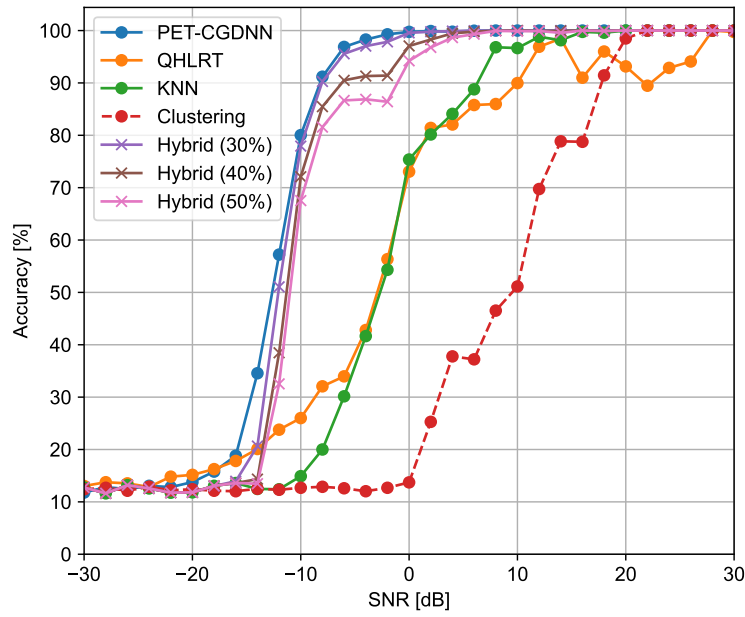


Figure 6.5: Classification performance of the different configurations of the hybrid classifier as SNR varies on unperturbed signals.

6.5.2 Performance on attacked signals

The different configurations of the hybrid classifier are tested on fixed-size and varying-size adversarial attacks on the modulated signals.

Fixed-size attacks

Figure 6.6 provides the performance of the hybrid classifiers on the fixed-size attacks of FGM and PGD. In addition, DeepFool attacks are also shown, but as previously discussed, it is not considered a fixed-size attack. For a baseline, the performance of the PET-CGDNN and KNN classifiers is also provided for easy comparison. Against the FGM attacks, all configurations of the hybrid classifiers show significant improvement in robustness against the attack, with the 50% percentile configuration showing the largest improvement. Against the PGD attacks, a smaller improvement is realised from a SNR of -4 dB and onwards. Even the 50% percentile configuration performs significantly worse than the KNN on the PGD attacks, but it outperforms the PET-CGDNN. It appears that the KNN is not selected often enough by the SVM of the hybrid classifiers when PGD attacks are present, or that it is also being fooled. As expected, the different configurations of the hybrid classifier struggle against the DeepFool attacks, as the PET-CGDNN and KNN also struggle against the DeepFool attacks.

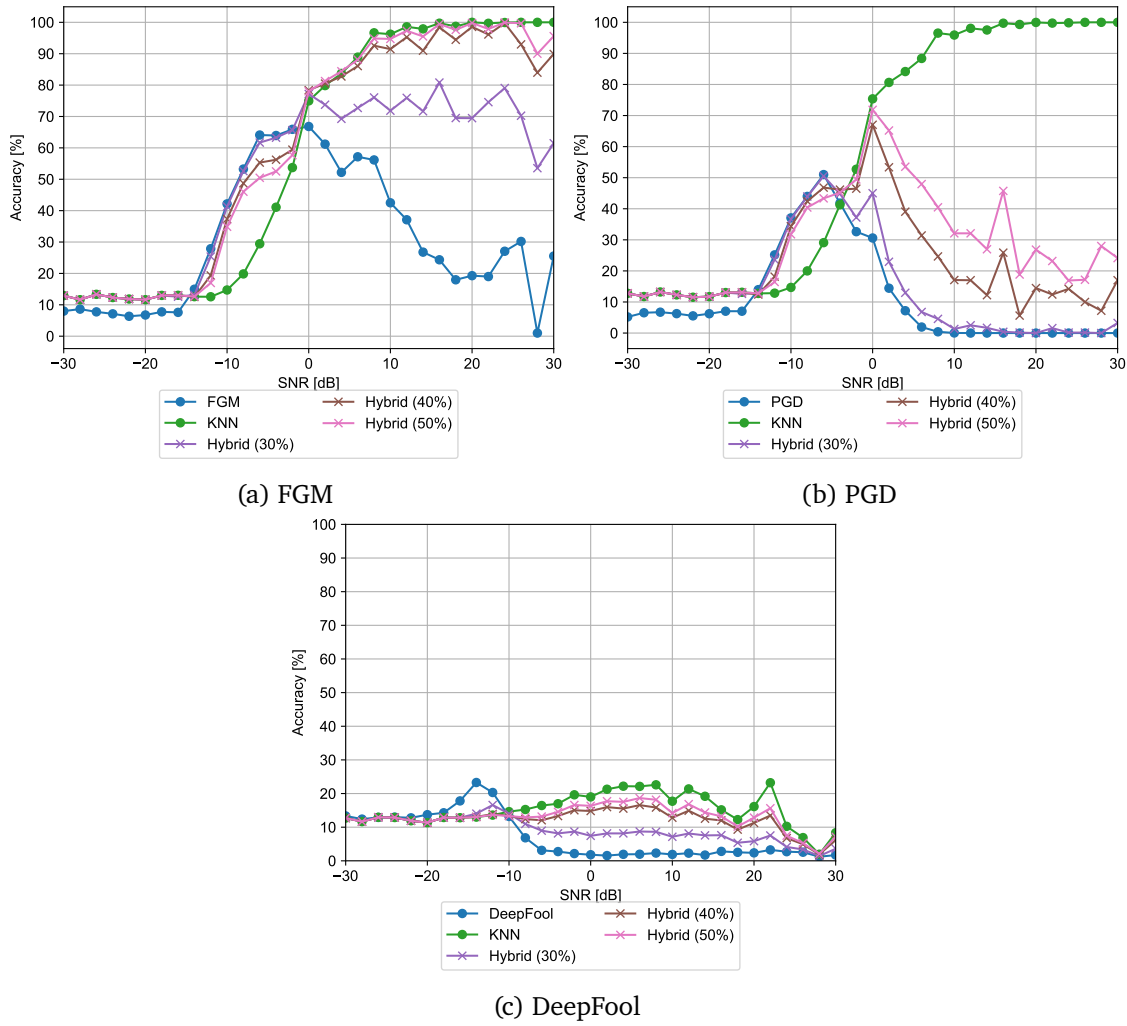


Figure 6.6: Classification performance of the different configurations of the hybrid classifier on fixed-size attacks as SNR varies.

Varying-size attacks

The classification performance of the hybrid classifiers against varying-size attacks using FGM and PGD as the PNR varies for specific SNRs is provided in Figure 6.7. One of the objectives of the hybrid classifier is to show improvement in robustness against the attacks, thus the PET-CGDNN is included in the plots for comparison. The KNN is excluded because its performance generally does not change with varying PNR. As seen with the fixed-size attacks, the different configurations of the hybrid classifier show the greatest increase in robustness against FGM attacks compared to PGD attacks. It can also be observed that when the SNR is -10 dB, the PET-CGDNN outperforms the hybrid classifiers, which is due to the KNN misclassifying some of the signals at this low SNR. The 30% percentile hybrid classifier has the highest accuracy among the hybrid classifiers when the SNR is -10 dB. As the PNR varies with the SNR at 0 dB, no variant of the hybrid classifier outperforms the other variants at all points. The best hybrid classifier variant changes as the PNR changes.

Starting with low PNRs, the 30% and 40% percentiles outperform the 50% percentile, but as the PNR approaches -14 dB and onwards, the 50% percentile outperforms the others. When the SNR is equal to 10 dB, the 50% percentile hybrid classifier outperforms the other hybrid classifier configurations at all PNRs. Overall, all of the hybrid classifiers balance classification performance with the PET-CGDNN and robustness with the KNN.

The previous result shows the classification performance for certain SNRs, but does not provide a complete understanding of the behaviour of the hybrid classifiers as the SNR and PNR vary. When tuning the SVM, we observed that the percentile that performs well changes with the SNR, and thus heatmaps can provide further clarity in this behaviour. We use heatmaps to visually show the improvements in the classification performance of the different configurations of the hybrid classifier as the SNR and PNR varies. The heatmaps show the difference in classification accuracy between the hybrid classifier and the PET-CGDNN, as follows:

$$\text{Heatmap accuracy (\%)} = \text{Hybrid classifier accuracy (\%)} - \text{PET-CGDNN accuracy (\%)} \quad (6.6)$$

The KNN is not considered as its performance is mostly constant over different PNRs.

The heatmaps for the different configurations of the hybrid classifier are provided in Figure 6.8. It is evident from the plots that as the percentile used to determine the rejection threshold of the hybrid classifier increases, the improvement in the classification accuracy compared to the PET-CGDNN increases. Interestingly, as the percentile increases, an area develops between the SNRs of -16 dB and 0 dB and at low PNRs where the hybrid classifier performs worse than the PET-CGDNN. We are unsure why this phenomenon occurs, and it is a possible area for future research. Furthermore, it is visible that the increase in classification performance for the hybrid classifiers is more spread over the SNRs and PNR against the FGM attacks, whereas against the PGD attacks it is restricted to a smaller area that changes as the PNRs and SNRs changes.

6.6 Chapter summary

This chapter presented a hybrid classifier combining DL and NDL techniques. The basis of the hybrid classifier was the neural rejection system, which is used to detect adversarial samples using a SVM on a pre-trained DNN. For the hybrid classifier, a SVM was used as a switching mechanism to decide whether to use the PET-CGDNN or the KNN using higher-order cumulants to classify an input signal. The PET-CGDNN was used on inputs classified by the SVM as non-adversarial, whereas the KNN was used on adversarial inputs. The training and hyperparameter tuning of the SVM was discussed. Furthermore, the tuning

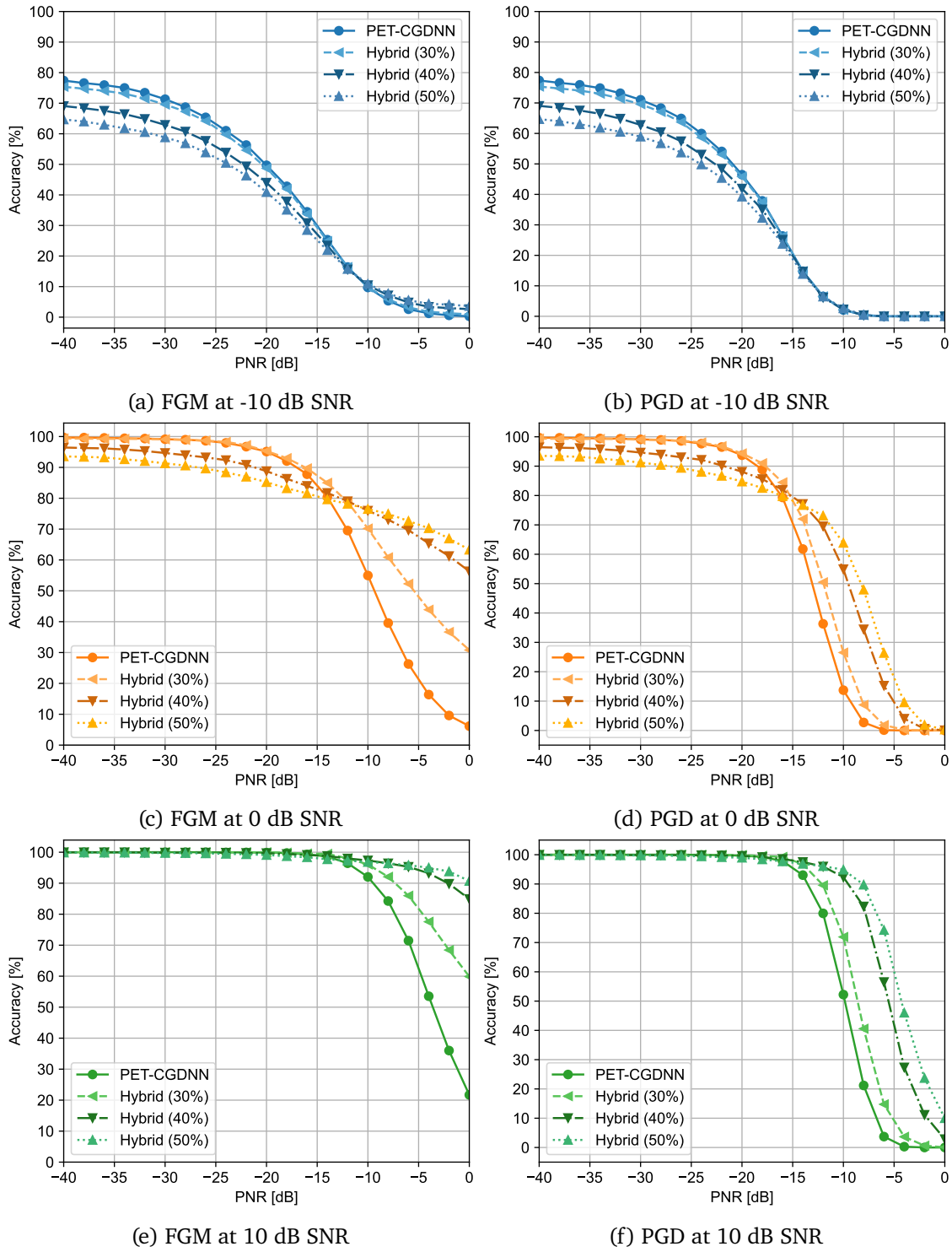


Figure 6.7: Classification performance of the different configurations of the hybrid classifier and the PET-CGDNN on signals attacked with varying-size attacks for specific SNRs as PNR varies.

of the hybrid classifier by choosing the correct rejection threshold for the SVM was discussed and the results of experiments done to determine this rejection threshold was provided.

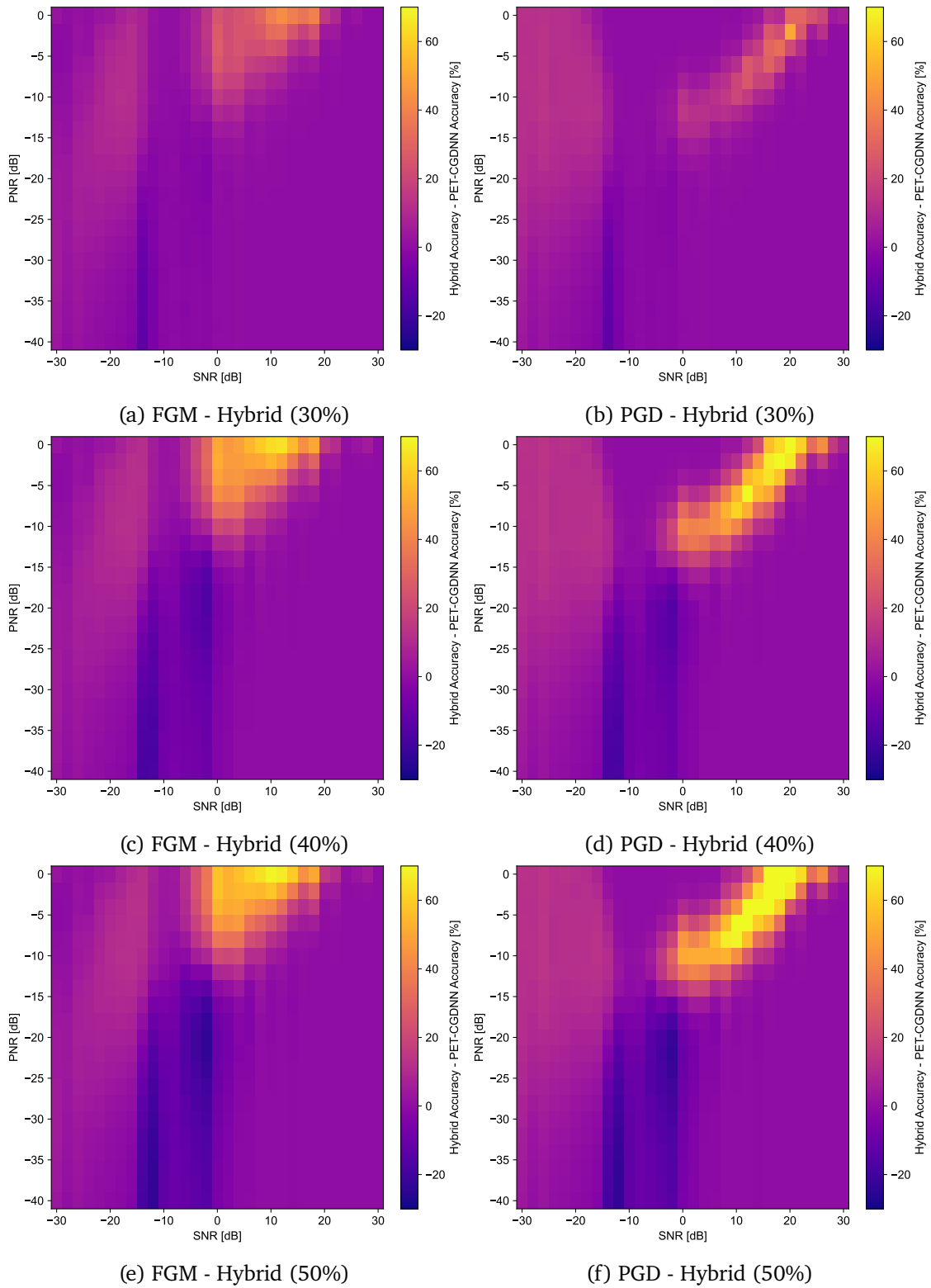


Figure 6.8: Difference between performance of different configurations of hybrid classifier and PET-CGDNN on signals attacked with varying-size attacks as SNRs and PNR varies.

Our findings indicated that there is no overall good rejection threshold determined using a

specific percentile that is not biased in favour of a specific attack method or scenario. We decided to evaluate different configurations of the hybrid classifier to examine their impact. We used the percentiles of 30%, 40%, and 50% for the rejection thresholds. The final hybrid classifier was evaluated on unperturbed signals and signals with fixed-size and varying-size attacks.

In our experiments, we found that the hybrid classifier improved robustness against adversarial attacks, especially against FGM attacks. The improvement against PGD attacks was less significant, but better than the performance the PET-CGDNN has on its own. When classifying unperturbed signals, the different configurations of hybrid classifier was able to achieve similar performance to the PET-CGDNN on its own, with the configuration using the rejection threshold from the 30% percentile outperforming the rest. Generally, against adversarial attacks, the rejection threshold from the 50% percentile outperformed the other hybrid classifier configurations. We conclude that the hybrid classifier was able to successfully achieve its objectives of keeping comparable performance to a DL classifier on unperturbed signals, while showing improvement in robustness against adversarial attacks.

Chapter 7

Conclusion

7.1 Introduction

We conclude the study by reflecting on the work done to address the research problem, questions, and objectives defined in the first chapter. The contributions of each chapter to the research objectives and questions is discussed. Possible future work for this domain is also provided.

7.2 Reflection on research objectives

AMC plays a crucial role in wireless communication systems and must be both reliable and precise. Previously, we discussed how the latest approach to AMC, namely DNNs, is highly vulnerable to adversarial attacks. We hypothesised that NDL methods for AMC do not have this vulnerability that the DNNs have. A research gap was identified in the literature that the robustness of AMC methods against attacks has not been fully investigated, specifically in the case of DL techniques versus NDL techniques for AMC. The transferability of adversarial attacks to NDL techniques requires more exploration.

In Chapter 1, we had set the following research objectives:

1. Select and recreate a DNN for AMC that is well-known and well-regarded. Select and recreate various NDL methods for AMC. Compare these AMC methods with and without adversarial attacks present and evaluate their robustness in classification accuracy in an AWGN channel.
2. Develop a new method that utilises both DNN and NDL techniques that demonstrates greater robustness against adversarial attacks and has comparable performance to

existing DNNs in AMC.

7.2.1 Research methodology

To complete the research objectives, we generated a dataset to test our selected AMC methods. Chapter 3 details the dataset, including signal frames, modulation types, AWGN signal channel, and the adversarial attacks applied to it. The white-box adversarial attacks consisted of the FGM, PGD and DeepFool methods. A varying-size and a fixed-size attack configuration was used for the FGM and PGD attacks to evaluate the AMC methods. The varying-size configuration enabled us to evaluate the effect of the PNR on the attack's effectiveness. The PNR was limited to a maximum of 0 dB, as we consider the adversarial perturbation to be imperceptible at this point and below. Constellation diagrams were provided for visual examination of the perturbations applied to the modulated signals.

We selected and evaluated the PET-CGDNN in Chapter 4 to complete our research objectives, in which we select a DL method and evaluate it. The PET-CGDNN model is well-known and well-regarded in the research field, as indicated by its high number of citations (the source paper has 158 citations at the time of writing). The architecture of the PET-CGDNN and its operations were discussed in detail. Furthermore, the implementation of the PET-CGDNN method was verified, and the hyperparameter tuning and training of the PET-CGDNN were provided. The PET-CGDNN was evaluated using both varying-size and fixed-size adversarial attacks from the dataset to assess its classification performance and the impact of these attacks on its performance.

To continue completing the research objectives, we selected and evaluated NDL methods for AMC in Chapter 5. The selected NDL classifiers consisted of a clustering technique, the QHLRT, and a KNN using higher-order cumulants. These classifiers covered a broad range of NDL approaches to AMC to assess the robustness of different NDL approaches. The theory behind each classifier was discussed. Additionally, the verification of the QHLRT and KNN was performed by comparing them with the source papers. We evaluated the NDL AMC methods both amongst themselves and against the PET-CGDNN using unperturbed and attacked signals from the generated dataset.

In Chapter 6, we developed a hybrid classifier in line with our research objectives. The hybrid classifier combines a DL classifier and a NDL classifier to retain the good classification performance of the DL method, while benefiting from the robustness against attacks of the NDL method. We discussed the design of the hybrid classifier by explaining and motivating the neural rejection system, where a SVM is used to reject adversarial samples. The theory behind SVMs was provided to explain their inner workings. Our neural rejection system setup was provided, where we use the SVM as a switching mechanism between a DL classifier and a NDL classifier. We chose the PET-CGDNN and KNN as the AMC methods

comprising the hybrid classifier, because the DNN is effective on unperturbed signals, while the KNN both performs well and exhibits more stable performance than the QHLRT. We experimented to determine the best rejection threshold for the SVM by selecting a specific percentile of unperturbed signals to reject by the SVM. It was found that no percentile performed the best across all scenarios, that is, with different SNRs, PNRs, and attack methods. We selected the 30%, 40%, and 50% percentiles to create different configurations of the hybrid classifier, each biased towards different situations. The 30% percentile hybrid classifier is biased to favour performance against unperturbed signals, whereas the 40% percentile hybrid classifier is biased to perform well against FGM attacks, and the 50% percentile is biased to perform well against PGD attacks.

7.3 Reflection on research questions

The research objectives were introduced to address the following research questions, as identified in Chapter 1:

1. Are adversarial attacks transferable to NDL methods for AMC and to what extent are they transferable?
2. Can NDL techniques be used in conjunction with DL methods to create a new AMC approach that has improved robustness against adversarial attacks?

In our study, we used white-box adversarial attacks generated by the trained PET-CGDNN, in both fixed-size and varying-size configurations. Due to the white-box nature of the attacks, they operated directly on the features that the DNN relies on for classification. It was found that the PET-CGDNN performs well on unperturbed signals, but its performance is significantly deteriorated when these generated adversarial perturbations are added. The DeepFool attack degrades the DNN's performance to a point where it is no longer practically effective. For the attacks with varying-sizes, the PET-CGDNN is mainly unaffected at low PNRs, but as PNR increases, the classification performance worsens. The PET-CGDNN showed limited robustness against the attacks. In contrast, the NDL classifiers, that is the QHLRT, a clustering classifier, and a KNN using higher-order cumulants, showed greater robustness against the adversarial perturbations, as their classification performance was mostly unaffected. The exception is the DeepFool attacks, which were strong enough to render the NDL methods ineffective, as the DeepFool attacks had done against the PET-CGDNN.

From the experiments conducted, it was observed that the FGM and PGD attacks have limited transferability to the NDL methods. The DeepFool attack was strong enough to disrupt classification attempts by all NDL classifiers. With the varying-size attacks, the

experiments were conducted only up to 0 dB PNR and it is highly possible that beyond this point, the FGM and PGD attacks have a greater effect on the NDL classifiers. The end point of 0 dB PNR was selected as at this point the adversarial perturbation has the same power level as the noise, and if the PNR exceeds 0 dB, then the attack can no longer be considered imperceptible. From the results, it can be concluded that the transferability of an adversarial attack from the DNN to the NDL methods depends on the attack method and its strength. The features that the DNN uses for classification differ from the features used by the NDL methods, which supports the motivation for a hybrid classifier that combines the two approaches.

We combined a DNN and a NDL method to form a hybrid classifier to address the last research question. We had found that there is no single best rejection threshold for the hybrid classifier, thus we evaluated three distinct configurations that used the 30%, 40%, and 50% percentiles for the SVM to determine rejection threshold. After evaluating the hybrid classifier configurations against unperturbed and attacked signals, it was found that all hybrid classifiers had similar performance to the PET-CGDNN against unperturbed signals. The hybrid classifier that used the 30% percentile outperforms the other configurations against the unperturbed signals. All hybrid classifier configurations were more robust than the PET-CGDNN against attacked signals, with the 50% percentile hybrid classifier outperforming the others as the PNR increased. The increased robustness against the FGM attack was more pronounced for all hybrid classifiers than against the PGD attack. Any increase in robustness against the DeepFool attack was marginal at best. The results obtained with the hybrid classifier supports the argument that it is possible to use NDL techniques with DL techniques to benefit from both approaches. Our hybrid classifier achieved greater robustness against adversarial attacks through NDL techniques while mostly retaining the classification performance of the DNN.

7.4 Future work

The study has several avenues for future work that can be explored:

- How performant are the AMC methods in more complex signal channels, such as a Rayleigh fading channel, and how is their robustness to adversarial attacks affected by these signal channels?
- How do adversarial defence techniques, such as adversarial training, compare to our hybrid classifier?
- Is it possible to improve the hybrid classifier by employing the deep neural rejection system, which uses multiple network layers, rather than the neural rejection system, which only uses the last feature layer?

- If the adversarial attacks and AMC methods are implemented in the real-world, do we still see the same observations in robustness and classification performance?
- How robust is the hybrid classifier when the loss function of the PET-CGDNN and the SVM is combined when generating adversarial attacks on the classifier?

7.5 In closing

In conclusion, AMC is an important function in wireless communication systems, where reliability and good classification performance are the primary goals. Recent research has focused on using DNNs for AMC due to their good performance. The DL classifiers for AMC outperform the NDL classifiers on signals without adversarial attacks present, as supported by the results obtained in this study. However, the DL classifiers are vulnerable to adversarial attacks, such as FGM, PGD, and DeepFool, whereas the NDL classifiers do not show the same degree of vulnerability. We tested the NDL classifiers of the QHLRT, a clustering technique, and a KNN using higher-order cumulants against the DL classifier of PET-CGDNN. The adversarial attacks of FGM, PGD and DeepFool were used to evaluate the robustness of the classifiers with a varying-size attack and fixed-size attack. Our results show that the NDL classifiers are more robust to adversarial attacks. We developed a hybrid classifier that combines the PET-CGDNN and KNN to achieve classification performance comparable to that of the PET-CGDNN when not under attack, while benefiting from the KNN's robustness against attacks.

Bibliography

- [1] Ridhima and A. Singh Buttar, “Fundamental Operations of Cognitive Radio: A Survey,” in *2019 IEEE International Conference on Electrical, Computer and Communication Technologies (ICECCT)*, Feb. 2019, pp. 1–5.
- [2] J. J. Popoola and R. van Olst, “The performance evaluation of a spectrum sensing implementation using an automatic modulation classification detection method with a Universal Software Radio Peripheral,” *Expert Systems with Applications*, vol. 40, no. 6, pp. 2165–2173, May 2013.
- [3] M. Rida Bahloul, M. Z. Yusoff, A.-H. Abdel-Aty, M. N. M. Saad, and L. Kudsi, “Spectrum Sensing for Cognitive Radio Systems Using Modulation Classification: A Review,” *Applied Mathematics & Information Sciences*, vol. 11, no. 3, pp. 857–865, May 2017.
- [4] F. Zhang, C. Luo, J. Xu, Y. Luo, and F.-C. Zheng, “Deep learning based automatic modulation recognition: Models, datasets, and challenges,” *Digital Signal Processing*, vol. 129, p. 103650, Sep. 2022.
- [5] Z. Zhu and A. K. Nandi, *Automatic Modulation Classification: Principles, Algorithms and Applications*. Wiley, 2015.
- [6] T. J. O’Shea, J. Corgan, and T. C. Clancy, “Convolutional Radio Modulation Recognition Networks,” in *Engineering Applications of Neural Networks*. Springer International Publishing, 2016, pp. 213–226.
- [7] S. Rajendran, W. Meert, D. Giustiniano, V. Lenders, and S. Pollin, “Deep Learning Models for Wireless Signal Classification With Distributed Low-Cost Spectrum Sensors,” *IEEE Transactions on Cognitive Communications and Networking*, vol. 4, no. 3, pp. 433–445, Sep. 2018.
- [8] Y. Wang, T. Sun, S. Li, X. Yuan, W. Ni, E. Hossain, and H. Vincent Poor, “Adversarial Attacks and Defenses in Machine Learning-Empowered Communication Systems and Networks: A Contemporary Survey,” *IEEE Communications Surveys & Tutorials*, vol. 25, no. 4, pp. 2245–2298, 2023.

-
- [9] W. Zhang, M. Krunz, and G. Ditzler, “Stealthy Adversarial Attacks on Machine Learning-Based Classifiers of Wireless Signals,” *IEEE Transactions on Machine Learning in Communications and Networking*, vol. 2, pp. 261–279, 2024.
- [10] Y. Lin, H. Zhao, Y. Tu, S. Mao, and Z. Dou, “Threats of Adversarial Attacks in DNN-Based Modulation Recognition,” in *IEEE INFOCOM 2020 - IEEE Conference on Computer Communications*, Jul. 2020, pp. 2469–2478.
- [11] 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, 2020.
- [12] D. Adesina, C.-C. Hsieh, Y. E. Sagduyu, and L. Qian, “Adversarial Machine Learning in Wireless Communications Using RF Data: A Review,” *IEEE Communications Surveys & Tutorials*, vol. 25, no. 1, pp. 77–100, 2023.
- [13] B. R. Manoj, P. M. Santos, M. Sadeghi, and E. G. Larsson, “Toward Robust Networks against Adversarial Attacks for Radio Signal Modulation Classification,” in *2022 IEEE 23rd International Workshop on Signal Processing Advances in Wireless Communication (SPAWC)*, Jul. 2022, pp. 1–5.
- [14] M. Usama, M. Asim, J. Qadir, A. Al-Fuqaha, and M. A. Imran, “Adversarial Machine Learning Attack on Modulation Classification,” in *2019 UK/ China Emerging Technologies (UCET)*, Aug. 2019, pp. 1–4.
- [15] T. Huynh-The, C.-H. Hua, Q.-V. Pham, and D.-S. Kim, “MCNet: An Efficient CNN Architecture for Robust Automatic Modulation Classification,” *IEEE Communications Letters*, vol. 24, no. 4, pp. 811–815, Apr. 2020.
- [16] A. P. Hermawan, R. R. Ginanjar, D.-S. Kim, and J.-M. Lee, “CNN-Based Automatic Modulation Classification for Beyond 5G Communications,” *IEEE Communications Letters*, vol. 24, no. 5, pp. 1038–1041, May 2020.
- [17] A. van der Merwe and A. Helberg, “Modulation classification accuracy of a clustering and a CNN technique under FGSM adversarial attacks,” in *Southern Africa Telecommunication Networks and Applications Conference (SATNAC)*, 2024, pp. 197–203.
- [18] —, “Hybrid Automatic Modulation Classification for Increased Robustness under White-Box Adversarial Attacks,” in *Southern African Conference for Artificial Intelligence Research*, 2025, pp. 39–55.
- [19] —, “Comparison of robustness of modulation classifiers against white-box adversarial attacks,” in *Southern Africa Telecommunication Networks and Applications Conference (SATNAC)*, 2025, pp. 140–149.
-

-
- [20] O. A. Dobre and F. Hameed, "Likelihood-Based Algorithms for Linear Digital Modulation Classification in Fading Channels," in *2006 Canadian Conference on Electrical and Computer Engineering*, May 2006, pp. 1347–1350.
- [21] J. Proakis and M. Salehi, *Fundamentals of Communication Systems*, 2nd ed. Boston: Pearson, Jun. 2013.
- [22] O. Dobre, A. Abdi, Y. Bar-Ness, and W. Su, "A Survey of Automatic Modulation Classification Techniques: Classical Approaches and New Trends," *IET Commun.*, vol. 1, no. 2, pp. 137–156, Apr. 2007.
- [23] P. Ghasemzadeh, S. Banerjee, M. Hempel, and H. Sharif, "Performance Evaluation of Feature-based Automatic Modulation Classification," in *2018 12th International Conference on Signal Processing and Communication Systems (ICSPCS)*, Dec. 2018, pp. 1–5.
- [24] U. Satija, M. Mohanty, and Barathram. Ramkumar, "Automatic modulation classification using S-transform based features," in *2015 2nd International Conference on Signal Processing and Integrated Networks (SPIN)*, Feb. 2015, pp. 708–712.
- [25] M. J. Evans and J. S. Rosenthal, *Probability and Statistics: The Science of Uncertainty*, 2nd ed. New York: W. H. Freeman, 2009.
- [26] D. H. Al-Nuaimi, I. A. Hashim, I. S. Zainal Abidin, L. B. Salman, and N. A. Mat Isa, "Performance of Feature-Based Techniques for Automatic Digital Modulation Recognition and Classification—A Review," *Electronics*, vol. 8, no. 12, p. 1407, Dec. 2019.
- [27] M. L. D. Wong, S. K. Ting, and A. K. Nandi, "Naïve Bayes classification of adaptive broadband wireless modulation schemes with higher order cumulants," in *2008 2nd International Conference on Signal Processing and Communication Systems*, Dec. 2008, pp. 1–5.
- [28] K. P. Murphy, *Machine Learning: A Probabilistic Perspective*. The MIT Press, Jul. 2012.
- [29] D. Xu and Y. Tian, "A Comprehensive Survey of Clustering Algorithms," *Ann. Data. Sci.*, vol. 2, no. 2, pp. 165–193, Jun. 2015.
- [30] J. P. Mouton, M. Ferreira, and A. S. J. Helberg, "A comparison of clustering algorithms for automatic modulation classification," *Expert Systems with Applications*, vol. 151, p. 113317, Aug. 2020.
- [31] N. Jafar, A. Paeiz, and A. Farzaneh, "Automatic modulation classification using modulation fingerprint extraction," *Journal of Systems Engineering and Electronics*, vol. 32, no. 4, pp. 799–810, Aug. 2021.
- [32] D. Zhang, Y. Ye, D. Liu, M. Zhang, J. Tang, K. He, and S. Wang, "Modulation Recognition and Nonlinearity Resistance of RoF System Based on Clustering Algorithm," in *2018 23rd Opto-Electronics and Communications Conference (OECC)*, Jul. 2018, pp. 1–2.
-

-
- [33] V. Sathyanarayanan, P. Gerstoft, and A. E. Gamal, "RML22: Realistic Dataset Generation for Wireless Modulation Classification," *IEEE Transactions on Wireless Communications*, vol. 22, no. 11, pp. 7663–7675, Nov. 2023.
- [34] 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. 2016.
- [35] T. J. O'Shea, T. Roy, and T. C. Clancy, "Over-the-Air Deep Learning Based Radio Signal Classification," *IEEE Journal of Selected Topics in Signal Processing*, vol. 12, no. 1, pp. 168–179, Feb. 2018.
- [36] K. Tekbıyık, A. R. Ekti, A. Görçin, G. K. Kurt, and C. Keçeci, "Robust and Fast Automatic Modulation Classification with CNN under Multipath Fading Channels," in *2020 IEEE 91st Vehicular Technology Conference (VTC2020-Spring)*, May 2020, pp. 1–6.
- [37] T. Huynh-The, Q.-V. Pham, T.-V. Nguyen, T. T. Nguyen, R. Ruby, M. Zeng, and D.-S. Kim, "Automatic Modulation Classification: A Deep Architecture Survey," *IEEE Access*, vol. 9, pp. 142 950–142 971, 2021.
- [38] Q. Zheng, P. Zhao, H. Wang, A. Elhanashi, and S. Saponara, "Fine-Grained Modulation Classification Using Multi-Scale Radio Transformer With Dual-Channel Representation," *IEEE Communications Letters*, vol. 26, no. 6, pp. 1298–1302, Jun. 2022.
- [39] J. Cai, F. Gan, X. Cao, and W. Liu, "Signal Modulation Classification Based on the Transformer Network," *IEEE Transactions on Cognitive Communications and Networking*, vol. 8, no. 3, pp. 1348–1357, Sep. 2022.
- [40] S. J. D. Prince, *Understanding Deep Learning*. The MIT Press, 2023.
- [41] F. Meng, P. Chen, L. Wu, and X. Wang, "Automatic Modulation Classification: A Deep Learning Enabled Approach," *IEEE Transactions on Vehicular Technology*, vol. 67, no. 11, pp. 10 760–10 772, Nov. 2018.
- [42] N. E. West and T. O'Shea, "Deep architectures for modulation recognition," in *2017 IEEE International Symposium on Dynamic Spectrum Access Networks (DySPAN)*, Mar. 2017, pp. 1–6.
- [43] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. ukasz Kaiser, and I. Polosukhin, "Attention is All you Need," in *Advances in Neural Information Processing Systems*, vol. 30. Curran Associates, Inc., 2017.
- [44] T. Lin, Y. Wang, X. Liu, and X. Qiu, "A survey of transformers," *AI Open*, vol. 3, pp. 111–132, Jan. 2022.
- [45] Z. Zhang, H. Luo, C. Wang, C. Gan, and Y. Xiang, "Automatic Modulation Classification Using CNN-LSTM Based Dual-Stream Structure," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 11, pp. 13 521–13 531, Nov. 2020.
-

-
- [46] S. Chang, S. Huang, R. Zhang, Z. Feng, and L. Liu, "Multitask-Learning-Based Deep Neural Network for Automatic Modulation Classification," *IEEE Internet of Things Journal*, vol. 9, no. 3, pp. 2192–2206, Feb. 2022.
- [47] J. Xu, C. Luo, G. Parr, and Y. Luo, "A Spatiotemporal Multi-Channel Learning Framework for Automatic Modulation Recognition," *IEEE Wireless Communications Letters*, vol. 9, no. 10, pp. 1629–1632, Oct. 2020.
- [48] F. Zhang, C. Luo, J. Xu, and Y. Luo, "An Efficient Deep Learning Model for Automatic Modulation Recognition Based on Parameter Estimation and Transformation," *IEEE Communications Letters*, vol. 25, no. 10, pp. 3287–3290, Oct. 2021.
- [49] Y. Xu, X. Zhong, A. Jimeno Yepes, and J. H. Lau, "Grey-box Adversarial Attack And Defence For Sentiment Classification," *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 4078–4087, 2021.
- [50] M. Sadeghi and E. G. Larsson, "Adversarial Attacks on Deep-Learning Based Radio Signal Classification," *IEEE Wireless Communications Letters*, vol. 8, no. 1, pp. 213–216, Feb. 2019.
- [51] B. Kim, Y. E. Sagduyu, K. Davaslioglu, T. Erpek, and S. Ulukus, "Over-the-Air Adversarial Attacks on Deep Learning Based Modulation Classifier over Wireless Channels," in *2020 54th Annual Conference on Information Sciences and Systems (CISS)*, Mar. 2020, pp. 1–6.
- [52] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and Harnessing Adversarial Examples," Mar. 2015.
- [53] Y. Dong, F. Liao, T. Pang, H. Su, J. Zhu, X. Hu, and J. Li, "Boosting Adversarial Attacks with Momentum," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Jun. 2018, pp. 9185–9193.
- [54] N. Carlini and D. Wagner, "Towards Evaluating the Robustness of Neural Networks," in *2017 IEEE Symposium on Security and Privacy (SP)*. IEEE Computer Society, May 2017, pp. 39–57.
- [55] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards Deep Learning Models Resistant to Adversarial Attacks," in *2018 International Conference on Learning Representations*, Feb. 2018.
- [56] S.-M. Moosavi-Dezfooli, A. Fawzi, O. Fawzi, and P. Frossard, "Universal Adversarial Perturbations," *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 86–94, Jul. 2017.

-
- [57] S.-M. Moosavi-Dezfooli, A. Fawzi, and P. Frossard, “DeepFool: A Simple and Accurate Method to Fool Deep Neural Networks,” *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2574–2582, Jun. 2016.
- [58] The MathWorks Inc., *MATLAB version: 23.2.0 (R2023b)*, Natick, Massachusetts, United States, 2023. [Online]. Available: <https://www.mathworks.com>
- [59] M.-I. Nicolae, M. Sinn, M. N. Tran, B. Buesser, A. Rawat, M. Wistuba, V. Zantedeschi, N. Baracaldo, B. Chen, H. Ludwig, I. M. Molloy, and B. Edwards, “Adversarial Robustness Toolbox v1.2.0,” Nov. 2019.
- [60] L. Zhang, S. Lambbotharan, G. Zheng, B. AsSadhan, and F. Roli, “Countermeasures Against Adversarial Examples in Radio Signal Classification,” *IEEE Wireless Communications Letters*, vol. 10, no. 8, pp. 1830–1834, Aug. 2021.
- [61] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama, “Optuna: A Next-Generation Hyperparameter Optimization Framework,” in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2019.
- [62] J. MacQueen, “Some methods for classification and analysis of multivariate observations,” in *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Statistics*. University of California Press, Jan. 1967, vol. 5.1, pp. 281–298.
- [63] D. L. Davies and D. W. Bouldin, “A Cluster Separation Measure,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-1, no. 2, pp. 224–227, Apr. 1979.
- [64] F. Hameed, O. A. Dobre, and D. C. Popescu, “On the likelihood-based approach to modulation classification,” *IEEE Transactions on Wireless Communications*, vol. 8, no. 12, pp. 5884–5892, Dec. 2009.
- [65] M. Venkata Subbarao and P. Samundiswary, “Performance Analysis of Modulation Recognition in Multipath Fading Channels using Pattern Recognition Classifiers,” *Wireless Pers Commun*, vol. 115, no. 1, pp. 129–151, Nov. 2020.
- [66] B. Dgani and I. Cohen, “Efficient Cumulant-Based Automatic Modulation Classification Using Machine Learning,” *Sensors*, vol. 24, no. 2, p. 701, Jan. 2024.
- [67] P. Panagiotou, A. Anastasopoulos, and A. Polydoros, “Likelihood ratio tests for modulation classification,” in *MILCOM 2000 Proceedings. 21st Century Military Communications. Architectures and Technologies for Information Superiority (Cat. No.00CH37155)*, vol. 2, Oct. 2000, pp. 670–674 vol.2.
- [68] A. Sotgiu, A. Demontis, M. Melis, B. Biggio, G. Fumera, X. Feng, and F. Roli, “Deep neural rejection against adversarial examples,” *EURASIP Journal on Information Security*, vol. 2020, no. 1, p. 5, Apr. 2020.
-

-
- [69] M. Melis, A. Demontis, B. Biggio, G. Brown, G. Fumera, and F. Roli, “Is Deep Learning Safe for Robot Vision? Adversarial Examples Against the iCub Humanoid,” in *2017 IEEE International Conference on Computer Vision Workshops (ICCVW)*, Oct. 2017, pp. 751–759.

Appendix A

Supporting figures for investigation of NDL techniques

In this appendix, supporting figures for the analysis of the classification performance of the NDL classifiers is included. These figures were excluded from Chapter 5 for the sake of brevity.

3D surface plots are created and provided in Figure A.1, showing the robustness of the NDL classifiers to the varying-size attacks of FGM and PGD. The same observations can be made from these graphs as was made in Chapter 5, where the NDL classifiers are largely unaffected by the varying-size attacks.

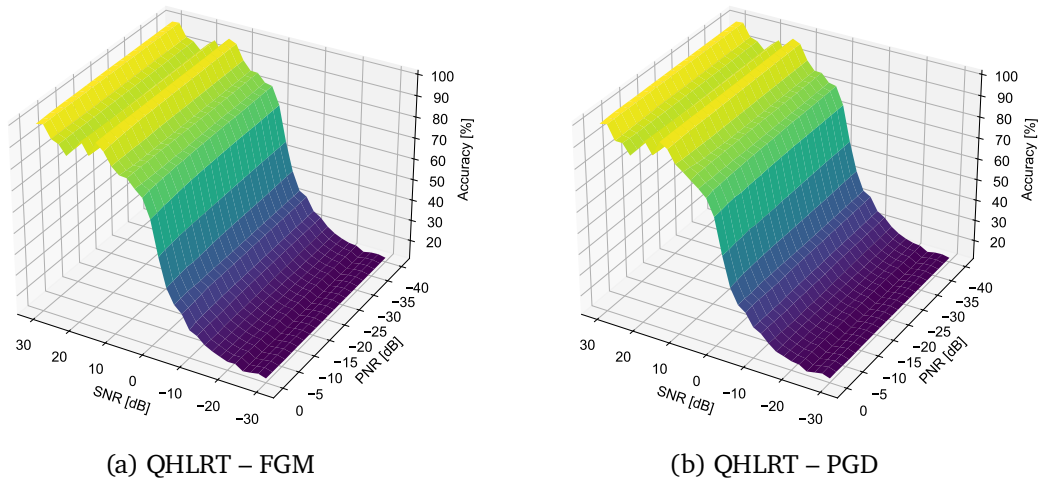
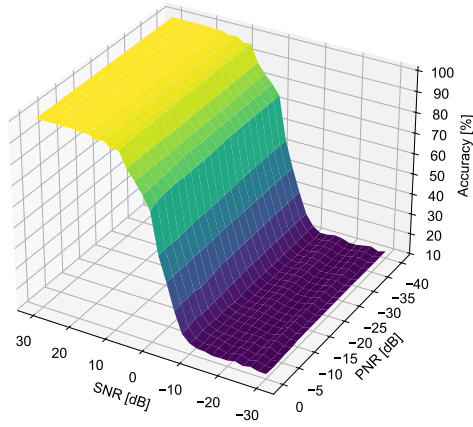
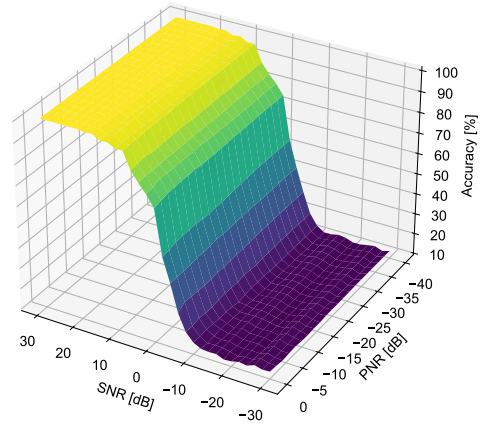


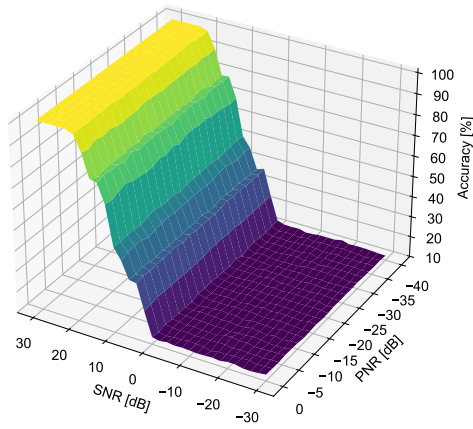
Figure A.1: 3D surface of the classification performance of NDL classifiers on varying-size attacks as SNR and PNR varies.



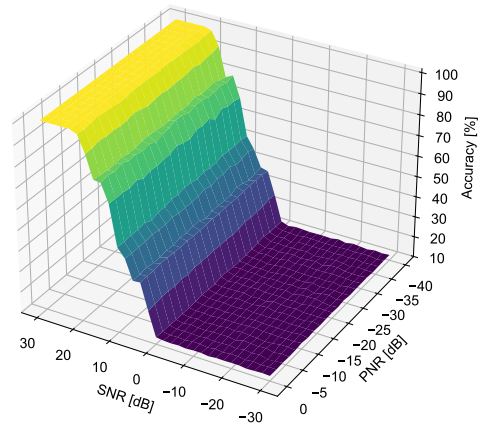
(c) KNN – FGM



(d) KNN – PGD



(e) Clustering – FGM



(f) Clustering – PGD

Figure A.1: 3D surface of the classification performance of NDL classifiers on varying-size attacks as SNR and PNR varies (Continued).