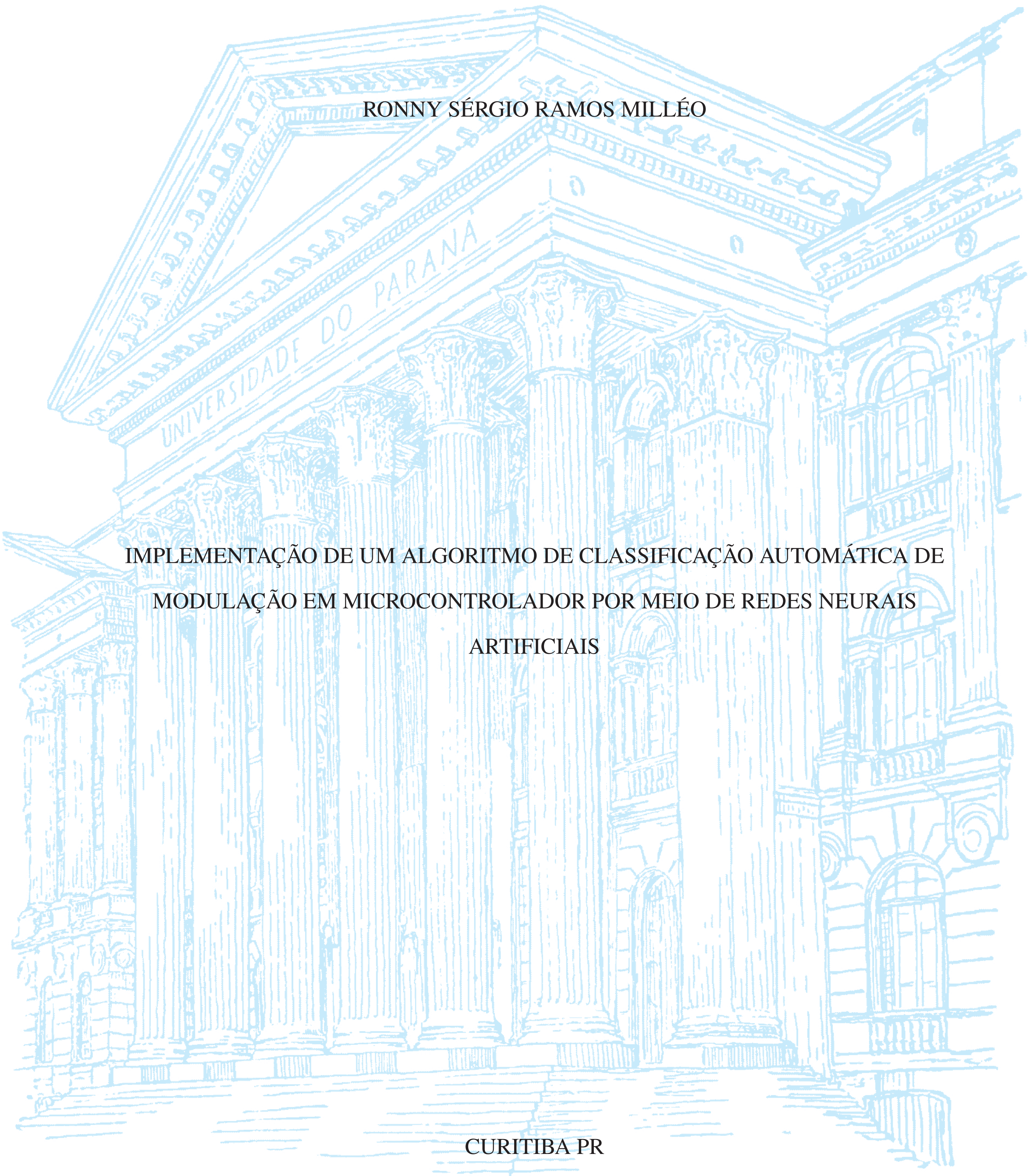
UNIVERSIDADE FEDERAL DO PARANÁ



RONNY SÉRGIO RAMOS MILLÉO

IMPLEMENTAÇÃO DE UM ALGORITMO DE CLASSIFICAÇÃO AUTOMÁTICA DE MODULAÇÃO EM MICROCONTROLADOR POR MEIO DE REDES NEURAIS ARTIFICIAIS

CURITIBA PR

2021

RONNY SÉRGIO RAMOS MILLÉO

IMPLEMENTAÇÃO DE UM ALGORITMO DE CLASSIFICAÇÃO AUTOMÁTICA DE MODULAÇÃO EM MICROCONTROLADOR POR MEIO DE REDES NEURAIS ARTIFICIAIS

Dissertação apresentada como requisito parcial à obtenção do grau de Mestre em Engenharia Elétrica no Programa de Pós-Graduação em Engenharia Elétrica, Setor de Tecnologia, da Universidade Federal do Paraná.

Área de concentração: *Engenharia Elétrica*.

Orientador: André Augusto Mariano.

Coorientador: Luis Henrique Assumpção Lolis.

CURITIBA PR

2021

Catalogação na Fonte: Sistema de Bibliotecas, UFPR
Biblioteca de Ciência e Tecnologia

M646i Milléo, Ronny Sérgio Ramos
Implementação de um algoritmo de classificação automática de modulação em microcontrolador por meio de redes neurais artificiais [recurso eletrônico] / Ronny Sérgio Ramos Milléo – Curitiba, 2021.

Dissertação - Universidade Federal do Paraná, Setor de Tecnologia, Programa de Pós-graduação em Engenharia Elétrica.

Orientador: André Augusto Mariano
Coorientador: Luis Henrique Assumpção Lolis.

1. Redes Neurais Artificiais. 2. Sistemas Embarcados (Computação). 3. Modulação (Eletrônica). I. Universidade Federal do Paraná. II. Mariano, André Augusto III. Lolis, Luis Henrique Assumpção. IV. Título.

CDD: 006.4

Bibliotecária: Roseny Rivelini Morciani CRB-9/1585

TERMO DE APROVAÇÃO

Os membros da Banca Examinadora designada pelo Colegiado do Programa de Pós-Graduação em ENGENHARIA ELÉTRICA da Universidade Federal do Paraná foram convocados para realizar a arguição da Dissertação de Mestrado de **RONNY SÉRGIO RAMOS MILLÉO** intitulada: **IMPLEMENTAÇÃO DE UM ALGORITMO DE CLASSIFICAÇÃO AUTOMÁTICA DE MODULAÇÃO EM MICROCONTROLADOR POR MEIO DE REDES NEURAIS ARTIFICIAIS**, sob orientação do Prof. Dr. ANDRÉ AUGUSTO MARIANO, que após terem inquirido o aluno e realizada a avaliação do trabalho, são de parecer pela sua APROVAÇÃO no rito de defesa.

A outorga do título de mestre está sujeita à homologação pelo colegiado, ao atendimento de todas as indicações e correções solicitadas pela banca e ao pleno atendimento das demandas regimentais do Programa de Pós-Graduação.

CURITIBA, 26 de Março de 2021.

Assinatura Eletrônica
27/03/2021 08:26:38.0
ANDRÉ AUGUSTO MARIANO
Presidente da Banca Examinadora

Assinatura Eletrônica
29/03/2021 09:56:27.0
LEANDRO DOS SANTOS COELHO
Avaliador Interno (UNIVERSIDADE FEDERAL DO PARANÁ)

Assinatura Eletrônica
26/03/2021 17:32:36.0
BERNARDO REGO BARROS DE ALMEIDA LEITE
Avaliador Interno (UNIVERSIDADE FEDERAL DO PARANÁ)

Assinatura Eletrônica
27/03/2021 08:14:09.0
WAGNER ENDO
Avaliador Externo (UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ)

Dep.Eng.Elétrica-DELT, Centro Politecnico - UFPR - CURITIBA - Paraná - Brasil
CEP 81531990 - Tel: 41 3361-3622 - E-mail: ppgee@eletrica.ufpr.br
Documento assinado eletronicamente de acordo com o disposto na legislação federal Decreto 8539 de 08 de outubro de 2015.
Gerado e autenticado pelo SIGA-UFPR, com a seguinte identificação única: 85368
Para autenticar este documento/assinatura, acesse https://www.prppg.ufpr.br/siga/visitante/autenticacaoassinaturas.jsp e insira o codigo 85368

To my mother, for being the strong foundation that held me in place through the years. To my family and friends who have always believed that we can achieve many things through dedication and commitment.

ACKNOWLEDGEMENTS

To André Mariano, thanks for all the patience, motivational speeches, and time spent on regular meetings pushing me forward to finish this project.

To Luis Lolis, thanks for all the complicated explanations about telecommunications signals and systems.

Thanks to my colleagues and friends at the research group for all the fun we had working and drinking coffee.

Thanks to all the electrical engineering department's staff for the quick answers and problems solved.

Thanks to CAPES (Coordenação de Aperfeiçoamento de Pessoal de Nível Superior) for the scholarship received during two years of development.

Thanks to Andressa Lopes and ST Microelectronics Brasil for the donation of a development board on which I was able to pursue my research.

RESUMO

A utilização de algoritmos de inteligência artificial tem crescido nos últimos anos com a maior disponibilidade de poder computacional a baixo custo. Uma demanda por dispositivos que usem comunicação sem fio também cresce e esses dispositivos podem ter diversas aplicações em inúmeros ambientes diferentes, portanto, é importante que o seu *hardware* possa adaptar-se às condições do ambiente ao redor para que a comunicação seja otimizada. A adaptação de conexão pode ser feita alterando-se a modulação em tempo real dependendo das características do canal. Como essa mudança é feita no transmissor, o receptor precisa conseguir identificar a modulação que está sendo usada no sinal recebido para que sua demodulação seja feita, ou seja, uma classificação automática de modulação é necessária no receptor para que o sistema funcione. As soluções mais populares encontradas no estado da arte para a classificação automática de modulação são o uso de redes neurais artificiais, que são técnicas de inteligência artificial. Porém, em sua maioria elas são análises computacionais do problema buscando uma acurácia maior e sistemas robustos que tentam classificar uma grande quantidade de modulações diferentes. A proposta desta dissertação é usar uma rede neural artificial com três camadas ocultas para classificar modulações digitais em fase e amplitude e ruído branco gaussiano para que a rede possa identificar se há um sinal chegando ou não, implementá-la em um sistema embarcado e comparar seu desempenho antes e depois da implementação. Para que a classificação seja possível, a rede neural artificial recebe como entrada uma série de características extraídas matematicamente desses sinais modulados, o que é chamado no estado da arte de classificação baseada em características. Como o objetivo principal da dissertação é a implementação em sistema embarcado, a quantidade de características de sinais utilizadas para a classificação deve ser reduzida para economizar recursos e a rede neural artificial é quantizada em ponto fixo para extrair maior desempenho do *hardware*. Uma varredura de parâmetros é feita com o objetivo de melhorar o desempenho da rede neural artificial em sua acurácia, como por exemplo variar o tamanho das suas camadas e o número de neurônios. Os testes finais são feitos usando sinais modulados com relação sinal-ruído variando de -10 dB a 20 dB, seis características de sinais e uma rede neural artificial com 3 camadas ocultas, com 26, 29 e 30 neurônios respectivamente. Obteve-se como resultado uma rede neural capaz de classificar com acurácia próxima a 100% as modulações PSK (*Phase Shift Keying*) para relação sinal ruído acima de 8 dB, enquanto as modulações QAM (*Quadrature Amplitude Modulation*) apresentaram um resultado pior, com acurácia variando demais sem uma estabilidade. No sistema embarcado os resultados das modulações PSK se repetiram, 64QAM piorou e 16QAM melhorou. Com isso, observa-se que é possível implementar um sistema de classificação automática de modulação em um sistema embarcado com poucas características sendo usadas na rede neural artificial.

Palavras-chave: Classificação Automática de Modulação. Sistemas Embarcados. Redes Neurais Artificiais.

ABSTRACT

The use of artificial intelligence algorithms has grown in recent years with the increased availability of low-cost computing power. Demand for devices that use wireless communication is also increasing. These devices can have many applications in numerous different environments, so self-adaptation is important to the surrounding environment's conditions so that the communication is optimized. Link adaptation can be made by changing the modulation in real-time depending on the channel characteristics. Since this change is made at the transmitter, the receiver needs to identify the modulation used in the received signal to demodulate it, i.e., an automatic modulation classification is required at the receiver for the system to work. The most popular solutions found in state-of-the-art automatic modulation classification are artificial neural networks, artificial intelligence techniques. However, most of them are computational analyses of the problem seeking higher accuracy and robust systems that try to classify many different modulations. This dissertation proposes to use an artificial neural network with three hidden layers to classify digital modulations in phase and amplitude and a Gaussian white noise so that the network can identify if there is an incoming signal or not. Implement it in an embedded system and compare its performance before and after implementation. For classification to be possible, the artificial neural network receives input a series of features extracted mathematically from these modulated signals called feature-based classification in state of the art. Since the dissertation's primary goal is implementation on an embedded system, the number of signal features used for classification must be reduced to save resources. Also, the artificial neural network is fixed-point quantized to extract more performance from the hardware. A sweep of parameters is done to improve the accuracy performance of the artificial neural network, varying the size of its layers and the number of neurons. The final tests are done using modulated signals with signal-to-noise ratios ranging from -10 dB to 20 dB, six signal characteristics, and a 3-layer artificial neural network with 26, 29, and 30 neurons. The result was a neural network able to classify with accuracy close to 100% the PSK (Phase Shift Keying) modulations for a signal-to-noise ratio above 8 dB. In contrast, the QAM (Quadrature Amplitude Modulation) modulations presented a worse result, with accuracy varying too much without stability. The PSK modulations' results were repeated in the embedded system, 64QAM worsened, and 16QAM improved. With this, it is observed that it is possible to implement an automatic modulation classification system in an embedded system with few features in the artificial neural network.

Keywords: Automatic Modulation Classification. Embedded Systems. Artificial Neural Networks.

LIST OF FIGURES

LIST OF TABLES

LIST OF ACRONYMS

AMC	Automatic Modulation Classification
ANN	Artificial Neural Network
API	Application Programming Interface
ARM	Advanced RISC Machine
AWGN	Additive White Gaussian Noise
BPSK	Binary Phase Shift Keying
CMSIS-NN	Cortex Microcontroller Software Interface Standard - Neural Networks
DFT	Discrete Fourier Transform
DSP	Digital Signal Processor
FFT	Fast Fourier Transform
FPGA	Field Programmable Gate Array
GPU	Graphic Processing Unit
IEEE	Institute of Electrical and Electronics Engineers
MATLAB	Matrix Laboratory
PC	Personal Computer
PSK	Phase Shift Keying
RISC	Reduced Instruction Set Computer
QAM	Quadrature Amplitude Modulation
QPSK	Quadrature Phase Shift Keying
ReLU	Rectified Linear Unit
RISC	Reduced Instruction Set Computer
RMS	Root Mean Square
RMSprop	Root Mean Square Propagation
SNR	Signal-to-Noise Ratio

LIST OF SYMBOLS

arctan	arctangent
$\arg z(t)$	argument of z(t)
$\in$	belongs to
f_c	carrier frequency
C_{pq}	complex cumulant (statistics)
M_{pq}	complex moment (statistics)
$y*$	conjugate of y
dB	decibels
dBW	decibel watts
e	euler's constant
γ	gamma
$\Im$	imaginary numbers set
j	imaginary number
μ	mu, mean
N_0	noise spectral density
π	pi constant
ϕ	phi
$\Re$	real numbers set
$\Re^2$	pairs of real numbers set
σ	sigma, standard deviation
θ	theta
σ^2	variance

CONTENTS

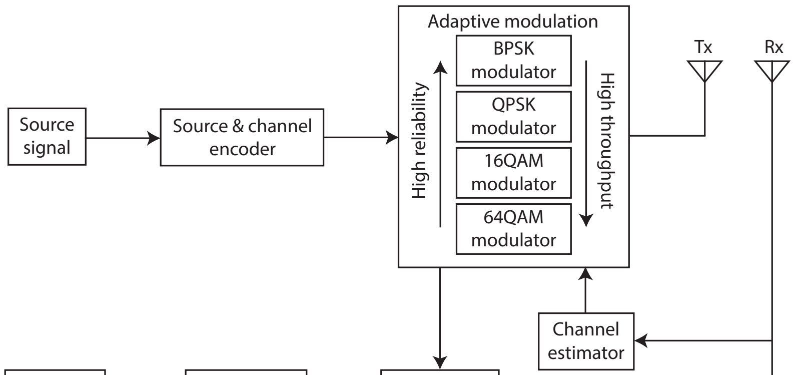
1 INTRODUCTION

The increasing computational power linked to microprocessors' low manufacturing costs has allowed the rapid development of solutions to various engineering problems, such as automation of processes, using algorithms of artificial intelligence. Communication between devices using wireless networks, called the Internet of Things, offers the possibility of data collection and control in both industrial environments and inside people's homes.

Combining the performance, low cost, and low power consumption found in microprocessors in recent years, the use of artificial intelligence algorithms to solve problems, and the internet of things, it is possible to develop simple and practical solutions to automate more and more of the surrounding environment. In contrast, the use of devices connected to the Internet of Things via wireless networks requires optimal connection performance since the environments in which these devices can be connected vary significantly, given their application.

The system needs to change the connection in real-time to offer the best connection in a dynamic environment, aiming to balance data transmission and connection robustness. Link adaptation is how a wireless communication system measures its environment and automatically changes, for example, the modulation used in transmission and reception to achieve the best connection. The link adaptation is made in the transmitter, so the receiver needs to identify the modulation used in the received signal in order for demodulation to occur, i.e., for the information to be correctly interpreted in the receiver. For the whole process to be fast, an automatic modulation classification algorithm is needed at the receiver for the whole system to work. As demonstrated in Figure 1.1, the modulation used is changed following a channel estimator that selects the best modulation for the environment from high reliability to high throughput.

Figure 1.1: AMC in link adaptation system



(NANDI, 2015)

The most popular solutions found in state-of-the-art automatic modulation classification are artificial neural networks and deep learning, which are artificial intelligence techniques. However, most of them are computational analyses of the problem seeking a higher accuracy and robust systems that try to classify as many modulations as possible as a benchmark for the proposed solution. The Automatic Modulation Classification (AMC) algorithms can use two different methods: likelihood-based method or feature-based method. This project will focus only on feature-based methods because it is the most simple to be implemented on embedded systems and can deliver satisfactory results. The features extracted from the receiver's signals are often statistical measures based on the instantaneous amplitude, phase, or frequency of these signals [1].

The importance of investigating the behavior and performance of a fully embedded automatic modulation classification algorithm based on artificial neural networks is valid because almost all scientific studies on the AMC subject are made with only the artificial neural network's accuracy performance in mind. In a few implementations of this algorithm, high-performance hardware is used, such as Field Programmable Gate Arrays (FPGA) and Graphic Processing Units (GPU) [2] [3], so implementing an AMC algorithm on an embedded system opens the possibilities for developing products using low-cost hardware and low power consumption.

Hence, the limitations of a low-cost embedded system are not yet studied. Sometimes, the solution for the AMC problem proposed in the state-of-the-art is also different from the proposal of this work, for example, some papers are using convolutional neural networks and images of the modulations constellations instead of the classical feature-based method discussed in the literature review [4] [5].

1.1 AIM

This project aims to implement a feature-based classification algorithm using fully-connected deep artificial neural networks in an embedded system and compare the Personal Computer (PC) and the embedded system's accuracy results. The PC results are obtained by the training and test of the neural network using some available tools in Python, like Keras and Tensorflow. The embedded system result is the data collected from the same neural network trained in the PC but properly deployed and embedded into a defined microprocessor.

1.2 OBJECTIVES

In order to achieve the desired aim of this project, the following objectives are being set:

- Select a small group of modulations and generate a large dataset of modulated signals with a wide range of signal-to-noise ratio;
- Find the most relevant features used to classify modulated signals and study their behavior;
- Implement most of the features found in the literature, and measure hardware usage;
- Use the data obtained from the implementation of the features selected, and reduce the necessary number of features to classify the selected modulations whenever it is possible;
- Try to obtain an optimal neural network for the reduced number of features using an available hyperparameter sweeping tool;
- Compare the embedded system's classification accuracy to the training accuracy found in the PC.

The following document is divided into four chapters, where the literature needed to understand the development of this project is described in chapter two. Chapter three summarizes the overall methodology used to generate modulated signals, study the features used by the feature-based algorithm, train the artificial neural networks, and communicate with the embedded system. Chapter four comprehends both results and discussions, and finally, chapter five gives the conclusions and open space for future work.

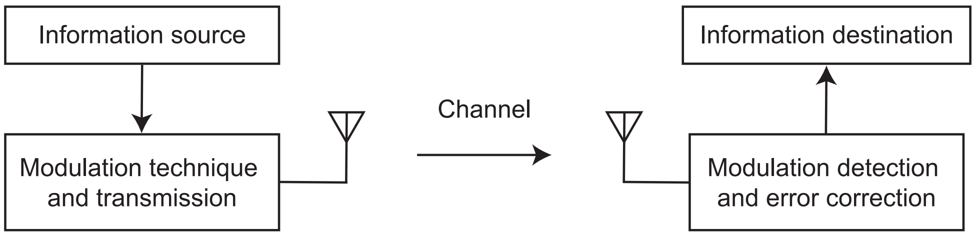
2 LITERATURE REVIEW

In this chapter, an introduction to the most important aspects of the theory behind this dissertation will be presented, starting with telecommunications systems, then some review on artificial neural networks, and finishing with embedded systems properties.

2.1 DIGITAL WIRELESS MODULATIONS

The transmission of wireless signals using digital modulation is widely used over analog modulations because it offers a high data rate, powerful error correction techniques, better security, and resistance to channel impairments. The transmitter needs to map the digital information into an analog signal. This technique transforms bits into analog symbols that are transmitted over the channel [6].

Figure 2.1: Basic wireless communication


Information source
Modulation technique and transmission
Channel
Information destination
Modulation detection and error correction

(THE AUTHOR, 2020)

Some considerations need to be made regarding the choice of a modulation. The first is the amount of data sent in a symbol, i.e., how many bits of information are converted to a single analog signal. The more bits are placed in a single symbol, the greater its spectral efficiency, measured in bits/s/Hz. A good example is to compare a modulation that uses a 100 kHz band to transmit 100 kbits/s and has an efficiency of 1 bit/s/Hz with another modulation that can send 200 kbits/s using the same 100 kHz band and therefore has an efficiency of 2 bit/s/Hz [6]. It is essential that the modulation has robustness against noise and other factors that may interfere with communication. That is, a robust modulation has a low error rate. Last but not least is the complexity and cost of its implementation [6].

The information can be modulated using one of two main categories of digital modulations: amplitude and phase modulation or frequency modulation. This work will address only the first category because amplitude and phase modulations have better spectral properties than frequency modulation [6].

2.1.1 Amplitude and phase modulation

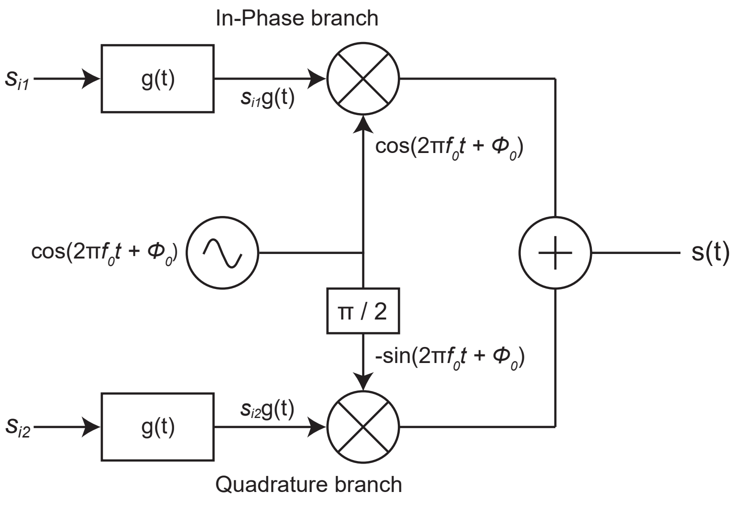
In amplitude and phase modulation, the binary information is encoded into amplitude or phase, sometimes in both. Over a time interval of length T_s (called symbol time), K bits are encoded, and K is defined to be $\log_2(M)$, where M is the modulation number. The PSK (Phase Shift Keying) technique uses only phase modulation, and QAM (Quadrature Amplitude Modulation) uses amplitude and phase modulations. Some PSK modulations are BPSK (Binary Phase Shift Keying) and QPSK (Quadrature Phase Shift Keying), and some QAM examples are 16QAM and 64QAM [6]. Also, amplitude and phase modulations are used in communication protocols such as WiFi and communication technologies for mobile telephony, such as the current fourth-generation (4G) and fifth-generation (5G) in the implementation state [7] [8].

The transmitted signal can be defined as an in-phase and quadrature components (Figure 2.2) where $s(t) = s_I(t)\cos(2\pi f_c t) - s_Q(t)\sin(2\pi f_c t)$ with a signal space representation written as $s(t) = s_{i1}(t)\phi_1(t) - s_{i2}(t)\phi_2(t)$, where the $\phi(t)$ functions are $\phi_1(t) = g(t)\cos(2\pi f_c t + \phi_0)$ and $\phi_2(t) = -g(t)\cos(2\pi f_c t + \phi_0)$, s_I is the In-Phase component, s_Q is the Quadrature component, f_c is the carrier frequency, ϕ_0 is an initial phase and g(t) is a shaping pulse that is designed to improve spectral efficiency. The signal constellation is defined based on the constellation point $s_i = (s_{i1}, s_{i2}) \in \Re^2, i = (1, 2, ..., M)$ and the complex representation of $s(t)$ is shown at the Equation 2.1 [6]:

$$s(t) = \Re\{x(t)e^{j\phi_0}e^{j(2\pi f_c t)}\} \tag{2.1}$$

where $x(t) = s_I(t) + js_Q(t) = (s_{i1}(t) + js_{i2}(t))g(t)$. The bit rate for this kind of modulation is calculated by K bits per symbol or $R = \log_2(M/T_s)$ bits per second [6].

Figure 2.2: Amplitude and phase modulator



(GOLDSMITH, 2005)

2.1.2 Phase Shift Keying

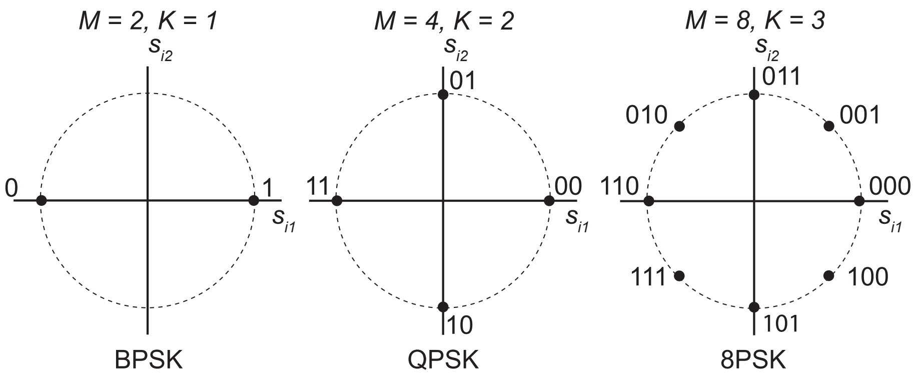
Since all information is coded in the phase of the transmitted signal, the signal over one symbol can be written as [6]:

$$\begin{aligned} s_i(t) &= \Re\{Ag(t)e^{j2\pi(i-1)/M}e^{j2\pi f_c t}\}, \qquad 0 \le t \le T_s \\ &= Ag(t)\cos\left[2\pi f_c t + \frac{2\pi(i-1)}{M}\right] \\ &= Ag(t)\cos\left[\frac{2\pi(i-1)}{M}\right]\cos(2\pi f_c t) - Ag(t)\sin\left[\frac{2\pi(i-1)}{M}\right]\sin(2\pi f_c t) \end{aligned} \tag{2.2}$$

The minimum distance between two constellation points is $d_{min} = 2A\sin(\pi/M)$ and A is a function of the signal energy. The PSK symbols are $s_{i1} = A\cos(2\pi(i-1)/M)$ and $s_{i2} = A\sin(2\pi(i-1)/M)$ for $i = (1, 2, ..., M)$. All possible transmitted symbols have the same energy and it is defined by the Equation 2.3. The figure 2.3 shows how the symbols are positioned in a geometric representation of the PSK signals [6]:

$$E_{s_i} = \int_0^{T_s} s_i^2(t)dt = A^2 \tag{2.3}$$

Figure 2.3: PSK constellations examples



(THE AUTHOR, 2020)

Figure 2.3 illustrates how the constellation mapping is done using Gray encoding to ensure that adjacent symbols only differ by one bit. Mistaking a symbol for an adjacent one causes no more than a one-bit error.

2.1.3 Quadratude Amplitude Modulation

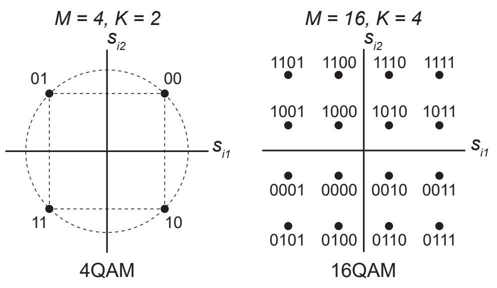
To encode the information in amplitude and phase, the transmitted signal is given by [6]:

$$\begin{aligned} s_i(t) &= \Re\{A_i e^{j\theta_i} g(t) e^{j2\pi f_c t}\}, \qquad 0 \le t \le T_s \\ &= A_i \cos(\theta_i) g(t) \cos(2\pi f_c t) - A_i \sin(\theta_i) g(t) \sin(2\pi f_c t) \end{aligned} \tag{2.4}$$

The distance between two symbols in the signal constellation for QAM is calculated by the Equation 2.5. The most common QAM techniques use square constellations, as shown in Figure 2.4. It can be hard to find a good mapping for irregular constellations where adjacent symbols only differ by one bit.

$$d_{ij} = \|s_i - s_j\| = \sqrt{(s_{i1} - s_{j1})^2 + (s_{i2} - s_{j2})^2} \tag{2.5}$$

Figure 2.4: QAM constellations examples



(THE AUTHOR, 2020)

The energy for QAM symbols is:

$$E_{s_i} = \int_0^{T_s} s_i^2(t)dt = A_i^2 \tag{2.6}$$

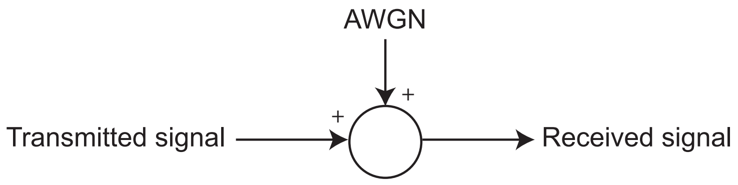
2.1.4 Channel modeling

In a communication system, the channel can be modeled differently to simulate various behaviors encountered in nature. This work will focus only on Additive White Gaussian Noise (AWGN, details in Figure 2.5), which is is one of the simplest mathematical models used in radio communication systems [9], so it is easy to simulate in a controlled environment. The sources of noise that can be found in a communication system may be external, for example, atmospheric noise or internal to the system like thermal noise [10].

The Gaussian distribution (Equation 2.7) that generates the white Gaussian noise can be described by the Equation 2.7, where f_X is the probability density function of x, μ is the mean and σ^2 is the variance.

$$f_X(x) = \frac{1}{\sqrt{2\pi}\sigma} \exp \frac{-(x-\mu)^2}{2\sigma^2} \tag{2.7}$$

Figure 2.5: AWGN channel model



(THE AUTHOR, 2020)

The term white denotes processes in which all frequency components have the same power [9], i.e., the white noise's power spectral density is equal for all frequencies and is described in the Equation 2.8. Considering that the signal transmitted over the channel has a limited bandwidth and the noise power is uniform inside the same bandwidth, the Signal-to-Noise Ratio (SNR) is written in the Equation 2.9, where P_S is the signal power [10]:

$$S_W(f) = \frac{N_0}{2} = \sigma_W^2 \tag{2.8}$$

$$SNR = \frac{P_S}{N_0/2} \tag{2.9}$$

2.2 DIGITAL MODULATION FEATURES

The feature-based modulation classification method is a method that uses information extracted from signals through mathematical, often statistical, calculations. It can be said that a feature is a result obtained by a calculation made with samples of a given modulated signal. This calculation can have variations when there is noise in the modulated signal, so it is repeated for several SNR values. The variation of the feature's value through a range of SNR is often plotted with SNR as the x-axis to understand the feature's behavior in different noise environments [1].

The information needed to classify modulations comes from many different features extracted from the signal spectrum, the instantaneous time values, or high-order statistics [1]. Each feature can be optimal for differencing a couple of modulations but not satisfactory for differencing many other modulations, so the combination of features is often used to correctly classify a group of desired modulations [11] [12] [13].

2.2.1 Obtaining the instantaneous values

The received modulated signal is sampled at a frequency fs and separated into frames containing N samples each. Most of the features selected for this work use the instantaneous amplitude, phase, or frequency of the unknown signal. The instantaneous amplitude is the amplitude measured from every sample from the received signal. The instantaneous phase is the angle between the complex-valued sample's imaginary and real amplitudes (respectively, the received Quadrature and In-Phase signals). Finally, the instantaneous frequency is defined by [14]:

$$f_i(t) = \frac{1}{2\pi}\frac{d}{dt}[\arg z(t)] \tag{2.10}$$

where $z(t)$ is the complex-valued signal and $\arg z[n]$ is the unwrapped phase of the received signal, and the derivative on the discrete-time domain is the difference between consecutive samples. So the Equation 2.10 can be rewritten as Equation 2.11. Phase unwrapping is the process by which phase differences between samples accumulate over time, forming a new array whose minimum or maximum phase is not defined between $-\pi$ and π.

$$f_i[n] = \frac{1}{2\pi} \mathit{diff}\left[\mathit{unwrap}\left(\arctan\left(\frac{\Im\{z[n]\}}{\Re\{z[n]\}} \right) \right) \right], \quad k = (1, 2, ..., N) \tag{2.11}$$

2.2.2 Spectrum-based features

The first feature is the maximum value of the spectral power density of the normalized and centered instantaneous amplitude of the received signal (γ_{max}), proposed by Azzouz and Nandi (1996a), Equation 2.12, and it is also called Gmax:

$$\gamma_{max} = \frac{\max |DFT(A_{cn}[n])|^2}{N} \tag{2.12}$$

where DFT is the Discrete Fourier Transform, A_{cn} is the normalized and centered instantaneous amplitude, N is the frame size, i.e., number of samples and $n = (1, 2, ..., N)$. The normalization needed to extract this feature is achieved by using the Equations 2.14 and 2.13, where μ_A is the mean of the instantaneous amplitude of one signal frame and $A[n]$ is the instantaneous amplitude of a sample. The normalization operation is designed to compensate the unknown channel attenuation [13].

$$A_n[n] = \frac{A[n]}{\mu_A} \tag{2.13}$$

$$A_{cn}[n] = A_n[n] - 1 \tag{2.14}$$

The two following features are the standard deviation of the absolute value of the non-linear component of the instantaneous phase (σ_{ap}) and standard deviation of the direct value of the non-linear component of the instantaneous phase (σ_{dp}), demonstrated respectively by the Equations 2.15 and 2.16 [13]:

$$\sigma_{ap} = \sqrt{\frac{1}{N_c}\left(\sum_{A_n[n]>A_t)} \phi_{NL}^2[n])\right) - \left(\frac{1}{N_c}\sum_{A_n[n]>A_t} |\phi_{NL}[n]|\right)^2} \tag{2.15}$$

$$\sigma_{dp} = \sqrt{\frac{1}{N_c}\left(\sum_{A_n[n]>A_t)} \phi_{NL}^2[n])\right) - \left(\frac{1}{N_c}\sum_{A_n[n]>A_t} \phi_{NL}[n]\right)^2} \tag{2.16}$$

where N_c is the number of samples with an amplitude above a specific threshold value (A_t), which filters out the low-amplitude samples taking out the noise. The term ϕ_{NL} is the non-linear component of the instantaneous phase of the nth signal sample, i.e., the signal phase without the contributions of the carrier frequency, as defined by:

$$\phi_{NL} = \phi_{uw} - \frac{2\pi f_c}{f_s} \tag{2.17}$$

where f_c is the carrier frequency, f_s is the sampling frequency, and ϕ_{uw} is the unwrapped phase sequence. The next two features are the standard deviation of the absolute value of the normalized and centered instantaneous amplitude of the signal (σ_{aa}) and standard deviation of the absolute value of the normalized and centered frequency of the signal (σ_{af}), shown respectively by the Equations 2.18 and 2.19 [13]:

$$\sigma_{aa} = \sqrt{\frac{1}{N}\left(\sum_{n=1}^{N} A_{cn}^2[n]\right) - \left(\frac{1}{N}\sum_{n=1}^{N} |A_{cn}[n]|\right)^2} \tag{2.18}$$

$$\sigma_{af} = \sqrt{\frac{1}{N_c}\left(\sum_{A_n[n]>A_t)} f_N^2[n]\right) - \left(\frac{1}{N_c}\sum_{A_n[n]>A_t} |f_N[n]|\right)^2} \tag{2.19}$$

where the centered instantaneous frequency $f_N[n]$ is normalized by the sampling frequency f_s, (Equations 2.20 and 2.21), and μ_f is the frequency mean [13]:

$$f_N[n] = \frac{f_m[n]}{f_s} \tag{2.20}$$

$$f_m = f[n] - \mu_f \tag{2.21}$$

The mean value of the signal magnitude (X) is also used, Equation 2.22, and the normalized square root value of sum of amplitude of signal samples (X_2) [15], Equation 2.23:

$$X = \frac{1}{N}\sum_{n=1}^{N} |A[n]| \tag{2.22}$$

$$X_2 = \frac{\sqrt{\sum_{n=1}^{N} |A[n]|}}{N} \tag{2.23}$$

Another two features that use instantaneous values are the kurtosis of the normalized and centered instantaneous amplitude (μ_{42}^{a}) and kurtosis of the normalized and centered instantaneous frequency (μ_{42}^{f}), Equations 2.24 and 2.25:

$$\mu_{42}^{a} = \frac{E\left\{A_{cn}^{4}[n]\right\}}{\left\{E\left\{A_{cn}^{2}[n]\right\}\right\}^{2}} \tag{2.24}$$

$$\mu_{42}^{f} = \frac{E\left\{f_{N}^{4}[n]\right\}}{\left\{E\left\{f_{N}^{2}[n]\right\}\right\}^{2}} \tag{2.25}$$

2.2.3 Moments and Cumulants

High order statistical features are being used as features to automatic modulation classification for a long time. They are based on expected values, like the standard deviation, the variance, and the kurtosis. Since the received signals are complex-valued, the complex moments and cumulants are defined in Equations 2.26 to 2.33. The notation of moments and cumulants are based on how many conjugates there are [16] [17] [18] [19]:

$$M_{pq} = E[y(n)^{p-q}(y^{*}(n))^{q}] \tag{2.26}$$

$$C_{20} = Cum(X, X) = M_{20} = E[y^{2}(n)] \tag{2.27}$$

$$C_{21} = Cum(X, X^{*}) = M_{21} = E[|y(n)|^{2}] \tag{2.28}$$

$$C_{40} = Cum(X, X, X, X) = M_{40} - 3M_{20}^{2} \tag{2.29}$$

$$C_{41} = Cum(X, X, X, X^{*}) = M_{41} - 3M_{20}M_{21} \tag{2.30}$$

$$C_{42} = Cum(X, X, X^{*}, X^{*}) = M_{42} - |M_{20}|^{2} - 2M_{21}^{2} \tag{2.31}$$

$$C_{60} = Cum(X, X, X, X, X, X) = M_{60} - 15M_{40}M_{20} + 30M_{20}^{3} \tag{2.32}$$

$$C_{63} = Cum(X, X, X, X^{*}, X^{*}, X^{*}) = M_{63} - 6M_{41}M_{20} - 9M_{42}M_{21} + 18M_{20}^{2}M_{21} + 12M_{21}^{3} \tag{2.33}$$

where E is the expected value and Cum is the cumulant. The notation often used is C_{ab}, where a is the number of times the input is repeated, and b is the number of conjugates on the a inputs.

Table 2.1 summarizes some of the features found in the literature and state-of-the-art chosen for study in this work. All features presented from Equation 2.12 to Equation 2.33 are listed on Table 2.1.

Table 2.1: List of selected features

Feature name and number
01 - γ_{max} (Gmax)
Continued on next page

Table 2.1 – continued from previous page

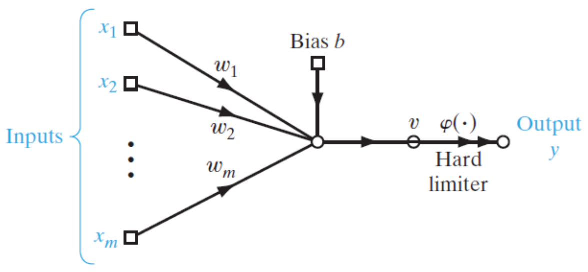
Feature name and number
02 - Standard deviation of the absolute instantaneous phase
03 - Standard deviation of the instantaneous phase
04 - Standard deviation of the absolute instantaneous centered normalized amplitude
05 - Standard deviation of the instantaneous frequency
06 - Mean value of the signal magnitude
07 - Normalized square root value of sum of amplitude
08 - Kurtosis of the centered normalized amplitude
09 - Kurtosis of the centered normalized frequency
10 - Cumulant C_{20}
11 - Cumulant C_{21}
12 - Cumulant C_{40}
13 - Cumulant C_{41}
14 - Cumulant C_{42}
15 - Cumulant C_{60}
16 - Cumulant C_{61}
17 - Cumulant C_{62}
18 - Cumulant C_{63}

2.3 ARTIFICIAL NEURAL NETWORKS

In this work, an ANN (Artificial Neural Network) is used to solve a classification problem. For this specific task, the computer is asked to specify which of the predefined categories some input belongs. As an example of a classification task, object recognition is one of the most used. The input is usually an image. The ANN returns a numeric code representing the class of the object found in the image with the most significant probability of being correct [20].

The single neuron model, known as the perceptron, is shown in Figure 2.6 and it can be expressed mathematically in the Equation 2.34, where w is the weight, b is the bias, x is the input, m is the number of inputs and v is the input of the activation function.

Figure 2.6: Perceptron model



(HAYKIN, 2009)

$$v = \sum_{i=1}^{m} w_i x_i + b \tag{2.34}$$

The single neuron perceptron is limited and can only classify two classes. More neurons can be added to upgrade the ANN capability to classify patterns [21]. Every neuron computes a linear combination of its inputs, using weights for every input and a bias for each neuron. The result of this calculation is then applied to a hard limiter, which is a fixed nonlinear function called activation function [20].

2.3.1 Activation functions

The most known and used activation functions in ANN are the ReLU (Rectified Linear Unit) function, the sigmoid function, and the hyperbolic tangent function. The most simple is the ReLU function because it returns zero if the input is negative and returns the input if it is positive. The ReLU function is shown in the Equation 2.35, and the most notorious advantage of using it is its linearity versus the non-linearity of the other functions [21].

The sigmoid and hyperbolic tangent are very similar, the main difference between the two functions is the output when the input is negative. Their Equations and the relationship between the two are shown in Equations 2.36 to 2.38.

$$g(v) = \max(0, v) \tag{2.35}$$

$$g(v) = \sigma(v) = \frac{1}{1 + e^{-v}} \tag{2.36}$$

$$g(v) = \tanh(v) = \frac{e^{v} - e^{-v}}{e^{v} + e^{-v}} \tag{2.37}$$

$$tanh(v) = 2\sigma(2v) - 1 \tag{2.38}$$

where v is defined by the Equation 2.34. The main problem with these non-linear function is the saturation that may occur due to a very high or very low input. This saturation can make the learning algorithm difficult because it often depends on the gradient of those functions. For this reason, their usage is discouraged [21].

2.3.2 Multilayer artificial neural networks

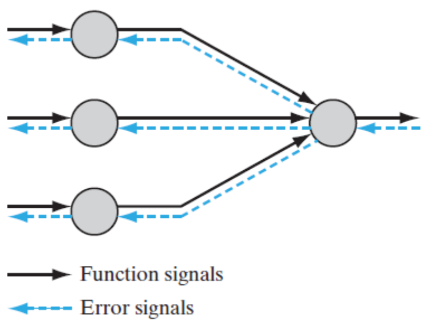
A multilayer perceptron is a ANN that contains more than one layer fully-connected to another. If there is any layer between the input layer and the output layer, it is called a hidden layer. Because the ANN is highly connected, the theoretical analysis of a multilayer ANN is challenging to undertake. The use of hidden layers of neurons makes the learning process obscure, and the understanding of what happens in those hidden neurons is a subject of proper research [20].

The most common algorithm to update the neuron's weights and biases for training is the backpropagation algorithm. It is evaluated in two phases: the ANN weights and biases are fixed (and randomly generated if it is the first time) in the forward phase. The input signal is propagated through the entire ANN during this phase.

The backward phase occurs when an error signal is propagated through the ANN from output to input. The adjustments of weights and biases of the ANN are made during this phase. [21]. The training algorithm's primary goal is to minimize the output error when comparing the classification result with the ground truth. The propagation signals used in both phases are shown in Figure 2.7.

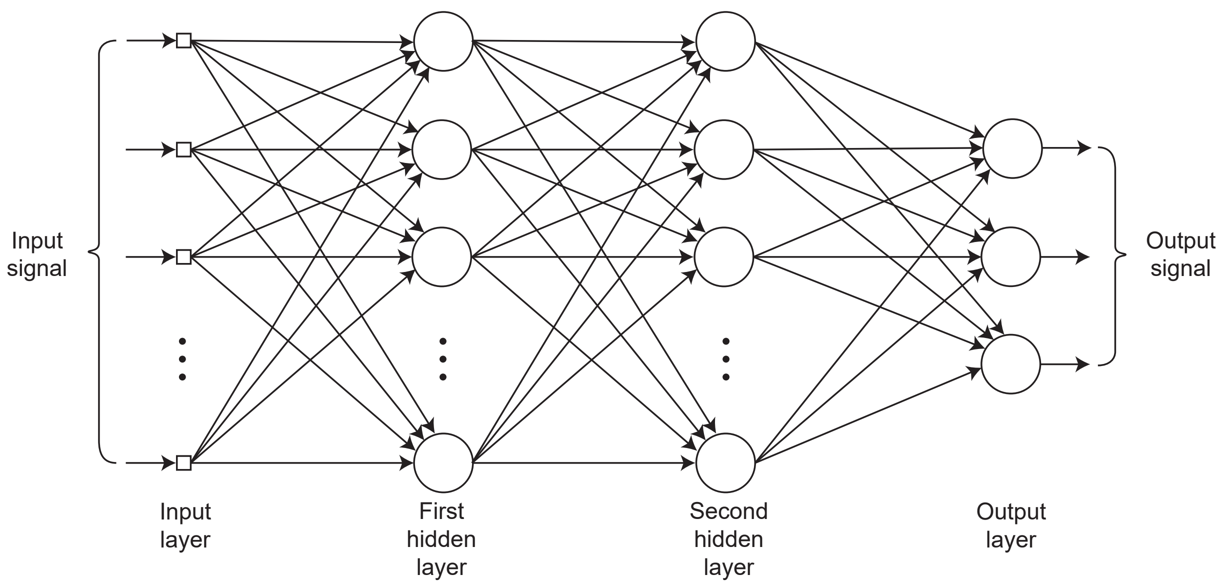
Figure 2.8 shows an example of a multilayer ANN with two hidden layers. The classification is performed when the signal flows from the inputs to the outputs. Each output corresponds to a class, and the output with the higher value is the class chosen by the ANN based on the selected input. The softmax function is commonly used at the end of multilayer ANN to select the output with a higher value automatically.

Figure 2.7: Propagation signals



(HAYKIN, 2009)

Figure 2.8: Multilayer perceptron



(HAYKIN, 2009)

2.3.3 Overfitting and underfitting

The main difficulty in training neural ANN is to avoid underfitting and overfitting the accuracy of the ANN. The underfitting problem can occur when the ANN fails to classify a determined dataset. It can be related to the dataset's size being too small and insufficient to differentiate the classes and not enough training. The dataset is also divided into training and test, so the proportion of the dataset used to train and test can lead to overfitting or underfitting.

The overfitting problem is related to the ability to classify a training dataset perfectly but performing poorly in test datasets that are different from the training dataset. It can be related to the ANN size and excessive training, the ANN memorizes the training data [21].

A technique called Dropout [22] can be used to prevent overfitting in a ANN. This technique consists of randomly disconnect neurons from each layer during the training. There are also other techniques like L1 regularization (Lasso Regression [23]) and L2 regularization (Ridge Regression [24]) and soft weight sharing [25][26].

2.3.4 Normalization and standardization

One of the most common and efficient kinds of normalization is the z-score normalization, also called standardization. It removes the mean and scales the input to unit variance [27]. Also, many elements used in a machine learning algorithm, such as L1 and L2 regularizers, assume that the ANN input is centered around zero and variance of the same order [28]:

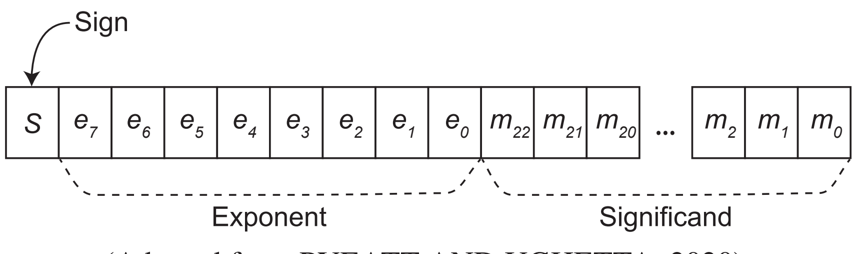
$$z = \frac{(x - \mu)}{\sigma} \tag{2.39}$$

where x is the input, μ is the mean of the input, σ is the standard deviation of the input and z is the normalized output. Any kind of normalization of data before the training of neural ANN is often beneficial to the ANN performance. However, it has been shown that if the training data is too large, the difference between the ANN performance (considering original and transformed data) is small. It is because ANN have the capabilities of learning all characteristics from a dataset [29] [30] [31].

2.4 WORKING WITH EMBEDDED SYSTEMS

The most used types of numbers in computing are floating-point numbers. Their accuracy and range are high enough to solve numerous problems with only 32 bits. The 32-bit floating-point numbers are called single precision. For 32-bit numbers, 8 bits are used for the exponent, while 23 bits are used for the mantissa. The first bit of the number is used to define its positive or negative sign, completing the 32 bits [32]. Many embedded systems have specific hardware for accelerating calculations involving floating-point numbers. The name given to this type of hardware is Floating-Point Unit (FPU). The IEEE (Institute of Electrical and Electronics Engineers) 754 Standard defines the representation of the floating-point number for Floating-Point Arithmetic (Figure 2.9).

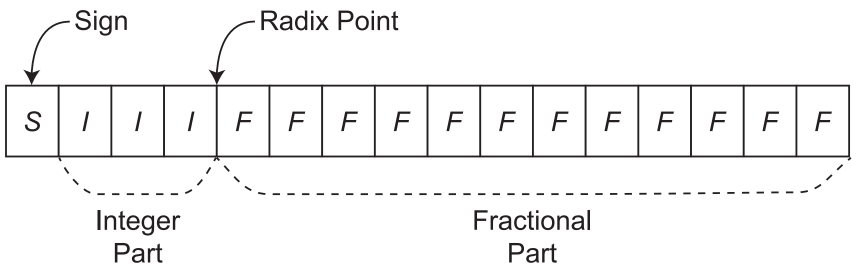
Figure 2.9: Floating-point number representation


Sign
S
e_7 e_6 e_5 e_4 e_3 e_2 e_1 e_0
m_{22} m_{21} m_{20} ... m_2 m_1 m_0
Exponent
Significand

(Adapted from PYEATT AND UGHETTA, 2020)

The representation of fixed-point numbers is shown in Figure 2.10. The first bit can be used to define the positive or negative sign of the number. The rest of the bits are divided between integer and fractional parts. One of the notations used to define fixed point is the format $Q(I, F)$, where I represents the number of bits for the integer part and F represents the number of bits for the fractional part [32].

Figure 2.10: Fixed-point number representation



(Adapted from PYEATT AND UGHETTA, 2020)

Fixed point numbers are widely used when hardware needs to be fast and straightforward. Their implementation is more manageable than floating-point numbers. They can work with fewer bits when the precision and range allows, thus taking up less space in memory. The transformation of floating point numbers into fixed point is done through the Tensorflow library [33]. The main characteristics of fixed-point numbers are their range and resolution. Both are shown in Table 1 for a specific number of fixed-point numbers in Q format [32].

Table 2.2: Fixed-point numbers range and precision

Q format number	Minimum	Maximum	Precision
Q0.15	-0.5	0.49996948	3.0517578e-05
Q1.14	-1	0.99993896	6.1035156e-05
Q2.13	-2	1.9998779	1.2207031e-04
Q3.12	-4	3.9997559	2.4414062e-04
Q4.11	-8	7.9995117	4.8828125e-04
Q5.10	-16	15.999023	9.7656250e-04
Q6.9	-32	31.998047	1.9531250e-03

Some processor architectures like Advanced Risc Machine (ARM) have libraries available for processing ANN developed specifically for best use of the architecture (like CMSIS-NN [34]). The use of ANN in embedded systems is done using numbers in fixed-point format. The fixed-point numbers usually vary between 8 and 16 bits depending on the precision required and available space in memory. Fixed-point numbers with more bits can describe decimal numbers with more precision while occupying more memory. Some researches on the implementation of ANN in embedded systems even use less than 8 bits [35].

3 METHODOLOGY

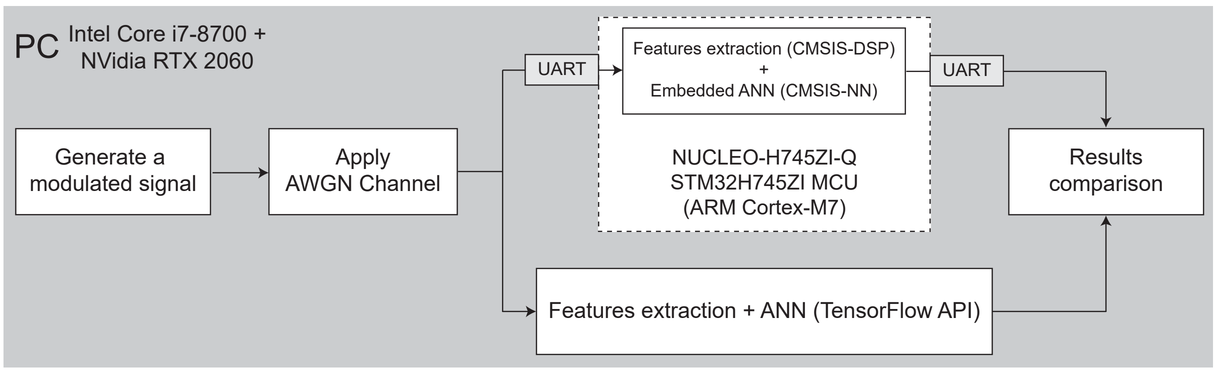
This chapter is divided into topics to understand better all the methodology involved in developing this project. The first step is to correctly generate all desired modulation signals and a white noise signal with the same power to calculate features. The normalization of the modulated signals prior to the calculation of the features is arbitrary, because if needed, the normalization in amplitude is described in the feature's equation. However, all signals will be generated with 0 dBW power so that the comparison between the signals is straightforward, and to be easier to add the white Gaussian noise with the desired signal-to-noise ratio.

After the generation of signals is validated through the time domain plot and the scatter plot (constellation diagram), the features that will be effectively used will be selected based on the embedded system's performance and their capability to recognize modulations of the neural network. With the dataset created and the features selected, the next step is to search for a good performance neural network to achieve the desired results in accuracy and resources needed to run the system. The search for a good network includes a parameter sweep analysis, like changing the number of neurons by layer and the number of layers.

After the neural network is selected, the next step is to implement that network into the embedded system using the tools available. The selected embedded system is the ARM Cortex-M7 [36], one of the high performance and low-cost system available on the market. The processor is embedded in an ST Microelectronics NUCLEO-H745ZI-Q board, using an STM32H745ZI microcontroller unit [37] [38]. This device was selected because it is expected to be more than enough to accomplish its task concerning the memory available and the processor's instruction set.

The results obtained by the embedded network can be compared to the ideal network trained on the PC. The goal is to pass a modulated signal through both systems and compare the results, as shown in Figure 3.1. All features calculations on the PC are made using Python 3 and an Intel Core i7 8700 processor, and the neural networks are trained using Google's Tensorflow 2 API (Application Programming Interface) with NVidia's RTX 2060 GPU (Graphic Processing Unit) to accelerate the training time.

Figure 3.1: Proposed methodology for comparing results


PC
Intel Core i7-8700 + NVidia RTX 2060
Generate a modulated signal
Apply AWGN Channel
UART
Features extraction (CMSIS-DSP) + Embedded ANN (CMSIS-NN)
NUCLEO-H745ZI-Q STM32H745ZI MCU (ARM Cortex-M7)
UART
Results comparison
Features extraction + ANN (TensorFlow API)

(THE AUTHOR, 2020)

3.1 GENERATION OF MODULATED SIGNALS

For every new frame, a random binary stream is generated with size $L\log_2(M)$ so that when the modulation occurs, every frame has the same number of symbols. The shaping filter selected is the raised cosine (RC) filter, with a roll-off factor of 0.2, 8 samples per symbol, and filter span equals 10. Since this project's intuit is not to study the effects of a pulse-shaping filter during the modulation classification, the parameters used are the default values encountered in MATLAB software. The raised cosine filter is often used in wireless communications, and its primary goal is to eliminate or control intersymbol interference. This distortion causes one symbol to interfere with subsequent symbols [6] [39].

The signal coming in the antenna is unknown, and the receiver does not know the incoming signal phase to synchronize itself. A normal distribution of values with zero mean and standard deviation equal to one radian will be generated for the initial random phase to simulate that behavior. For the simulation to be more controlled and the same initial phases to be tested for all SNR values, a fixed number of phases will be generated equal to the number of frames created.

All signals will be created in MATLAB with 32-bit floating-point precision because it is the maximum floating-point precision that the selected embedded system can calculate in hardware, and the proposed modulations to be used are BPSK, QPSK, 8PSK, 16QAM, and 64QAM. One of the objectives of this work is to reduce the number of features used in a feature-based classification looking for optimization concerning hardware use. Because of that, only five modulations were selected among all modulations used by other paper's work in the state-of-the-art. All features will be extracted from the white Gaussian noise to be used as an input to the neural network to see if the network can identify if it is receiving a modulated signal or just noise.

Algorithm 1 Dataset generation

```
for all modulations do
  for $SNR = -10$ to 20 with step = 2 do
    for $frame = 0$ to 1000 do
      generate new frame
      apply random phase offset
      normalize power
    end for
  end for
end for
```

3.2 FEATURE SELECTION

After studying all features, it is intended to select a few of them to be used in the final neural network. The proposal is to reduce the computational cost and thus reduce energy consumption, aiming at the application in low-budget embedded systems.

The methodology of selecting the features will be to analyze these features' behavior with the signals modulated in different SNR and phase configurations. All the selected features for this work will also be implemented in an embedded system to analyze the calculation time and computational complexity required. Based on this information, it will be possible to considerably reduce the number of features used in the final neural network.

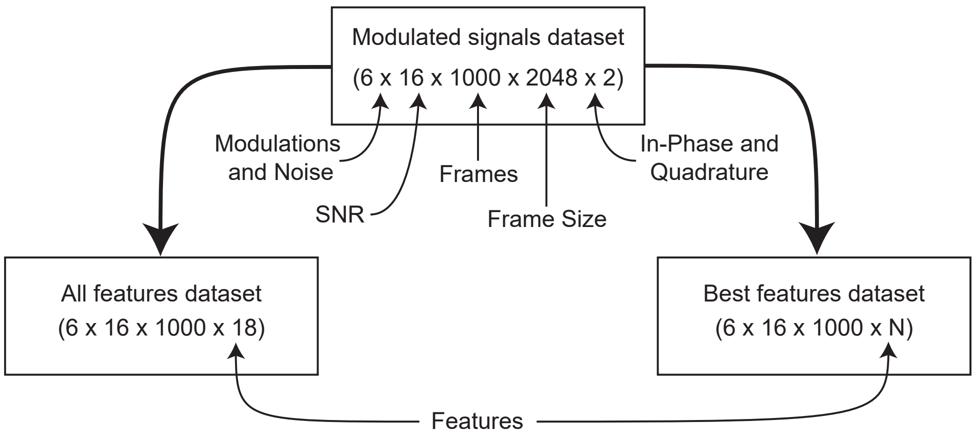
Other feature selection methods can be found in the literature, and they take into account the importance of the feature to the ANN using mathematical analysis. However, as this project requires a finite time to be done, they will be left for future work.

3.3 ARTIFICIAL NEURAL NETWORKS

The artificial neural network consisting of an input layer, three hidden layers and an output layer is trained done using the Keras API for Tensorflow. The dataset will consist of 1000 frames per SNR for each modulation. This database of the modulated signals will be converted into two distinct datasets, one containing all the studied features and the other containing the values of the best-selected features, where N is the number of selected features (Figure 3.2). The noise is simulated with the same power multiple times for the variation of SNR of other modulations. It is done to simplify the matrix use and creation of the dataset and store it in memory.

The goal behind creating datasets with a different number of features is to compare the performance of neural networks to define whether or not it is viable to use only a few features to classify modulations. Both neural nets will be trained using different parts of the dataset. The first training will be done with all available SNR values. Then newly trained nets will be created only with SNR equal to or greater than zero. Finally, new networks will be created and trained with SNR equal to or greater than 10 dB. With this result, it will be possible to identify whether neural network training with low SNR values is viable. Every neuron network will use all the available techniques to avoid overfitting.

Figure 3.2: Distinct datasets in detail

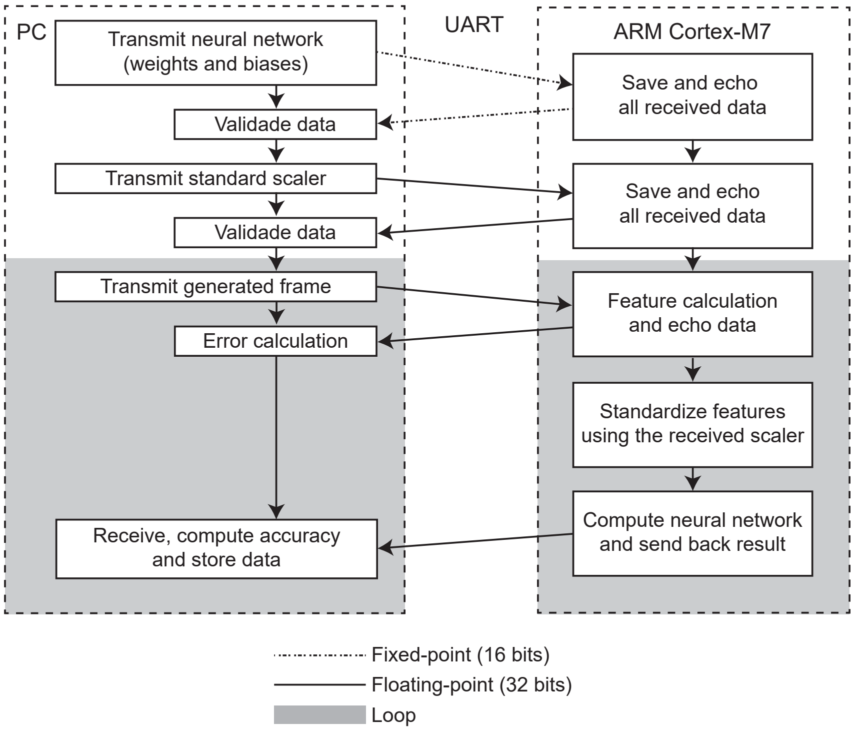


(THE AUTHOR, 2020)

Figure 3.3 summarizes how the communication between PC and the embedded system will occur. The neural network's weights and biases will be converted into 16 bit fixed-points and sent to the board via UART communication because the CMSIS-NN requires it to evaluate the ANN. The board will respond with an echo of all received values, and if this echo is valid, then the communication can continue. The standard scaler used to normalize and standardize the data before training is also sent to the board and validated through the same echo process. After this first part of sending data to the board, the embedded system will wait until a full frame of 4096 32-bit floating-point values is sent (2048 for In-Phase and 2048 for Quadrature).

The frame size controls the number of symbols that can be used to calculate features, so with a higher number of samples and symbols, the features will be more accurate. However, it will also need more available memory in the embedded system. Choosing a frame size is a compromise between feature precision and memory allocation, and 2048 samples are the maximum power of 2 number samples that does not overflow the selected embedded system. The frame size needs to be a power of 2 so that the Fast Fourier Transform (FFT) algorithm can be used. After calculating features by the embedded system, the features are standardized using the previously sent standard scaler, and then the features are input into the neural network. The classification result is sent back to the PC to compute accuracy and store the data.

Figure 3.3: Embedded system communication and tests


PC
UART
ARM Cortex-M7
Transmit neural network (weights and biases)
Validade data
Transmit standard scaler
Validade data
Transmit generated frame
Error calculation
Receive, compute accuracy and store data
Save and echo all received data
Save and echo all received data
Feature calculation and echo data
Standardize features using the received scaler
Compute neural network and send back result
Fixed-point (16 bits)
Floating-point (32 bits)
Loop

(THE AUTHOR, 2020)

The neural network to be implemented in an embedded system will be the network that will have the best features as input and is optimized regarding the number of neurons by layer. The optimization of neural networks will be done using the sweeping tools available at Weights and Biases website and Python library to track the experiments [40].

The neural network could be implemented using 32 bits in floating-point. However, the use of numbers and calculations in fixed-point reduces memory use and allows the neural network to be embedded in systems without floating-point calculation capabilities. The calculation of the features is done using floating-point numbers. It ends up requiring an FPU or a processor that can calculate floating-point numbers, so it would be interesting to transform the features into fixed-point numbers. However, the analysis of the feature's performance in fixed point is better suited for future work. The last test will compare the neural network implementation in the embedded system and its version generated in Python 3.8 on the PC.

For the embedded system, the programming language for the implementation is C. The C++ language was also available, but the object-oriented language is not necessary for this project. The main library used for the signal processing and neural network is the CMSIS from ARM [34].

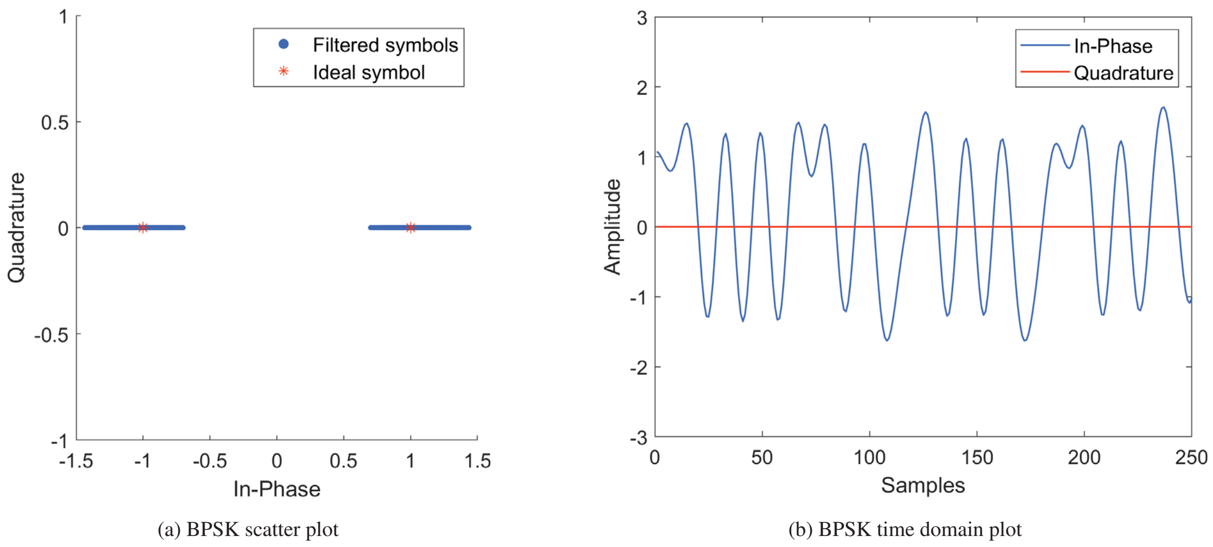
4 RESULTS AND DISCUSSION

The results are written based on the methodology's topics, starting from the generation of modulated signals to the final implementation and comparison of the ANNs.

4.1 MODULATED SIGNAL GENERATION

Figure 4.1 shows a BPSK signal waveform generated for the entire dataset. To be more comfortable to visualize, the time domain signal is cut to the first 250 samples, and the simulation was done without the addition of noise. On the left, the constellation of this modulation is shown using a scatter plot, and on the right is the time domain plot. Every scatter-plot used in this work shows the ideal symbol position for the modulation in context with average unit power and all the symbols simulated via software after applying the RC filter. The results show that the filter changes the symbols' position regarding the ideal mathematical symbol for every modulation. For the validation of the signals in the dataset, the initial phase offset was turned off.

Figure 4.1: BPSK signal example

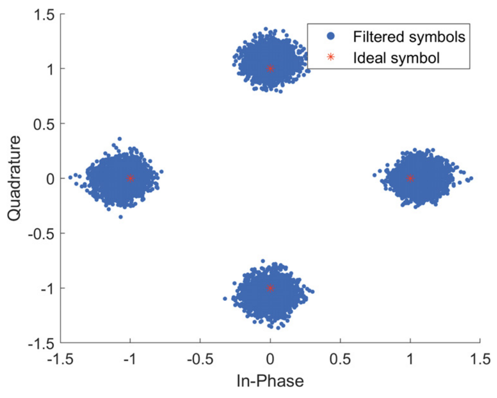


(a) BPSK scatter plot (b) BPSK time domain plot

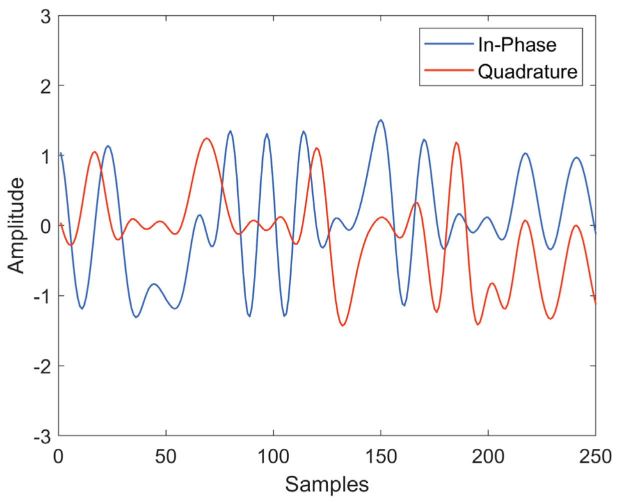
The BPSK signal generation is valid because only the In-Phase signal on the time domain plot for the initial phase equals zero. The symbols are grouped around the ideal symbols expected for this modulation. Figure 4.2 shows a QPSK signal and Figure 4.4 shows a 8PSK signal. Both modulations were simulated correctly because the constellation position after the application of RC filter matches with the ideal symbols, but of course with a slight deviation caused by the filter.

It is shown in Figure 4.3(a) that the application of a $\pi/8$ phase offset changes the position of the constellation regarding the ideal one. The waveform is multiplied by $e^{j\theta}$, where θ is the phase angle, resulting in a rotation of the signal constellation. Figure 4.5 shows the 16QAM signal waveform and its constellation, and Figure 4.6 shows the same for 64QAM. Just like the PSK signals, both 16QAM and 64QAM have their constellation matching with what is expected from the theory.

Figure 4.2: QPSK signal example

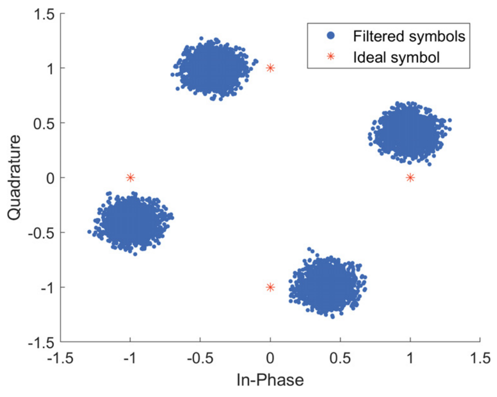


(a) QPSK scatter plot

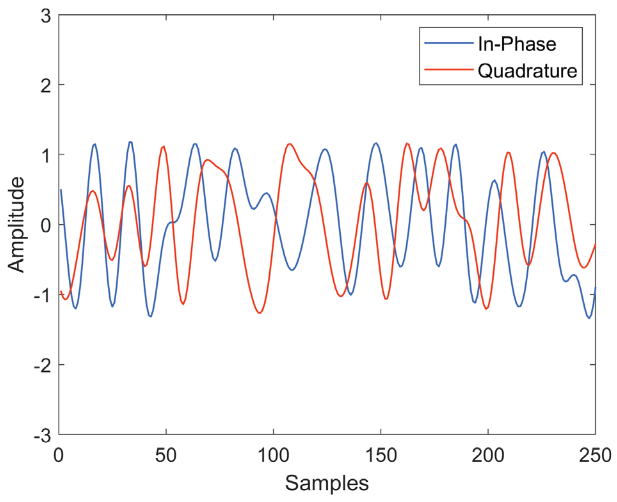


(b) QPSK time domain plot

Figure 4.3: QPSK signal example with phase offset

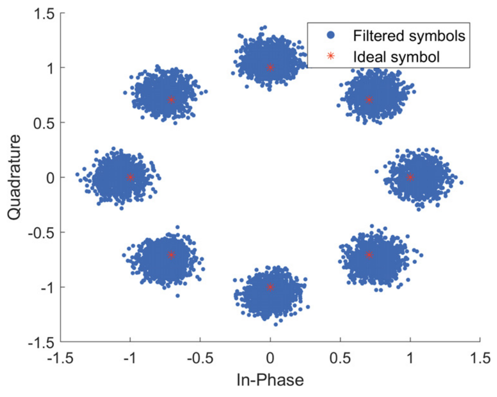


(a) QPSK scatter plot with phase offset

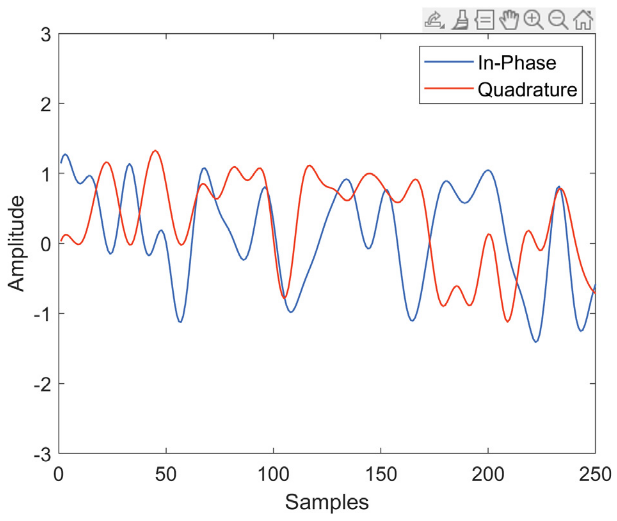


(b) QPSK time domain plot with phase offset

Figure 4.4: 8PSK signal example

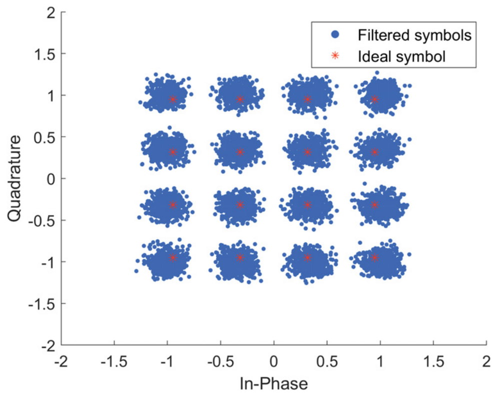


(a) 8PSK scatter plot

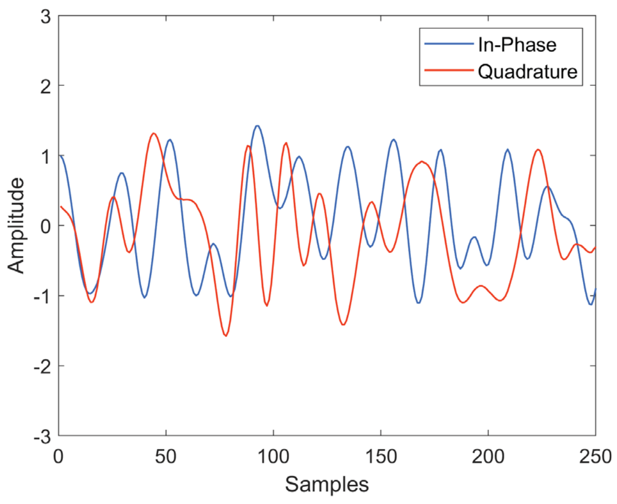


(b) 8PSK time domain plot

Figure 4.5: 16QAM signal example

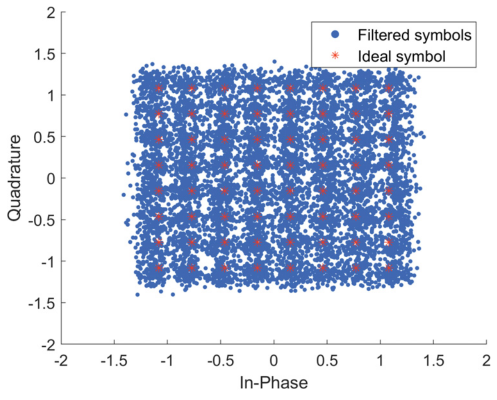


(a) 16QAM scatter plot

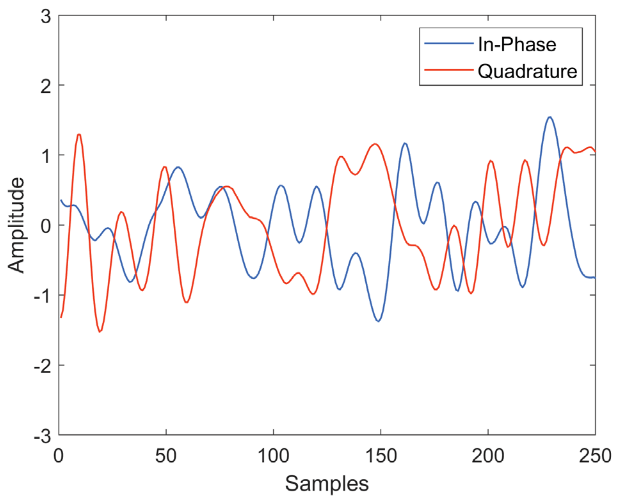


(b) 16QAM time domain plot

Figure 4.6: 64QAM signal example



(a) 64QAM scatter plot



(b) 64QAM time domain plot

All the generated signals are considered the transmitted signals over-the-air received and converted to baseband by the quadrature demodulator. From this point forward, the modulation signal is a complex envelope containing In-Phase and Quadrature components, and all features are calculated using these complex signals.

4.2 FEATURES EXTRACTION

After calculating features from all generated frames of all modulations, it is necessary to analyze the behavior of the feature's values. This analysis is done to see if the values of the features are different for every modulation for the selected range of SNR in the methodology.

The features are being extracted from at least 100 frames for each selected SNR because each frame has its initial phase, and it can change the feature's value. In the end, the mean value of the feature for those 100 frames is calculated for the visual analysis to be more accessible. Also, since the ANN is trained with thousands of frames, and the values of the weights and biases are updated using small batches of features, it can be expected that the average value of the features will prevail.

Figure 4.7 shows an aggregate of a hundred frames for each modulation plotted at the same graphic for the γ_{max} feature. This graphic demonstrates how the feature's value varies depending not only on the noise but on the received phase offset. A mean value is taken for each modulation to simplify the visualization, and the result is shown in Figure 4.8. The γ_{max} feature is expensive computationally because of the need for a Discrete Fourier Transform (DFT).

Figure 4.7: Feature 1 - γ_{max} with 100 frames plotted for each modulation and the noise

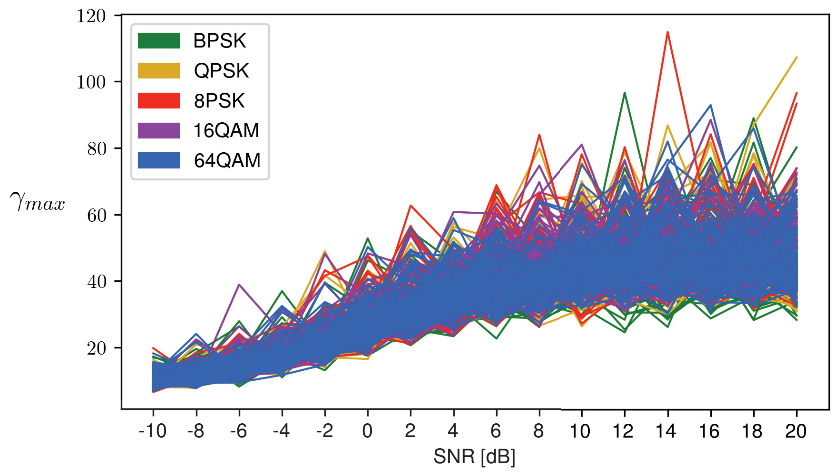
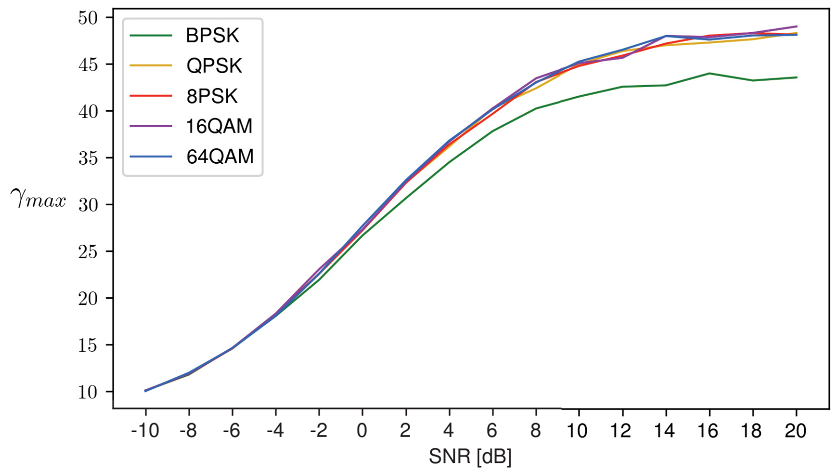


Figure 4.8: Feature 1 - γ_{max} with the mean of 100 frames plotted for each modulation and the noise



Considering Figures 4.7 and 4.8, the variation of noise in the modulated signals changes the value of the γ_{max} feature from less than 20 for all modulations up to almost 120 in one of the 8PSK frames. All modulations show the same result, on average, from -10 dB to 20 dB of SNR, aside from BPSK, which seems to give an average result of γ_{max} below the other modulations.

The features 2 and 3 shown in Figures 4.9 and 4.10 are very similar in behavior considering the variation of SNR. Besides the amplitude difference, feature 2 is more smooth, so it is more robust to phase offset and noise changes. Both features come from calculating the instantaneous phase of the modulated signals, and BPSK is more distinguishable from the other modulations when the SNR goes high.

Figure 4.9: Feature 2 - Standard deviation of the abs. value of the non-linear component of the instantaneous phase

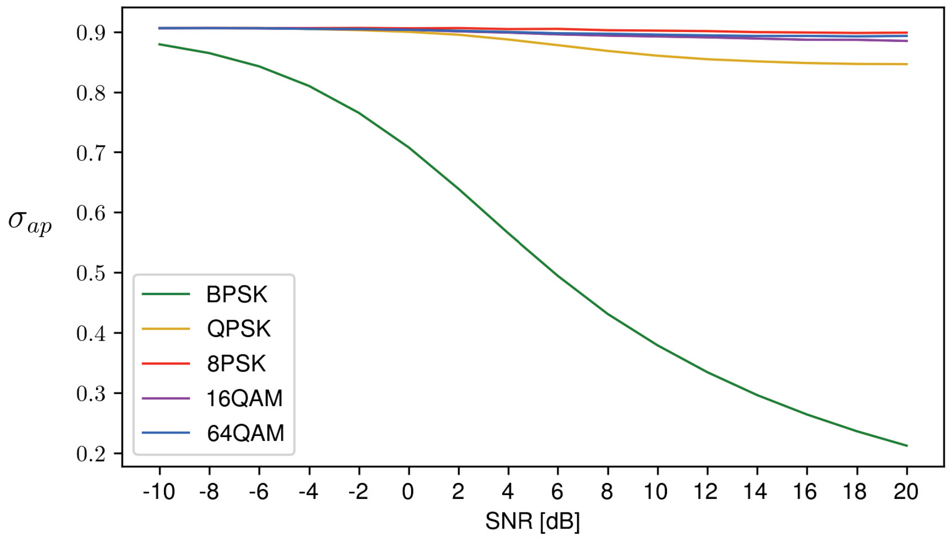


Figure 4.10: Feature 3 - Standard deviation of the non-linear component of the instantaneous phase

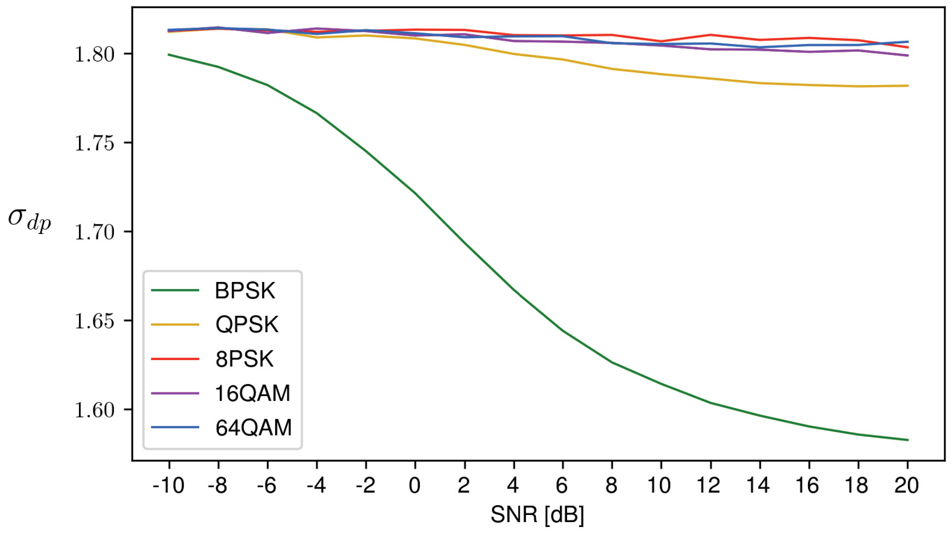


Figure 4.11 illustrates feature four and shows that it tends to separate all modulations' values when the SNR is high. However, for SNR equal and below 8 dB, QPSK and 8PSK modulations have the same value of the feature, and BPSK intercepts 16QAM at 8 dB and 64QAM at 12 dB. This behavior can lead to a decrease in the ANN's accuracy for these discussed values and modulations.

Feature five, on the other hand (Figure 4.12), does not separate very well the modulated signals. This behavior occurs because the signals studied in this project are not modulated in frequency. Therefore the standard deviation of the instantaneous frequency is not suitable for classifying amplitude and phase-modulated signals. It is a proven mistake of this work to have selected such features to study.

Figure 4.13 shows the mean value of the signal magnitude, and it looks good to separate QPSK and 8PSK from the other modulations but not between them. The ANN can use these feature values to help classify QPSK or 8PSK in addition to other features.

Figure 4.11: Feature 4 - Standard deviation of the abs. value of the normalized and centered instantaneous amplitude

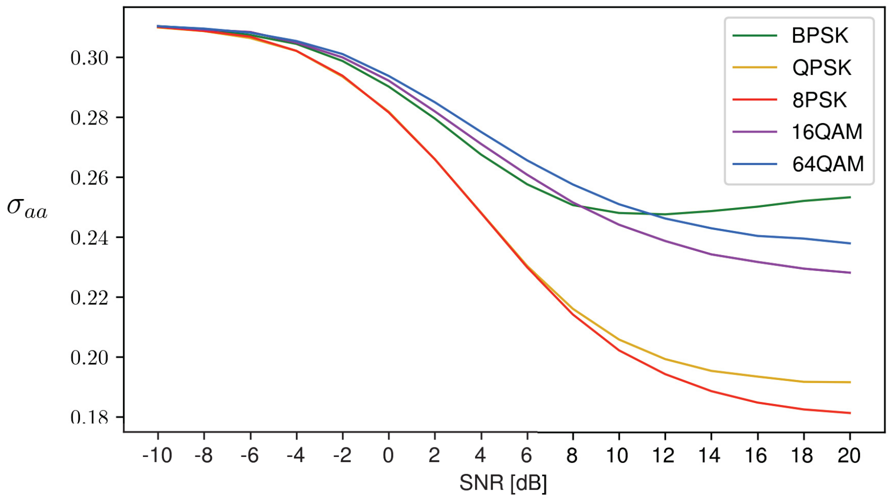


Figure 4.12: Feature 5 - Standard deviation of the abs. value of the instantaneous frequency

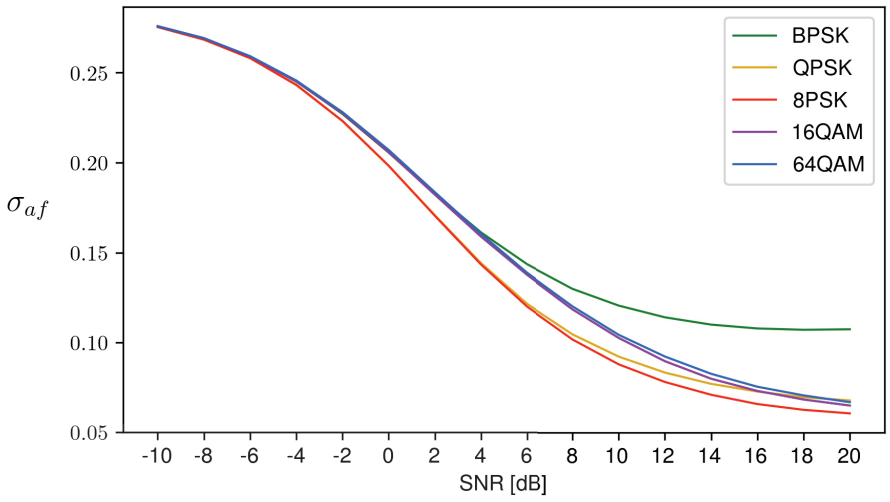
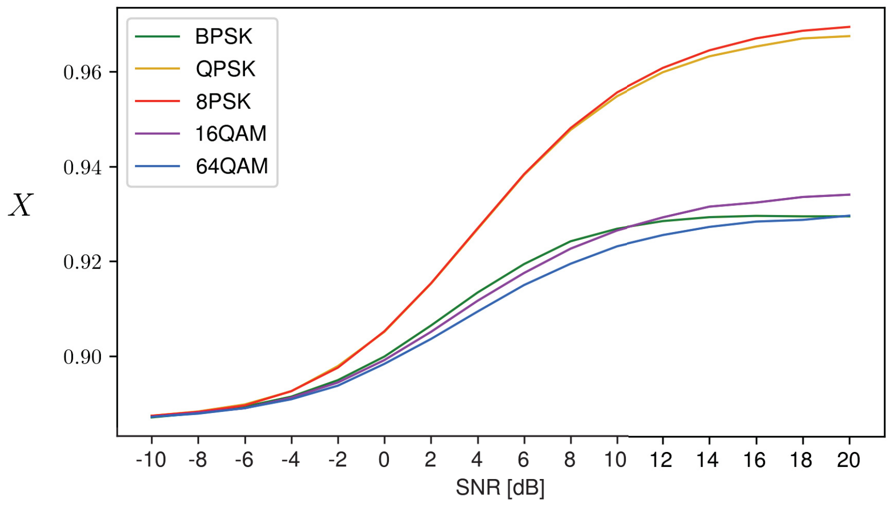
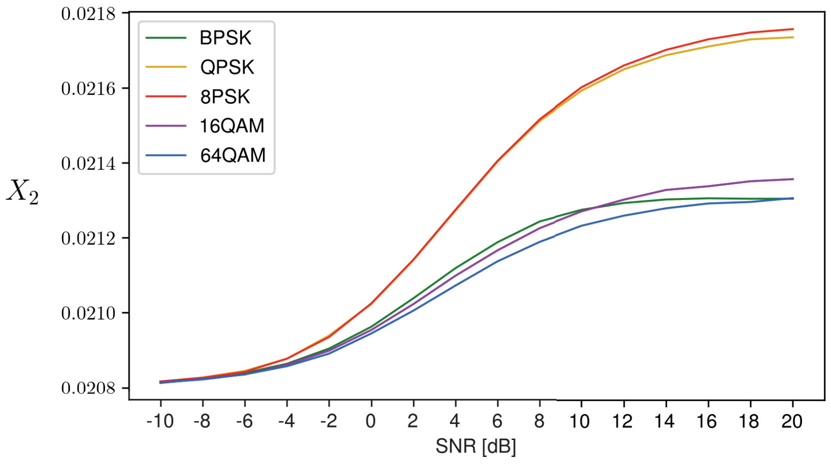


Figure 4.13: Feature 6 - Mean value of the signal magnitude



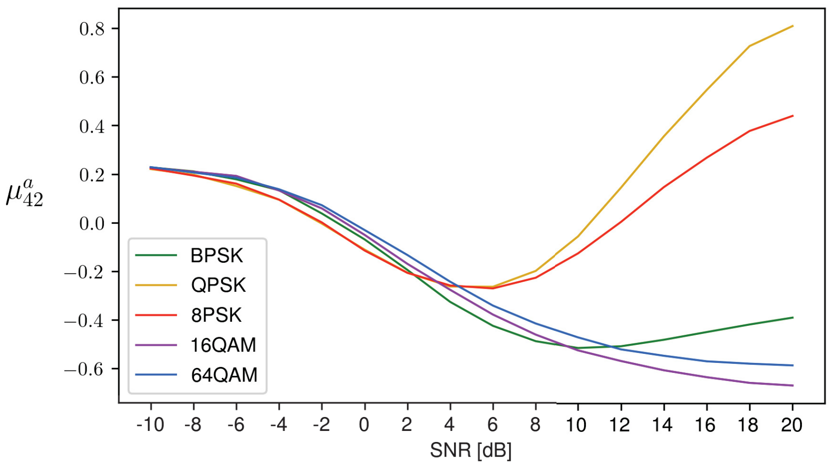
For feature 7, shown in Figure 4.14, its similarity to the previous feature is quite apparent, differing only by its magnitude values. It can be observed that the difference between its maximum and minimum values for this SNR variation is two orders of magnitude smaller than feature 6. These small values can be a problem for identification by the ANN. If, on average, the feature value is almost the same for every modulation, with a random initial phase, they will merge like feature 1.

Figure 4.14: Feature 7 - Normalized square root value of sum of amplitude of signal samples



Feature 8 (Figure 4.15) is one of the best features used so far because it shows a clear separation between the values corresponding to QPSK and 8PSK modulations considering an SNR above 5 dB. Each modulation curve follows a different path. This differentiation is important for this project, since most features seen so far do not show a clear separation, on average, for these modulations.

Figure 4.15: Feature 8 - Kurtosis of the normalized and centered instantaneous amplitude



Feature 9 presents a problem concerning the differentiation between QAM signals. Figure 4.16 indicates that both 16QAM and 64QAM modulations show the same average value through all frames studied for almost all SNR values. The second-order cumulants are shown in Figures 4.17 and 4.18. None of these cumulants gave any satisfactory result because the only modulation that had an advantage in its identification was the BPSK modulation through Feature 10.

Figure 4.16: Feature 9 - Kurtosis of the instantaneous frequency

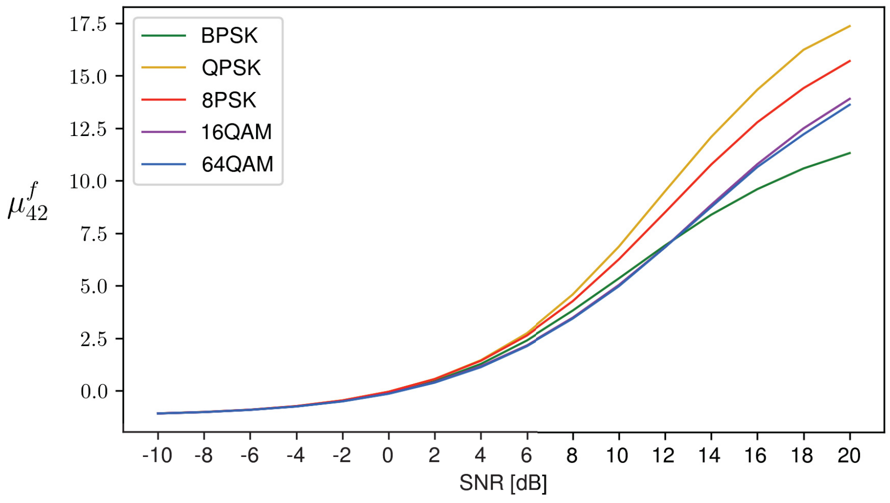


Figure 4.17: Feature 10 - Cumulant 20

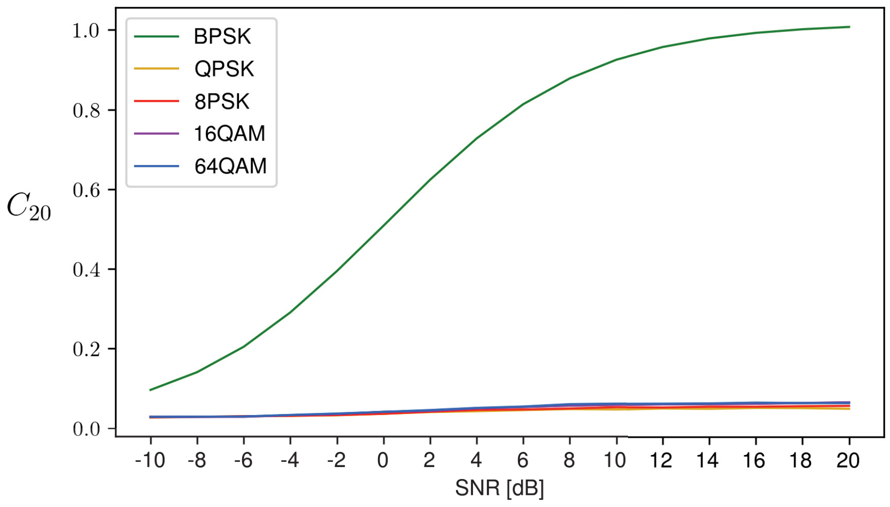
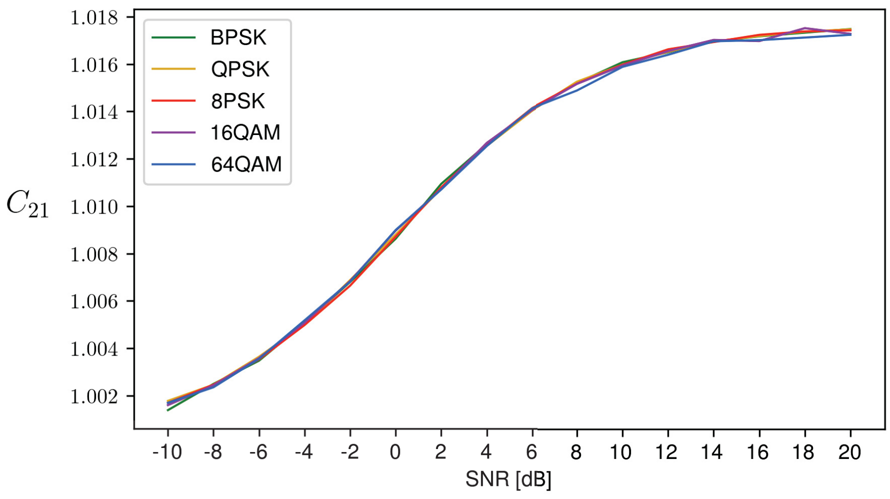


Figure 4.18: Feature 11 - Cumulant 21



The fourth-order cumulants are arranged in Figures 4.19 to 4.21. Feature 12 (Figure 4.19) manages to differentiate the PSK modulations very well. There is a slight difference for the QAM modulations, but it is not considered satisfactory considering the other modulations' difference. Feature 13 (Figure 4.20) shows the same behavior as Feature 10 (Figure 4.17), so it can be discarded because it needs more calculations steps with no actual benefit.

Figure 4.19: Feature 12 - Cumulant 40

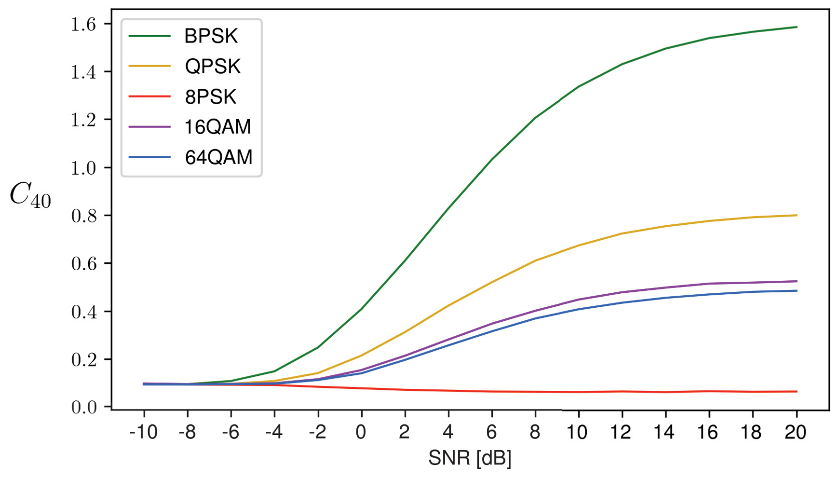
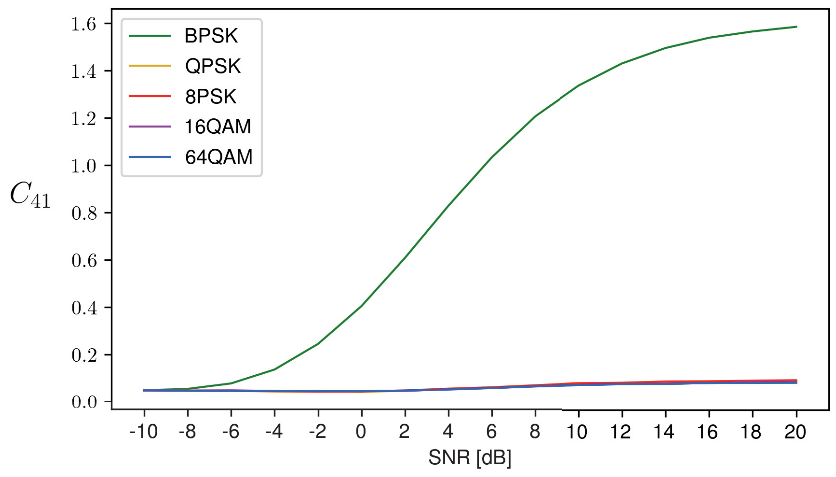


Figure 4.20: Feature 13 - Cumulant 41



Feature 14 (Figure 4.21) is similar to Feature 12, but it has the disadvantage of mixing QPSK and 8PSK completely. Here, one can see that after analyzing many features giving relative values between QPSK and 8PSK, if MPSK modulations with $M > 8$ were used, it will probably be merged with QPSK and 8PSK.

Sixth order cumulants require a longer processing time but do not offer a good cost-benefit. Of the cumulants analyzed, Feature 16 and Feature 18 (Figures 4.23 and 4.25) gave the best result, separating QPSK and 8PSK, respectively, from the other modulations. The only problem is that these cumulants have not yet offered an essential differentiation for QAM modulations. As none of the features presents a clear differentiation for the QAM modulations, the ANN will have difficulties classifying the selected QAM modulations.

Figure 4.21: Feature 14 - Cumulant 42

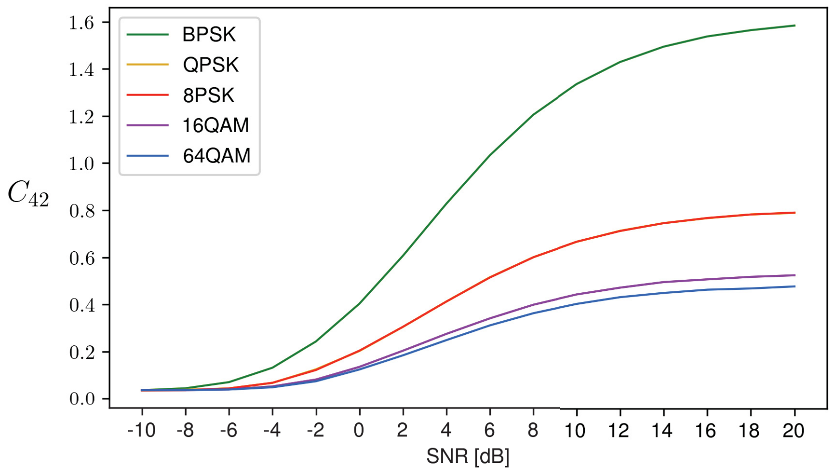


Figure 4.22: Feature 15 - Cumulant 60

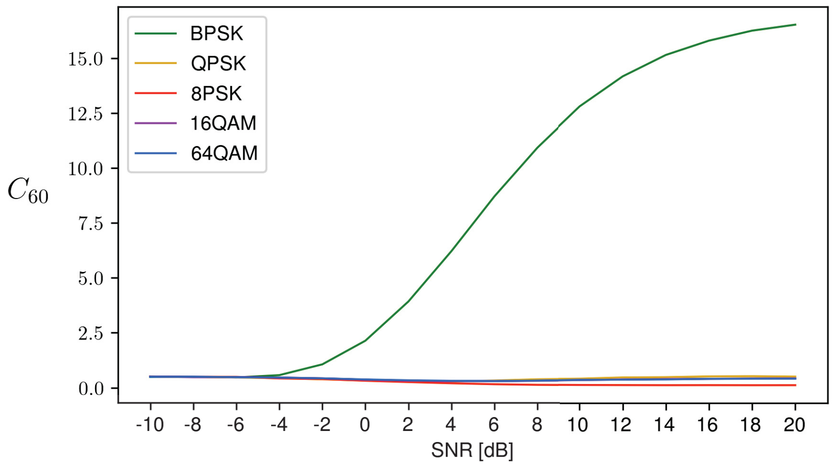


Figure 4.23: Feature 16 - Cumulant 61

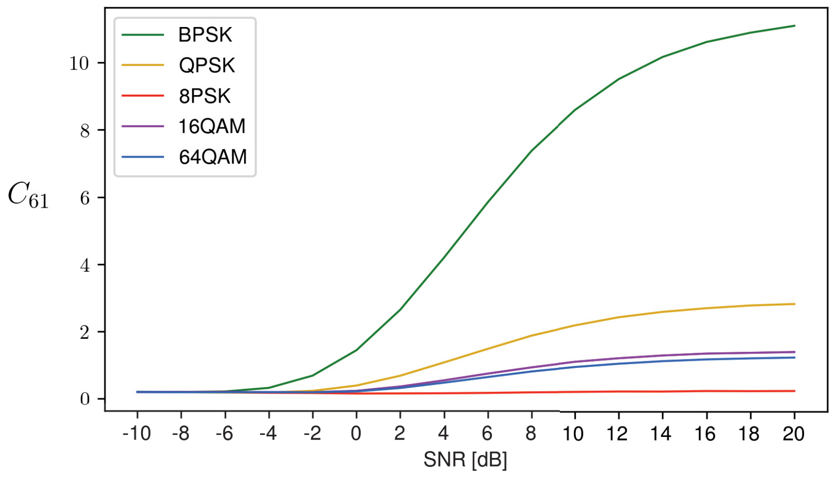


Figure 4.24: Feature 17 - Cumulant 62

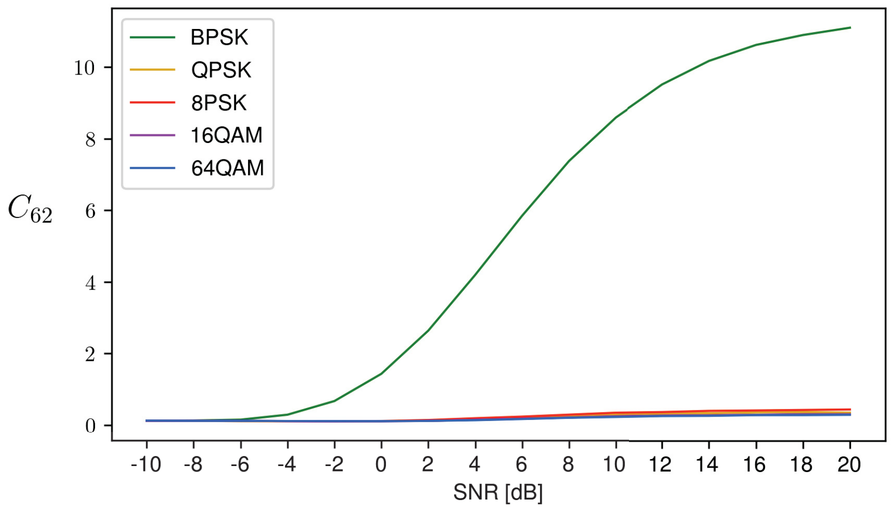
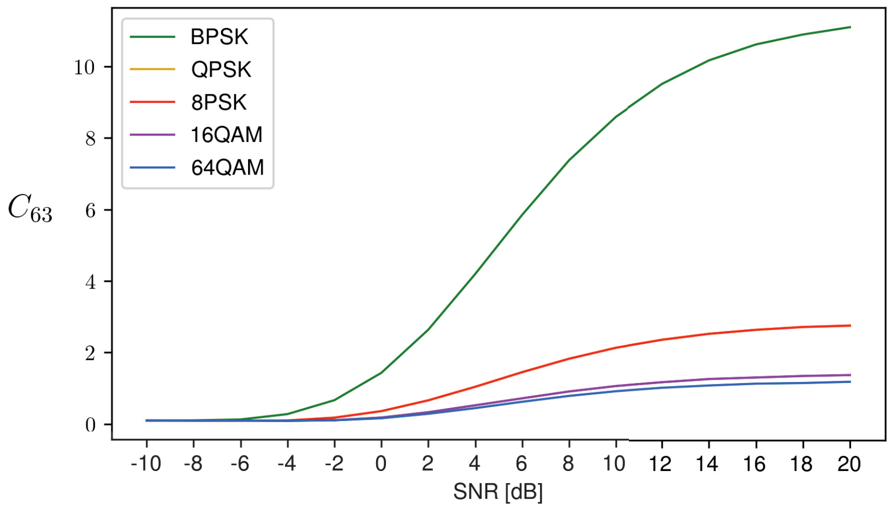


Figure 4.25: Feature 18 - Cumulant 63



After visualizing all features' behavior, they were implemented in the embedded system (ARM Cortex-M7). A selection of the best features could be made considering the graphic results and their computational complexity and hardware use. The Table 4.1 was generated to compare calculation errors between PC and ARM and their times (measured in clock cycles). Because there are errors in the calculations comparing the embedded system and the PC, it can be expected that the implemented ANN will not show the same performance as the original.

The features selected as the best are arranged in Table 4.2. The first feature selected aims at detecting the BPSK modulation. The second feature serves to differentiate all the modulations used in this work. The third feature was selected for its low complexity compared to the others and still helps separate the PSK and QAM modulations, except for the BPSK, as previously discussed. The fourth feature also serves to differentiate all modulations, and the last two features selected are the best cumulants that need less than 200000 clock cycles to be computed.

It is expected that the reduction of the number of features for the ANN will have a decrease in accuracy because, from the eighteen available features from the literature review, only a third of these features remain. Training a ANN with a third of the available information about the modulations is a risky solution. Even if this available information is not precisely the best, the ANNs can learn something from it, thus giving the best performance over the minimal network.

To measure the performance of the ANNs, they were trained using different ranges of SNR. This training expects to see if the ANNs can classify modulating signals with negative SNR and the networks' general behavior for all ranges. Even if the ANN is trained using a small variety of SNR, they are tested using all defined range.

Table 4.1: Feature calculation timing

Features and instantaneous values	**Aprox. max error**	**Timing**
Inst. absolute value	1.19E-07	39786
Inst. phase value	2.38E-07	173906
Inst. unwrapped phase value	1.91E-05	239044
Inst. frequency value	6.34E-07	304598
Inst. centered norm. amp. value	1.43E-06	124762
01 - Gmax	3.81E-06	209080
02 - Std. dev. of the abs. inst. phase	3.15E-04	182906
03 - Std. dev. of the inst. phase	4.06E-04	178930
04 - Std. dev. of the abs. inst. centered norm. amp.	6.01E-05	133080
05 - Std. dev. of the inst. frequency	4.47E-08	309450
06 - Mean value of the signal magnitude	5.36E-07	43994
07 - Norm. square root value of sum of amp.	5.59E-09	54732
08 - Kurtosis of the centered norm. amp.	2.56E-06	66854
09 - Kurtosis of the centered norm. frequency	8.58E-06	331902
10 - Cumulant C_{20}	4.17E-07	39570
11 - Cumulant C_{21}	4.17E-07	55132
12 - Cumulant C_{40}	1.28E-05	88922
13 - Cumulant C_{41}	4.29E-06	147356
14 - Cumulant C_{42}	2.62E-06	147098
15 - Cumulant C_{60}	6.48E-05	158548
16 - Cumulant C_{61}	4.29E-05	273214
17 - Cumulant C_{62}	3.72E-05	370714
18 - Cumulant C_{63}	3.24E-05	375430

Table 4.2: Selected features

Selected features	**Aprox. max error**	**Timing**
02 - Std. dev. of the abs. inst. phase	3.15E-04	182906
04 - Std. dev. of the abs. inst. centered norm. amp.	6.01E-05	133080
06 - Mean value of the signal magnitude	5.36E-07	43994
08 - Kurtosis of the centered norm. amp.	2.56E-06	66854
12 - Cumulant C_{40}	1.28E-05	88922
14 - Cumulant C_{42}	2.62E-06	147098

4.3 ARTIFICIAL NEURAL NETWORKS

Using a generic ANN with hyperparameters described in Table 4.3. Apart from the default values found in Tensorflow and Keras, the ReLU activation function recommended, as discussed in Chapter 2, all other values were random guesses. The ANN use 80% of the input to train the weights and biases and 20% to validade the training.

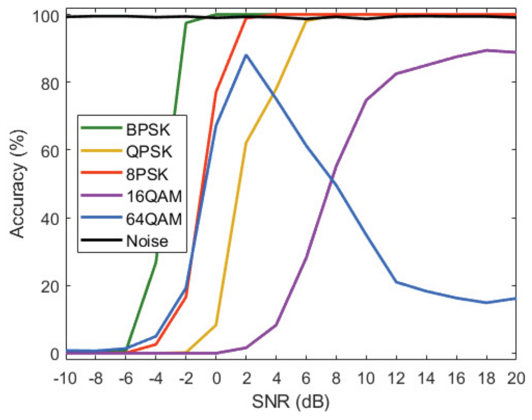
Table 4.3: Generic neural network

Hyperparameter	**Values**
Activation function	ReLU
Batch size	32
Dropout	0.4
Epochs	10
Number of hidden layers	3
Layer size	18
Optimizer	RMSprop (default)
Learning rate	1e-3 (default for RMSprop)

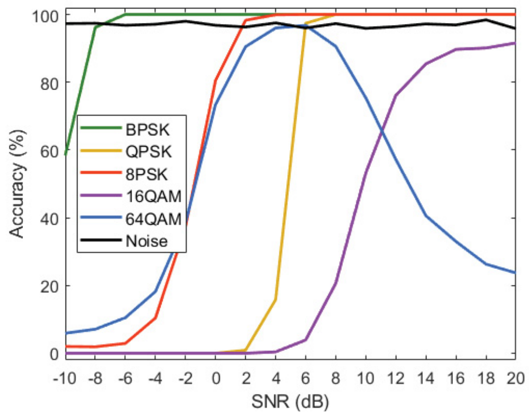
The result of ANN training for applying the six best selected features studied in this work in all the proposed SNR values is in Figure 4.26(a). It is possible to observe good accuracy for most of the modulations, except for 64QAM, because its peak is at 90% accuracy in 2 dB of SNR, and it behaves poorly for higher and lower SNR values. Thinking about a device that will often adapt its connection link due to changes in the telecommunication channel, it can be expected that the received signals will have a high SNR value during most of the communication. So the neural network must correctly classify signals with high SNR.

Training the network with all SNR values is interesting for detecting MPSK modulations with low SNR. This finding can be used in future work to implement a system capable of working in harsh environments. By changing the database to SNR values equal to or greater than zero dB, an interesting phenomenon occurs (Figure 4.26(b)). First, the BPSK modulation is identified with lower SNR values. Then QPSK requires more SNR to be classified, and finally, the 64QAM modulation is a bit better, reaching above 90% from 2 dB to 9 dB of SNR.

Figure 4.26: Training with different datasets



(a) Example of ANN accuracy trained with all dataset

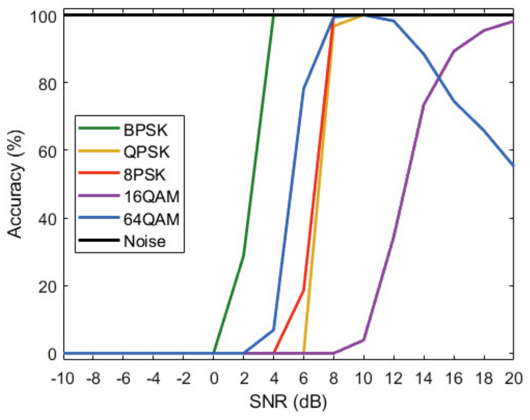


(b) Example of ANN accuracy trained with 0 dB to 20 dB of SNR

Finally, training only with SNR values equal to or above 10 dB, the result is shown in the Figure 4.27. There is a clear improvement in ANN accuracy, especially for 64QAM modulation. The only disadvantage is that the neural net cannot identify any modulation with negative SNR and needs values above 15 dB SNR to discern 16QAM modulation with accuracy above 80%.

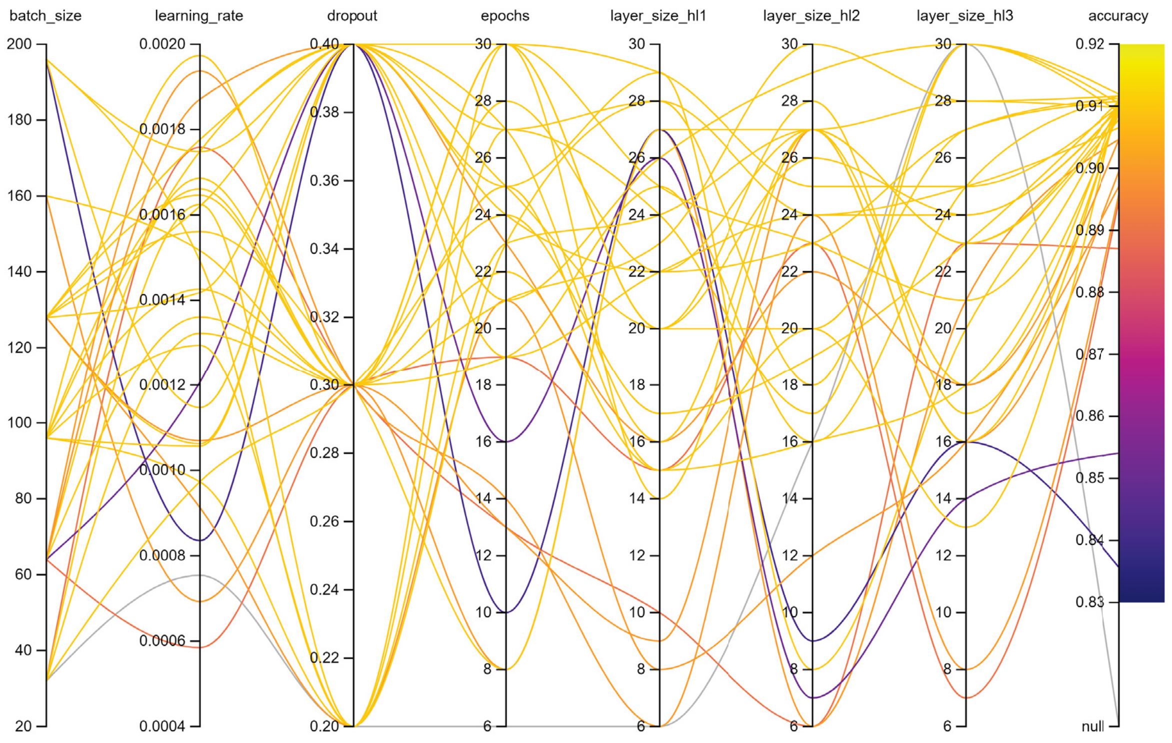
Considering the three ANNs trained so far, both ANNs shown in Figures 4.26(b) and 4.27 have advantages versus the other. The only problem with the ANN in Figure 4.26(b) is that 64QAM drops too much its accuracy for higher SNR values, and it seems to work properly for a range of SNR in which 16QAM does not work. This behavior also occurs in Figure 4.27, but in this case, there are more points in which both modulations can be classified with more accuracy.

Figure 4.27: Example of ANN accuracy trained with 10 dB to 20 dB of SNR



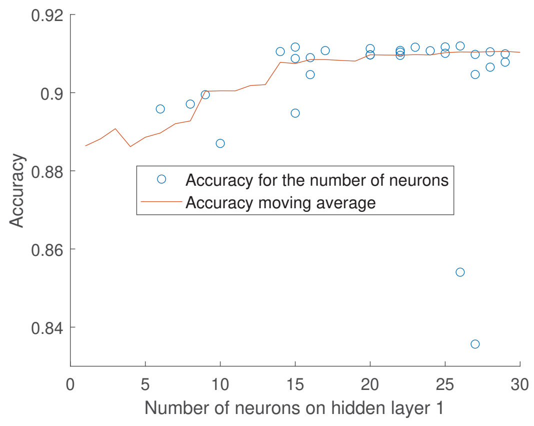
The best method to train the ANN is to use the dataset that gives the best accuracy for all modulations for a given SNR. The last trained ANN gives at least around 60% accuracy for all modulations above 13 dB of SNR. With the Weights and Biases website and Python library, a sweep was made using the hyperparameters shown in the columns of Figure 4.28. The Weights and Biases sweep algorithm has an optimization that selects the next iteration based on the ANN evaluation.

Figure 4.28: Hyperparameter sweep

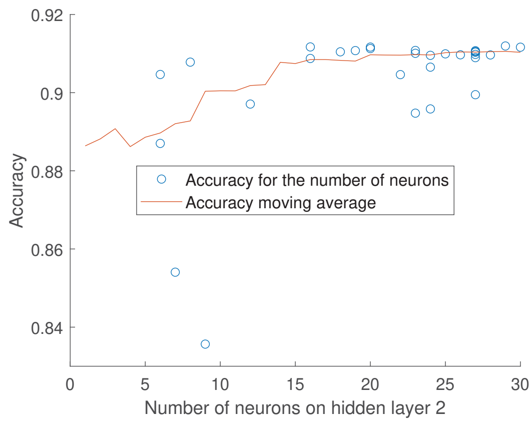


The columns of Figure 4.28 show the sweep range for each hyperparameter, and the colors identify the final accuracy. Figures 4.29(a), 4.29(b) and 4.30 show the accuracy obtained training a few ANNs with only the six best features selected before and dataset with only 10 dB to 20 dB of SNR. The x-axis for these graphics shows the number of neurons used for each hidden layer. Each circle in the graphic corresponds to a fully trained ANN with three hidden layers, but with the hidden layer in focus with the number of neurons determined by the x-axis.

Figure 4.29: Hidden layers 1 and 2 size

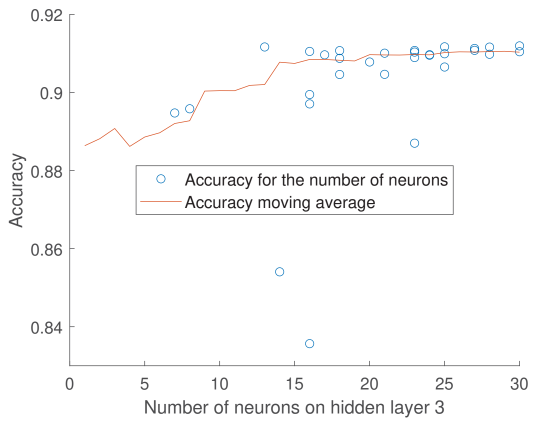


(a) Hidden layer 1 sweep



(b) Hidden layer 2 sweep

Figure 4.30: Hidden layer 3 sweep



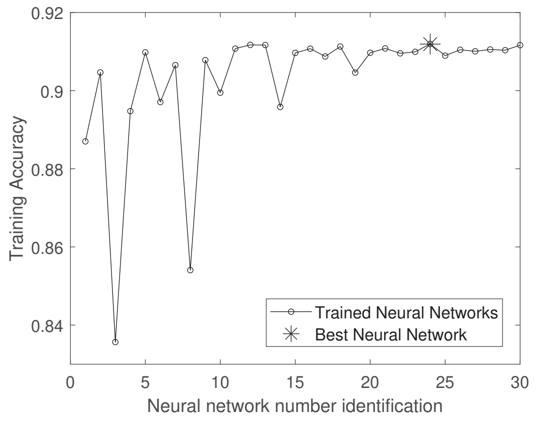
The moving average for the overall accuracy of the trained ANNs is also shown in these graphics. It is easier to see if adding neurons to hidden layers is optimal. It can be seen that with some neurons above 15, approximately for all figures, the accuracy of the ANN is almost optimal, not raising much after adding more neurons.

After finding the approximate optimal range to look for the number of neurons for each hidden layer, a series of ANNs were trained using another sweep of parameters, but this time trying to find the best number of neurons by layer from 15 neurons minimum to 30 at maximum.

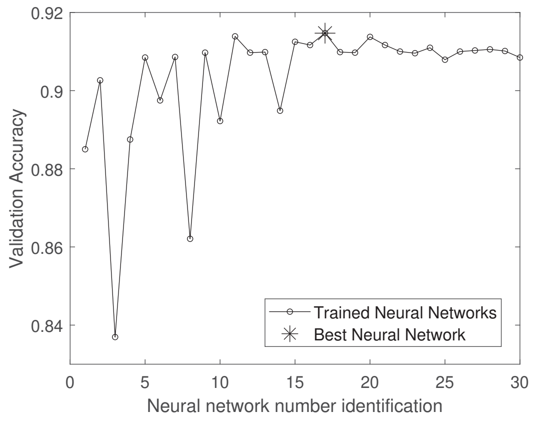
Figures 4.31(a) and 4.31(b) show the accuracy results for the training of a few ANNs. The x-axis identifies different ANNs for both figures. The most accurate ANNs on training and validation were identified with a star. It can be seen that the best ANN is not the same for training and validation. Also, the validation accuracy is higher than the training accuracy, which is not expected.

Figures 4.32(a) and 4.32(b) show training loss and validation loss, respectively. Just like the accuracy results, the validation loss is better than the training loss. It was expected that the validation accuracy and validation losses were worst than the training ones. Because from the dataset selected for the ANN training, actually 80% of this dataset was used for real training, i. e., updating weights and biases of the network, and only 20% was used to validate the ANN to make sure that there is no overfitting.

Figure 4.31: Accuracy for the ANN sweep

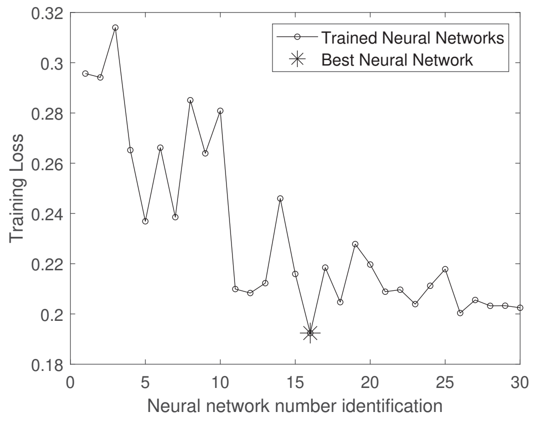


(a) Accuracy

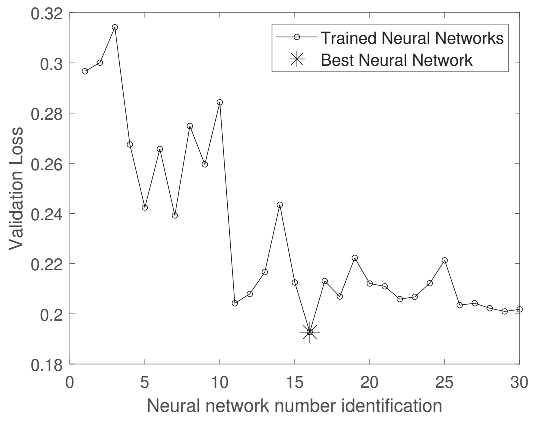


(b) Validation Accuracy

Figure 4.32: Loss



(a) Loss

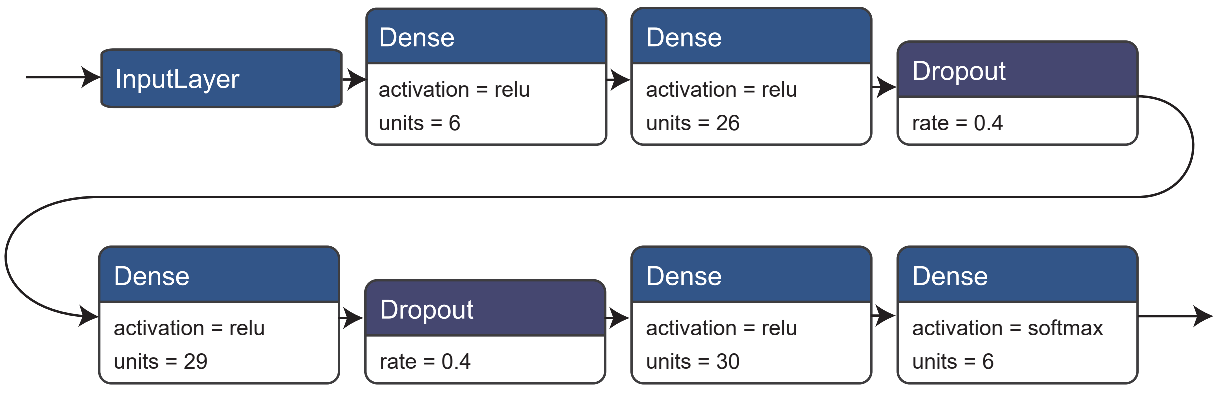


(b) Validation Loss

The structure found that fits the best the automatic modulation classification with six features is shown in Figure 4.33. The number of neurons and the activation functions is described together with the dropout rate used to avoid overfitting in the network. Converting this ANN to fixed-point numbers was done looking for the best Q-format number for the input, weights, biases, and outputs of every layer (Table 4.4.

The best Q-format number for quantization is the best range of values that fits the number to be quantized; the ranges are shown in Table 2.2. Since the quantization of a floating-point number takes out half of the bits used to give its precision, it is expected that the fixed-point number has a quantization error, and this error for the weights and biases of the ANN is shown in Table 4.5.

Figure 4.33: Final ANN structure



Figures 4.34(a) and 4.34(b) show the final results for this project based on the best ANN found in the hyperparameter sweeps. These figures compare the best ANN's accuracy and evaluated on PC versus the same network quantized and evaluated by the embedded system and using features calculated inside that system.

The errors found in calculating the features by the embedded system and the quantization error can explain why both graphics are not the same. Some overflow in the embedded system causes errors found in the feature calculations. Since both on the PC and on the embedded system the calculations are done with 32 bits floating point, these errors should not exist. Another explanation is that using the standard C language libraries may have lower precision on the embedded system when a numerical approximation is made.

The details of the Figures 4.34(a) and 4.34(b) are shown in Tables 4.6 and 4.7. The figures and tables were generated using data from one hundred frames by modulation and by SNR (ARM) and one thousand frames by modulation and SNR (PC). The absolute precision of the data presented is 1% for the embedded system and 0.1% for the PC.

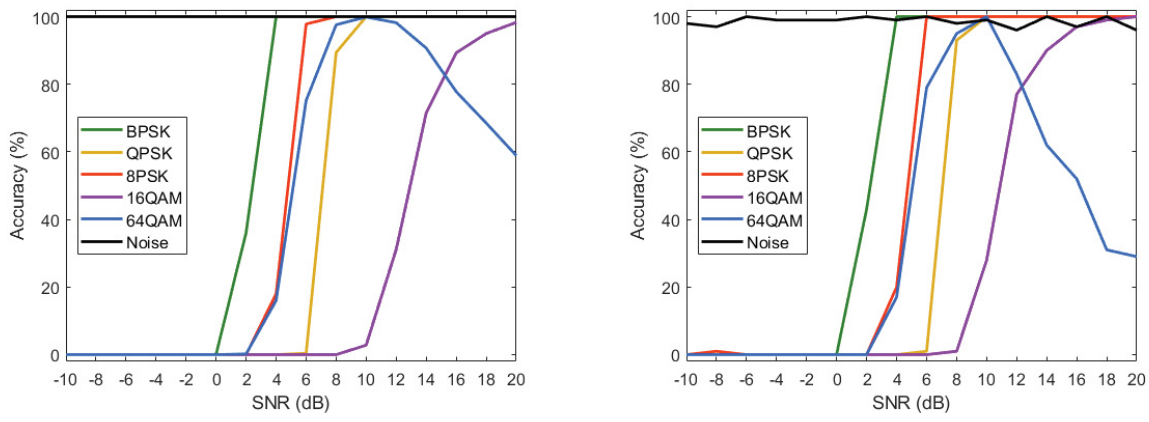
Table 4.4: ANN quantization

Location in the network	**Optimal Q number format**
Input	Q3.12
Layer 1 weights	Q2.13
Layer 1 biases	Q1.14
Layer 1 outputs	Q4.11
Layer 2 weights	Q1.14
Layer 2 biases	Q0.15
Layer 2 outputs	Q4.11
Layer 3 weights	Q0.15
Layer 3 biases	Q0.15
Layer 3 outputs	Q4.11
Layer 4 weights	Q0.15
Layer 4 biases	Q1.14
Layer 4 outputs	Q4.11
Layer 5 weights	Q3.12
Layer 5 biases	Q1.14
Layer 5 outputs	Q2.13

Table 4.5: Quantization error

Location in the network	**Maximum error**
Layer 1 weights	3.05e-05
Layer 1 biases	7.39e-06
Layer 2 weights	1.52e-05
Layer 2 biases	5.87e-06
Layer 3 weights	7.54e-06
Layer 3 biases	7.53e-06
Layer 4 weights	7.59e-06
Layer 4 biases	1.38e-05
Layer 5 weights	6.1e-05
Layer 5 biases	9.15e-06

Figure 4.34: Results comparison between PC and embedded system



(a) Accuracy results for best NN on PC

(b) Accuracy results for best NN on ARM

Table 4.6: Accuracy results for best NN on PC

	Signals accuracy					
SNR (dB)	BPSK	QPSK	8PSK	16QAM	64QAM	Noise
-10 to -2	0 %	0 %	0 %	0 %	0 %	100 %
0	0 %	0 %	0 %	0 %	0 %	100 %
2	36.1 %	0 %	0 %	0 %	0.2 %	100 %
4	100 %	0 %	18 %	0 %	16 %	100 %
6	100 %	0.4 %	97.8 %	0 %	75.2 %	100 %
8	100 %	89.4 %	100 %	0 %	97.6 %	100 %
10	100 %	100 %	100 %	2.8 %	99.9 %	100 %
12	100 %	100 %	100 %	31.1 %	98.2 %	100 %
14	100 %	100 %	100 %	71.5 %	90.7 %	100 %
16	100 %	100 %	100 %	89.3 %	77.8 %	100 %
18	100 %	100 %	100 %	95 %	68.5 %	100 %
20	100 %	100 %	100 %	98.3 %	58.9 %	100 %

Table 4.7: Accuracy results for best NN on ARM

	Signals accuracy					
SNR (dB)	BPSK	QPSK	8PSK	16QAM	64QAM	Noise
-10	0 %	0 %	0 %	0 %	0 %	98 %
-8	0 %	0 %	1 %	0 %	0 %	97 %
-6	0 %	0 %	0 %	0 %	0 %	100 %
-4	0 %	0 %	0 %	0 %	0 %	99 %
-2	0 %	0 %	0 %	0 %	0 %	99 %
0	0 %	0 %	0 %	0 %	0 %	99 %
2	43 %	0 %	0 %	0 %	0 %	100 %
4	100 %	0 %	20 %	0 %	17 %	99 %
6	100 %	1 %	100 %	0 %	79 %	100 %
8	100 %	93 %	100 %	1 %	95 %	98 %
10	100 %	100 %	100 %	28 %	100 %	99 %
12	100 %	100 %	100 %	77 %	83 %	96 %
14	100 %	100 %	100 %	90 %	62 %	100 %
16	100 %	100 %	100 %	97 %	52 %	97 %
18	100 %	100 %	100 %	99 %	31 %	100 %
20	100 %	100 %	100 %	100 %	29 %	96 %

The number of frames by modulation and SNR evaluated by the embedded system is ten times smaller than the PC version. The serial communication and validation of data take at least 4 seconds per frame. Every time something changes regarding the ANN, every tested frame needs to be tested again. Using 100 frames, six different signals, 16 different SNR values give 9600 frames that need to be sent and validated, approximately 10 hours and 40 minutes. Multiplying the result by 10 (that is, the entire dataset) is four days, 10 hours, and 40 minutes.

5 CONCLUSION

The correct generation of modulated signals with a wide range of SNR was crucial for analyzing the features and neural networks training. This analysis showed that, on average, values of each feature change according to noise and facilitated a selection of better features to be used in the final neural networks. Besides their numerical behavior, the analysis of computational use of the implementation of the features was an essential factor in eliminating the most expensive features that do not give a favorable result to the point of cost-benefit.

Also, the individual study of features was important to understand from which SNR value this feature becomes usable. Most features ended up showing an average behavior that favors the detection of modulations for high SNR values. Some features like Cumulant C21 and Gmax ended up giving horrible results, and it was not possible to identify the modulations used. This information can define an operating point for the device if it is turned into a product.

With the best-selected features and some trained neural nets, it was possible to observe the net's behavior for training with different SNR values. It was observed that training neural nets with much noise hinders their performance and confuses the net, causing the QAM modulations to be misinterpreted by the ANN and causing the hit rates to be misaligned. When 64QAM is classified with higher accuracy, 16QAM is not, and vice versa.

Supposing this behavior occurs for QAM modulations. In that case, it is expected that if the same neural network architecture and features were used to classify modulations with higher spectral efficiency, such as 256QAM and 1024QAM, the classification result of the network as a whole would be worse.

Also, training with low SNR causes the accuracy for high SNR to drop for the QAM modulations. It is not good, because as discussed before, it is expected that the device is receiving a high SNR modulation signal most of the time. After optimizing the neural network with only six features, its quantization was done, looking for the best Q formats representing the numbers found in the network weights and biases.

The quantization error is as small as possible because a simple algorithm was developed to look for the best Q-format representation that fits into the data that is being quantified. The final results show that using a feature-based modulation classification using neural networks can be done, expecting it to be as good as the PC version. Further work can be done looking for the quantization effects on the calculation of features and the impact on the neural network's accuracy.

The achievement of this paper is not its result regarding the accuracy found in the ANN or its implementation in an embedded device, but rather to show that after this first step many future topics can be followed. The first work that can be done on improving this is on a better selection of features and even the development of a feature with suitable characteristics to differentiate QAM modulated signals.

After this, a second possible work is to optimize the C implementation for more ARM processors, DSPs (Digital Signal Processors), and other embedded system architectures that exist on the market. The ARM architecture has the advantage of having a floating-point multiplier, which is not always the case for other processors. It would be interesting to research how to calculate features using only fixed-point numbers. With this, this implementation in embedded systems becomes broader, and at the same time, there is the difficulty of dealing with more quantization errors.

Finally, it would be interesting to research different neural network and machine learning architectures that can solve the modulation classification problem more simply and objectively, with their implementation in an embedded system to make a comparison.

REFERENCES

[1] Z. Zhu and A. K. Nandi. *Automatic Modulation Classification*. John Wiley & Sons, Ltd, New Jersey, USA, 2014.

[2] S. Tridgell, D. Boland, P. H. W. Leong, and S. Siddhartha. Real-time automatic modulation classification. In *International Conference on Field-Programmable Technology (ICFPT)*, pages 299–302, Tianjin, China, 2019.

[3] S. Kumar, A. Singh, and R. Mahapatra. Hardware implementation of automatic modulation classification with deep learning. In *IEEE International Conference on Advanced Networks and Telecommunications Systems (ANTS)*, pages 1–6, Goa, India, 2019.

[4] Y. Kumar, M. Sheoran, G. Jajoo, and S. K. Yadav. Automatic modulation classification based on constellation density using deep learning. *IEEE Communications Letters*, 24(6):1275–1278, 2020.

[5] 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*, 12(1):168–179, 2018.

[6] A Goldsmith. *Wireless Communications*. Cambridge University Press, 2005.

[7] Fa-Long Luo and Charlie (Jianzhong) Zhang. *Signal Processing for 5G: Algorithms and Implementations*. John Wiley & Sons, Ltd, New Jersey, USA, 2016.

[8] E. Khorov, A. Kiryanov, A. Lyakhov, and G. Bianchi. A tutorial on ieee 802.11ax high efficiency wlans. *IEEE Communications Surveys Tutorials*, 21(1):197–216, 2019.

[9] Proakis and Salehi. *Fundamentals of Communication Systems, 2nd Edition*. Pearson, New York City, USA, 2014.

[10] S. S. Haykin. *Digital Communication Systems*. Wiley, New Jersey, USA, 1 edition, 2013.

[11] A. K. Nandi and E. E. Azzouz. Automatic analogue modulation recognition. *Signal Processing*, 46(2):211 – 222, 1995.

[12] E. E. Azzouz and A. K. Nandi. Automatic identification of digital modulation types. *Signal Processing*, 47(1):55 – 69, 1995.

[13] E. E. Azzouz and A. K. Nandi. Procedure for automatic recognition of analogue and digital modulations. *IEEE Proceedings - Communications*, 143(5):259–266, 1996.

[14] B. Boashash. Estimating and interpreting the instantaneous frequency of a signal. i. fundamentals. *Proceedings of the IEEE*, 80(4):520–538, 1992.

[15] B. Kim, J. Kim, H. Chae, D. Yoon, and J. W. Choi. Deep neural network-based automatic modulation classification technique. In *International Conference on Information and Communication Technology Convergence (ICTC)*, pages 579–582, Jeju Island, Korea, 2016.

[16] A. Swami and B. M. Sadler. Hierarchical digital modulation classification using cumulants. *IEEE Transactions on Communications*, 48(3):416–429, 2000.

[17] M. R. Mirarab and M. A. Sobhani. Robust modulation classification for psk /qam/ask using higher-order cumulants. In *6th International Conference on Information, Communications Signal Processing*, pages 1–4, Singapore, 2007.

[18] V. D. Orlic and M. L. Dukic. Automatic modulation classification: Sixth-order cumulant features as a solution for real-world challenges. In *20th Telecommunications Forum (TELFOR)*, pages 392–399, 2012.

[19] M. W. Aslam, Z. Zhu, and A. K. Nandi. Automatic modulation classification using combination of genetic programming and knn. *IEEE Transactions on Wireless Communications*, 11(8):2742–2750, 2012.

[20] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. The MIT Press, Cambridge, USA, 2016.

[21] S. S. Haykin. *Neural Networks and Learning Machines*. Neural networks and learning machines. Prentice Hall, Hoboken, Nova Jersey, EUA, 2009.

[22] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958, 2014.

[23] R. Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, 58(1):267–288, 1996.

[24] A. E. Hoerl and R. W. Kennard. Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 12(1):55–67, 1970.

[25] S. J. Nowlan and G. E. Hinton. Simplifying neural networks by soft weight-sharing. *Neural Computation*, 4(4):473–493, 1992.

[26] X. Ying. An overview of overfitting and its solutions. *Journal of Physics: Conference Series*, 1168:022022, feb 2019.

[27] L. Al Shalabi, Z. Shaaban, and B. Kasasbeh. Data mining: A preprocessing engine. *Journal of Computer Science*, 2006.

[28] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[29] A. I. Abubakar, H. Chiroma, and S. Abdulkareem. Comparing performances of neural network models built through transformed and original data. In *International Conference on Computer, Communications, and Control Technology (I4CT)*, pages 364–369, Kuching, Malaysia, 2015.

[30] M. Shanker, M. Y. Hu, and M. S. Hung. Effect of data standardization on neural network training. *Omega*, 24(4):385–397, 1996.

[31] Sotiris B Kotsiantis, Dimitris Kanellopoulos, and Panagiotis E Pintelas. Data preprocessing for supervised leaning. *International Journal of Computer Science*, 1(2):111–117, 2006.

[32] Larry D. Pyeatt and William Ughetta. *ARM 64-Bit Assembly Language*. Newnes, 2020.

[33] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[34] L. Lai, N. Suda, and V. Chandra. CMSIS-NN: efficient neural network kernels for arm cortex-m cpus. *CoRR*, abs/1801.06601, 2018.

[35] R. Gong, X. Liu, S. Jiang, T. Li, P. Hu, J. Lin, F. Yu, and J. Yan. Differentiable soft quantization: Bridging full-precision and low-bit neural networks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019.

[36] ARM. *Arm Cortex-M7 Processor Technical Reference Manual*. Arm Limited, Nov 2018.

[37] ST Microelectronics. *STM32F7 Series and STM32H7 Series Cortex®-M7 processor programming manual*, 6 2019. Rev. 5.

[38] ST Microelectronics. *Dual 32-bit Arm® Cortex®-M7 up to 480MHz and -M4 MCUs, up to 2MB Flash, 1MB RAM, 46 com. and analog interfaces, SMPS*, 5 2019. Rev. 1.

[39] N. S. Alagha and P. Kabal. Generalized raised-cosine filters. *IEEE Transactions on Communications*, 47(7):989–997, 1999.

[40] L. Biewald. Experiment tracking with weights and biases, 2020. Software available from wandb.com.