Machine Learning for Radio Frequency Fingerprint Recognition

Katri Haapalinna

School of Science

Thesis submitted for examination for the degree of Master of Science in Technology.
Espoo 27.5.2024

Supervisor

Prof. Visa Koivunen

Advisor

TkT Maarit Melvasalo

Aalto University
School of Science

Aalto University
School of Science

Aalto University, P.O. BOX 11000, 00076 AALTO
www.aalto.fi
Abstract of the master's thesis

Author Katri Haapalinna

Title Machine Learning for Radio Frequency Fingerprint Recognition

Degree programme Master's Programme in Mathematics and Operations Research

Major Systems and Operations Research **Code of major** SCI3055

Supervisor Prof. Visa Koivunen

Advisor TkT Maarit Melvasalo

Date 27.5.2024 **Number of pages** 65 **Language** English

Abstract

In radio frequency fingerprint recognition, devices transmitting radio signals are identified based on their analogue imperfections. Although the identification has traditionally been performed by constructing models and recognising features in signals, the state of the art usually relies on machine learning. In data-driven techniques, the machine learning model extracts features of the signals and classifies them accordingly. Neural networks (NN) are a popular choice among machine learning classifiers.

Radio-frequency signals are inherently complex-valued, and thus it is often appropriate to apply complex-valued operations to them. More precisely, proper methods should be used for non-circular signals that have correlated real and imaginary parts. In-phase quadrature (IQ) imbalance is a signal impairment that creates non-circularity into a signal.

In this Master's thesis, radio frequency fingerprint classification is studied with the help of two different neural networks, that use real-valued or complex-valued signal processing, respectively. Non-circular radio frequency fingerprint data using IQ-imbalance is simulated for the classification. The differences of the performances of the two neural networks are examined, as well as their robustness with regard to the number of trainable parameters, i.e. the size of the NN, and the size of the training data. Based on the results, complex-valued neural networks offer robustness when classifying non-circular radio frequency fingerprint signals because they show less variation in classification accuracy when the NN size is changing. On the other hand, classification accuracy of real-valued neural networks is highly dependent on the NN size. Thus, complex-valued neural networks are recommended for classification of non-circular radio-frequency fingerprint data.

Keywords radio-frequency fingerprint, complex-valued neural network, machine learning, deep learning, neural networks, specific emitter identification, circularity

Aalto University
School of Science

Aalto-yliopisto, PL 11000, 00076 AALTO
www.aalto.fi
Diplomityön tiivistelmä

Tekijä Katri Haapalinna

Työn nimi Koneoppiminen radiosormenjälkitunnistuksessa

Koulutusohjelma Master's Programme in Mathematics and Operations Research

Pääaine Systems and Operations Research **Pääaineen koodi** SCI3055

Työn valvoja Prof. Visa Koivunen

Työn ohjaaja TkT Maarit Melvasalo

Päivämäärä 27.5.2024 **Sivumäärä** 65 **Kieli** Englanti

Tiivistelmä

Radiosormenjälkitunnistuksessa radiosignaalia lähettävät laitteet tunnistetaan analogisten epäideaalisuuksiensa perusteella. Vaikka tunnistus on perinteisesti perustunut malleihin ja signaalista erotettavien piirteiden suunnitteluun, käyttävät nykymenetelmät yleisesti koneoppimista. Datapohjaisissa menetelmissä koneoppimismalli sekä tunnistaa itsenäisesti signaalien piirteitä että lajittelee ne eri luokkiin. Neuroverkot (NN) ovat laajasti käytettyjä koneoppimispohjaisia luokittimia.

Radiosignaalit ovat luonnostaan kompleksiarvoisia, ja siksi on usein suositeltavaa käsitellä niitä kompleksiarvoisin operaatioin. Sopivia menetelmiä kannattaa käyttää erityisesti epäsirkulaarisille signaaleille, joiden reaali- ja imaginääriosat ovat korreloituneita. Samanvaiheis-kvadratuurin (IQ) imbalanssi on signaalihäiriö, joka lisää signaaliin epäsirkulaarisuutta.

Tässä diplomityössä radiosormenjälkien luokittelua on tutkittu kahden, reaali- ja kompleksiarvoista signaalinkäsittelyä hyödyntävän, neuroverkon avulla. Epäsirkulaarista, IQ-imbalanssia sisältävää radiosormenjälkidataa on simuloitu luokittelua varten. Eri neuroverkkojen suoriutumisten eroja sekä niiden robustisuutta opetettavien verkkoparametrien lukumäärän eli neuroverkon koon sekä opetusdatan koon suhteen on analysoitu. Tulosten perusteella kompleksiarvoiset neuroverkot ovat epäsirkulaarisia radiosormenjälkisignaaleja luokitellessaan reaaliarvoisia robustimpia, sillä verkkoparametrien lukumäärän muutos vaikuttaa niiden luokittelutarkkuuteen vain vähän. Reaaliarvoisilla neuroverkoilla taas luokittelutarkkuudet riippuvat suuresti valitusta neuroverkon koosta.

Avainsanat radiosormenjälki, kompleksiarvoinen neuroverkko, koneoppiminen, syväoppiminen, neuroverkot, lähetinlaitteen tunnistus, sirkulaarisuus

Preface

I would like to express my warmest gratitude to all who have supported my completion of this thesis. Much obliged.

Tämä diplomityö on tehty Aalto-yliopiston sähkötekniikan korkeakoulun informaatio- ja tietoliikennetekniikan laitoksella professori Visa Koivusen tutkimusryhmässä. Kyseiseen ryhmään päädyin aikoinaan kohdattuani sattumalta vappuna 2019 Ullanlinnanmäen laella TkT Maarit Melvasalon, joka sittemmin ohjasi diplomityöni. Suuri kiitos Inulle kärsivällisestä kysymyksiin vastaamisesta, rohkaisusta työn haasteissa sekä perusteellisesta paneutumisesta sen aihepiiriin.

Tahdon kiittää myös valvojaani Visaa, joka jaksoi kerran toisensa jälkeen perehtyä luonnoksiini samalla lukemattomia hyödyllisiä kommentteja antaen. Matematiikan ja systeemianalyysin laitoksen professori Ahti Salollekin haluan osoittaa kiitokseni korjausehdotuksista sekä työtä kohtaan osoitetusta mielenkiinnosta.

Kiitos myös koko tutkimusryhmälle opastuksesta akateemisen maailman saloihin sekä mukavista rupattelu- ja lounashetkistä.

Lopuksi kiitän vielä ystäviäni vertaistuesta läpi opintojen sekä perhettäni ja sukulaisiani kuuntelusta ja kannustuksesta.

Otaniemessä 21.5.2024

Katri Haapalinna

Contents

Symbols and abbreviations

Symbols

lower case letter in **bold**	Vector
capital letter in **bold**	Matrix
$\mathbb{E}()$	Expectation value
σ^2	Variance
τ	Pseudovariance
$j = \sqrt{-1}$	Imaginary unit
$\bar{z}$	Complex conjugate of variable z
^{T}	Transpose
^{H}	Hermitian transpose
ϱ	Circularity quotient
∇	Gradient operator
$L()$	Loss function
η	Learning rate
γ	Learning rate decay
p_{val}	Patience for validation
n_{max}	Maximal number of epochs
λ	Kernel length for one-dimensional convolution
p_{pad}	Padding for convolutional layers
s	Stride for convolutional layers
β	Batch size

Abbreviations

AM	Amplitude Modulation
AWGN	Additive White Gaussian Noise
BPSK	Binary Phase Shift Keying
CES	Complex Elliptically Symmetric
CFO	Carrier Frequency Offset
CNN	Convolutional Neural Network
CReLU	Complex Rectified Linear Unit
CUSUM	Cumulative Sum
CVNN	Complex-Valued Neural Network
DAC	Digital-to-Analog Converter
DL	Deep Learning
DWT	Discrete Wavelet Transform
FC	Fully Connected
FFT	Fast Fourier Transform
GAN	Generative Adversarial Network
GLRT	Generalised Likelihood Ratio Test
GPS	Global Positioning System
GPU	Graphics Processing Unit
I/Q	In-phase/Quadrature
IF	Intermediate Frequency
IMEI	International Mobile station Equipment Identity
IoT	Internet of Things
IP	Internet Protocol
k-NN	k-Nearest-Neighbour
LO	Local Oscillator
LOS	Line Of Sight
LPD	Low Probability of Detection
LPI	Low Probability of Intercept
LSTM	Long Short-Term Memory
MAC	Media Access Control
MAE	Mean Absolute Error
ML	Machine Learning
MLP	Multilayer Perceptron
MSE	Mean Squared Error
NAC	Network Access Control
NLP	Natural Language Processing
NN	Neural Network
OFDM	Orthogonal Frequency Division Multiplexing
OQPSK	Offset Quadrature Phase Shift Keying
PA	Power Amplifier
PCA	Principal Component Analysis
PG	Percent Good
PNN	Probabilistic Neural Network
PSK	Phase Shift Keying
QAM	Quadrature Amplitude Modulation
QPSK	Quadrature Phase Shift Keying

RACH	Random Access Channel
ReLU	Rectified Linear Unit
RF	Radio Frequency
RFF	Radio Frequency Fingerprinting
RFFI	Radio Frequency Fingerprint Identification
RNN	Recurrent Neural Network
RVNN	Real-Valued Neural Network
SDR	Software-Defined Radio
SEI	Specific Emitter Identification
SGD	Stochastic Gradient Descent
SNR	Signal-to-Noise Ratio
SOM	Self-Organising Map
SPRT	Sequential Probability Ratio Test
STFT	Short-Time Fourier Transform
SVM	Support Vector Machine
UAV	Unmanned Aerial Vehicle
UMTS	Universal Mobile Telecommunications System
WiFi	Wireless Fidelity
wrt	with reference to

1 Introduction

In the digitalised world of today, there are more and more electronic devices connected to different communications networks. What's more, there is also an increasing number of wireless networks, such as wireless fidelity (WiFi), 4G, and 5G, to which radios connect. Since wireless communications take place over the electromagnetic spectrum, they are subject to various nonidealities and propagation effects. Consequently, they are not as controllable as with the wired case in which all signals come from known ports. Thus, a concern arises as one may not be sure whether the transmitting devices in a wireless network are those one is willing to accept. The issue is a network access control (NAC) problem. Hence the emitters need to be authenticated. Transmitters include e.g. mobile phones from different manufacturers, access points, base-stations, tablets, vehicular radios, localisation systems, satellites, software-defined radios (SDR), and unmanned aerial vehicles (UAV). [24]

Recently, Global Positioning System (GPS) jamming by Russia in Finland and other European countries has been a timely problem, see e.g. [35]. To overcome this issue in general, transmissions from deceptive jammers should first be detected. Navigation receivers such as GPS should also be able to determine whether the navigation signal is coming from one of the intended satellites or it is produced by a disturbing jammer. Consequently, the signal emitted by the jammer should be identified and distinguished from the satellite signal. This is another application for transmitter identification. [7]

In order to identify and suppress adversarial transmitters that pose a threat for a network, it is often required that the devices register via some access mechanism, for example password or another secure key. For instance, the messages in the network may be encrypted unless a user is an authorised one that possesses a key to decipher them [70]. On the other hand, the network may require authentication by its users. For this, passwords are commonly used. Besides, there exist identifiers, addresses, or labels that are unique to a device such as Internet Protocol (IP) address, Media Access Control (MAC) address, and International Mobile station Equipment Identity (IMEI) number, and that can thus be used as means for authentication as well. However, there are several issues concerning these methods. Encryption may end out unpractical, laborious and expensive if a separate cryptographic module is needed in a device [17]. Moreover, the devices may be unable to manage passwords due to their inexpensive character and large number of operations, as is the case with some Internet-of-things (IoT) equipment. In a large-scale IoT especially, where the number of devices is large, password and key management would be a demanding problem. Besides, there is a danger that publicly available IP or MAC addresses, or IMEI numbers, will be copied and thereby faked by malicious network users that will thus pretend to be someone else [8]. Hence, a more secure way to identify transmitting devices is needed [90],[63],[8].

Another solution for the above described network access control problem is to identify different transmitters without an active authentication with passwords, or easily spoofed addresses. That is, the devices should be recognisable by the network when they transmit, and the means of recognition should not rely on properties that

can be easily faked. Hence, some factors within an emitter that are inherent, stable, and unique to it should be employed. This is where radio-frequency (RF) fingerprints, that can be seen as equivalents of biometric fingerprints that identify humans, can be extracted, identified, and recognised for transmitting devices. Typically such fingerprints are associated with a variety of nonidealities or imperfections present in any implemented real-world radio frequency device.

RF fingerprints are applicable to a wide range of applications. They are used for recognising radios in securing wireless networks and IoT, for wireless localisation, for spectrum monitoring, for detecting malfunctioning devices, and for signal intelligence in electronic warfare systems. Indeed, the first such methods were to recognise radars in the time of the Vietnam War [8]. The fingerprints are difficult to imitate, and the identification method based on them can thus detect malicious, unlicensed network users. Tasks that may be performed with this means also include validation of network actions via the connection between an event and the device behind it, and tracking of the functioning of a transmitter when its normal operation is known [17].

Another highly important application for RF fingerprints is user localisation. Alterations in propagation path of the signal can be utilised in order to determine the location of the emitted signal [86], whereas location-invariant transmitter impairments are employed in emitter identification. In this thesis, the interest lies on the latter. Similarly, radar target recognition deals with its own fingerprints such as high resolution range profiles consisting of target radar cross section and its scatterers as well as micro-Doppler signatures [40].

Numerous techniques can be applied for RF fingerprinting. Fingerprints are found from the received signal by retrieving a certain collection of properties from it. The process of extracting the fingerprint can be performed either by designing and combining features, or by employing data-driven methods. For fingerprint classification, e.g. supervised machine learning using neural networks or more classical statistical classification methods can be used to find out the origin of the signal.

1.1 Scope of the thesis

This thesis studies the RF fingerprint classification task using neural networks (NN). Two different NN classifiers, one using real-valued and the other employing complex-valued calculus, are trained with simulated RF fingerprint data, and their classification results are compared. The fingerprints are assumed to originate from the transmitter hardware, including radio frequency front-ends and circuitry, power amplifiers, and transceiver structure. The performances of the NN classifiers with regard to varying network or training data size are examined. The problem of data acquisition, creation of features, as well as radio-channel-based fingerprints utilised in wireless localisation, are out of scope of the thesis.

1.2 Contributions of the thesis

For this thesis, RF fingerprint data that uses IQ (in-phase quadrature) imbalance as an impairment, as presented in [82], was simulated. A real-valued convolutional neural network (RVNN) was implemented using Pytorch modules and trained with the simulated data. A complex-valued convolutional neural network (CVNN) presented in [42] was trained with the same data. Experiments were run to study the performances of the NNs using the simulated test data. Finally, the obtained results were examined and analysed.

The thesis offers novelty in terms of conducting RF fingerprint classification using the complex-valued neural network that involves complex-valued differentiation using Wirtinger calculus. As a result, it can be stated that the complex-valued NN is more robust than the real-valued NN in classifying non-circular RF fingerprint signal samples. The complex-valued NN provides stable performance with regard to changing NN size, whereas the RVNN is highly dependent on the right choice in NN size. Consequently, CVNN has better classification accuracy than RVNN at high signal-to-noise ratio (SNR) regime when averaged over NN sizes. As expected, both NNs show improvement when the amount of training data is increased, but in practical scenarios, the data is often limited.

1.3 Structure of the thesis

This Master's thesis is organised as follows. Some background concerning radio-frequency fingerprints, methods for finding them, and classification of them, along with a short survey about publicly available fingerprint data, are covered in Section 2. An important and widely used technique within the subject, machine learning, is presented in Section 3 in a neural-network-oriented manner. A separate section, Section 4, is dedicated to an introduction to complex-valued neural networks and their underlying calculus principles. The methods and data employed in this thesis are presented in Section 5, and the simulations conducted with their results in Section 6. Finally, a conclusion is drawn in Section 7.

2 Overview of RF fingerprinting

2.1 Radio-frequency fingerprints

Radio-frequency fingerprints, or radiometric signatures, employed in radio-frequency fingerprinting (RFF), are properties that allow identification of transmitting wireless devices based on the differences in their structure and implementation of specific features [19], [8]. The process of recognising transmitters may also be called e.g. emitter identification, as in [24], specific emitter identification (SEI) as in [12], or physical-layer device identification, as has been done in [10]. Surveys covering the field of RF fingerprint studies include [72] and [24].

Radio-frequency fingerprinting is also used in wireless localisation where the location, rather than identity, of the emitter is to be found. In such applications the fingerprint refers to the channel impulse response or received signal strength from a particular location to a base station or access point, for example in a WiFi or 4G or 5G cellular communication system. This approach is best suitable in indoor localisation where it is simpler to produce representative training data for supervised learning from different parts of a building. Such training data would contain location-RF-fingerprint pairings for a large number of locations in the operational environment. In outdoor scenarios, localisation would require collecting and labeling massive amounts of training data. This may be done by wireless operators that are able to acquire such data over long periods of time. Another application that follows similar principles is radar target recognition. In this case, the corresponding channel of the radar target serves as a fingerprint. The use of RF fingerprints in localisation, nor in radar target recognition, is not further addressed in this thesis. See [86] and [40], respectively, for more detailed information and additional references.

The RF fingerprints that are of interest in this thesis are based on the fact that transmitters have distinct features and unique non-idealities, often called impairments. Even though the differences are more significant between manufacturers and their transceiver implementations, devices that have same supplier and model variate as well. In practice the physical components and the systems they form are never absolutely identical. Due to the manufacturing process of the various analogue parts of the hardware, there always occurs differences between components that are supposed to be alike. However, tolerances set for this variation are strict, and the fluctuation ranges for component properties are therefore small, if not tiny. The less expensive a piece of hardware is, the more there are impairments and variation in the device. Thus, there always exists more or less variation among in principle similar devices as well. Besides, the functionalities of the components alter differently when the components age, which increases the differences between devices even more. Temperature, and in battery-operated devices, the charge level of the batteries, impact behaviour of the components and hence may further increase the differences. Also if a high transmit power is used, the differences may become more significant, especially those based on the amplifiers.[81]

It may be challenging to find out the root cause of impairments accurately. Components, the imperfections of which have been thought to form fingerprints

include frequency synthesizer or local oscillator (LO), antennas, charge pumps, oscillators, mixers, power amplifiers (PA), digital-to-analog converters (DAC), and various filters [10],[82],[90]. The receiver's analogue front-end structure, involving direct-conversion architecture, low-Intermediate Frequency (IF) architecture, or superheterodyne principle, impacts the nonidealities and various imbalances as well [82]. The way the components are connected may also play a role. An example of component configuration is presented in Figure 2.1. When the devices and their circuits are complicated enough, the impairments cannot be tracked with same precision. Thus component groups, e.g. modulator or analogue subcircuitries, are known to contain relevant imperfections for RFF without specifying the individual component. [10] The trend has been to integrate all the needed circuits to the same chip, and hence it may be even more difficult to point out the specific circuit causing the impairment.

The actual impairments that have previously been exploited with RFF in the literature include e.g. local oscillator imperfections such as carrier frequency offset (CFO), or mixer-related in-phase (I) and quadrature (Q) imbalance, or even power amplifier nonlinearity. Impairments occur for instance when components overflow, or batteries cause delays. For instance, I and Q signals may leak to one another if the wires are too close to each other. Carrier frequency offsets depend on the oscillator clock imprecisions that are usually fixed by synchronisations, but they may also be caused by relative radial mobility leading to Doppler shift. Figure 2.1 illustrates common signal impairments such as quadrature errors, self-interference of signals, and amplitude clipping or saturation of signals. However, the actual form of the impairments and their effect vary based on the transmitter structure. It may not be necessary, though, to thoroughly comprehend the root cause of functioning of impairments. This is because the fingerprint comprises a combination of several imperfections, and it may also be used as such. There is such a large number of sources of impairment that rigorous modelling of them with mathematical equations may be tedious. Hence, there is a modelling deficit and learning directly from data becomes an attractive option. [90]

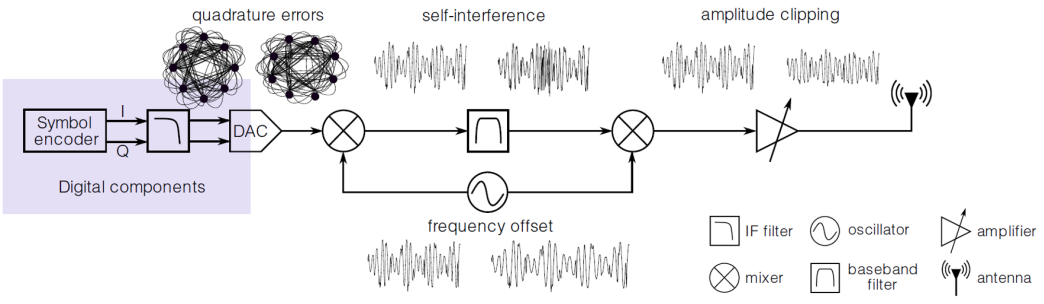

Figure 2.1: Transmitter blockchain and common impairments. DAC: Digital-to-Analog Converter, IF: Intermediate Frequency. Source of the figure: [8]

The fingerprints should be such that it would be possible to extract them from

every device, and they should be unique, invariant with respect to time and environment, and measurable [72]. For instance, location-dependent properties such as radio channels do not make appropriate fingerprints when the goal is to identify transmitters as they vary with respect to environment. Besides, not even all hardware impairments are suitable for utilising in RF fingerprints. As LO imperfections lack stability and predictability in time and temperature changes, they provide no useful properties for the RFF and should be compensated before the fingerprint extraction. Instead, phase and gain imbalances and PA nonlinearities are recommended for employing in identification. The different hardware impairments utilised in RF fingerprints are compared in more details in [90].

An RF fingerprint of a device is found when analysing the signals emitted by it. Signals are, for example, the waveforms used in radars or information bearing communication signals transmitted over the electromagnetic spectrum in radio frequencies. The fingerprint can be seen as an alteration in the ideal signal when it is transmitted through emitter hardware. As this change is inherent to a transmitting device, it allows for perceiving the differences between signals from different emitters and thus for extracting the fingerprints. Finally, fingerprints enable identification of the device that transmitted it.

When examining over-the-air signals, they must be both transmitted and received. This poses a challenge for RF fingerprint identification since receivers have their own impairments that affect the signal as well, even though only transmitters and their imperfections are of interest in this case. However, there is no exhaustive research about the effect of receivers on the RFF [90].

The procedure of RF fingerprinting can be subdivided into smaller tasks:

- Data collection
- Labelling of the data
- Processing into fingerprints
- Classifier training
- Classification of signals

First, data is collected by receiving transmissions from the emitter of interest. Labels are attached to the data such that the emitter of each signal is known. The received data is then processed to reveal the distinct features needed for identification or processed so that it can be fed to a classifier such as a neural network or feature-based classifiers. The classifier is trained using supervised learning principles and labelled training data to identify transmitters. Thereafter, newly received signals may be processed, and when used as input to the classifier the unknown transmitter may be recognised. The upcoming subsections 2.2 and 2.3 will contain more details on the subject. The basics of supervised machine learning (ML) often used in classification are covered in Section 3.

2.2 Finding RF fingerprints

There are two different types of fingerprints. The form of the fingerprint is either a selection of characteristics calculated from the received signal, called *features*, or a sequence of raw observed data. There are several means to perform the extraction of RF fingerprints. The simplified hierarchy of different techniques is illustrated in Figure 2.2.

Feature-based methods require domain knowledge in engineering the features. This way, classification can be done based on methods typically used in classical pattern recognition. Furthermore, the features are divided into two separate groups based on the part of the signal they are calculated from, that is, transient or steady-state, as shown in Figure 2.2. There are also various kinds of features, e.g. transforms into frequency domain, filter outputs, parametric models, basis function expansions, statistical descriptors, and errors with reference to an ideal signal. The feature-based methods are more thoroughly discussed in the following subsection.

The raw-signal-based methods are nowadays a more common approach in RFF due to the progress in deep learning using neural networks, which in turn has been enabled by the increase of computational power in the recent years. The term "raw signal" refers here to time-domain complex-valued IQ samples. They may have been preprocessed such as normalised or whitened. Moreover, noise attenuation or outlier rejection may also have been performed to the samples that are called "raw" in this thesis.

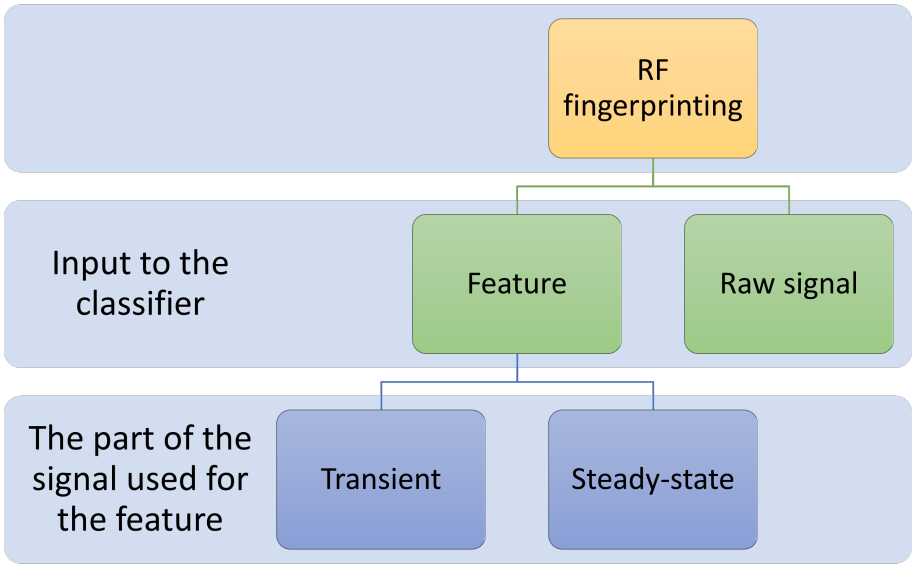

Figure 2.2: A simplified presentation of different kinds of RF fingerprinting techniques.

2.2.1 Feature-based methods

Originally, a radio-frequency fingerprint usually involved features extracted from the signal. A feature-based RF fingerprint is comprised of one, or usually more, features.

If multiple features have been extracted, they are stacked to form a feature vector. A feature vector is computed from received data and classified using machine learning, such as pattern recognition methods or neural networks. Features are commonly designed by engineers with deep understanding of transceiver structures, and their implementations, RF circuits, and nonidealities. In these feature-employing RFF procedures, transmitters are identified based on their corresponding values of their features, and either neural network output or closest distance to different transmitter classes in the feature space.

Traditionally, either transient or steady-state part of the signal may be used to find a feature. It is possible to compute features from intermediate frequency (IF) of the signals when the signal is moved from carrier frequency to baseband frequency via IF. IF nonidealities are nevertheless not that well visible at longer distances since they tend to have weak power. Still, they are a very rich source of features since IF implementation varies a lot among device types.

A transient signal is observed during the activation of the transmitting device. It is found in a short time interval usually in the order of μs or ms. It is a typical time interval in which the transmission power increases until it attains the actual transmission level. Simultaneously, the frequency synthesizer of the transmitter finds the right frequency for transmission. [27],[76],[79] An example of a transient is illustrated in Figure 2.3. The transient in the picture is also called turn-on transient in order to distinguish it from the end transient related to the turn-off of the transmitter [27]. The amplitude may change abruptly or gradually, which affects the form of the transient signal and also has an effect on the appropriate methods for transient extraction [19]. Before the extraction, the beginning of the transmission needs to be detected so that the collection of data and computation of features may be performed. That is, there is a hypothesis testing task where a decision is made about whether there is signal present or noise only.

The features are then computed from the transients. It has been practical to study transients, since they occur every time for every transmitter as it is turned on, and are thus independent of the actual signal that is being transmitted. RF fingerprinting studies that utilise transients are for instance [81], [19], [21], [9], [20], and [76].

When calculating features from a transient and forming the fingerprints, the major challenge is to correctly locate the transient signal on temporal domain. That is, the presence of signal is first detected and the beginning of the leading edge of it is identified next. It must be known at which time point the transient starts, in order to ensure that one really is calculating features from the transient, and not from channel noise or steady-state signal only. Otherwise there is a risk of accidentally developing the fingerprint of either noise or signal, which would eventually lead to incorrect classification of the transmitters. Even though noise as completely random data sequence would be very far from any transmitter class centre in the feature space, the classifier would still associate it to the closest one in the feature space unless there is a separate class for noise samples. When the starting point of a transient has been found, the searching for the end point is substantially a more straightforward task. The end point has been found experimentally in [19]. Nevertheless, in order

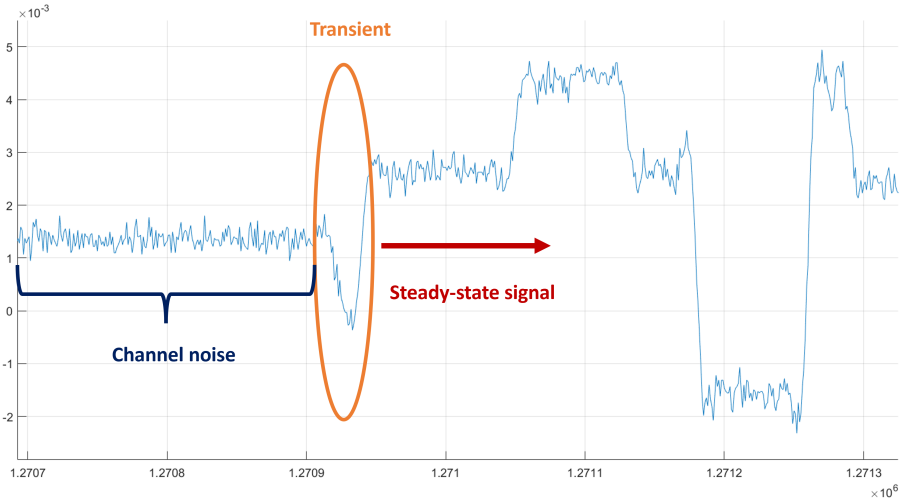

Figure 2.3: Part of interception in I-branch of a receiver when transmission of a 8-phase-shift keying (8PSK) signal starts, illustrating channel noise, transient, and steady-state signal.

to accurately find the transient, high sampling rates are required for the receiver, especially as the length of the transient may only be microseconds [27].

There are multiple methods to find the point where the transient starts. The change between the originally received channel noise and the transient signal can be either gradual or abrupt, and this affects the appropriate method for extracting the location of the transient from the obtained samples. If the transition between noise and signal is abrupt, for example Variance Fractal Dimension Trajectory [67] or Bayesian Change Point [79] detectors, or in general sequential detectors, including sequential probability ratio test (SPRT), change-point detection, and cumulative sum (CUSUM) test, may be applied. The former models the signal as a sequence of local variance dimensions, that is, fractality, of windows taken of the signals, and finds the start of the transient by a change in the sequence. When it comes to the latter, the method relies on the assumption that the distributions of noise and transient samples be different, and employs Bayes' formula to analyse them. On the other hand, if the signal starts gradually and the increase in amplitude is smooth, the former might not find the transient accurately enough. In this case, for example Bayesian ramp change detector, as in [80], or detection via phase characteristics, as in [19], could be used along with the Bayesian Change Point detector.

Feature-based RF fingerprints may also be calculated from the steady-state signal instead of the transient. The steady-state signal, i.e. the actual transmitted signal that starts immediately after the turn-on transient, and analogously ends before the turn-off transient, carries the useful information, for example payload data in wireless communications or radar pulse waveform in radar. Steady-state signals are utilised in RFF in e.g. [17] and [27]. When using this approach, there is in fact no need to separate the actual steady-state signal and transients as the algorithms are not dependent on finding a specific time interval, even though the features may be

different in the transient phase. However, this approach requires that a specified signal be transmitted from all the devices in order to ensure that the differences observed do not originate from different transmissions. For this purpose, for example Universal Mobile Telecommunications System (UMTS) Random Access Channel (RACH) and Ethernet frame preambles can be utilised [27],[17]. Also known pilot signals, used by most communications systems for channel estimation and synchronisation purposes, could be used this way.

The nonidealities used for RFF are present regardless of the transmitted data, even though they might be difficult to observe. The useful payload signals may either be random, such as communications data, or deterministic, e.g. a radar waveform from a certain code family or a known pilot signal. Thus, there are no restrictions concerning the data itself, as long as the signals are similar to each other. If different signals were fed to the same RF fingerprint classifier, they would have different statistical and structural properties which would lead to classification of signals based on their modulation instead of emitter. Furthermore, modulation type might impact the use of transmit power if the amplitude is modulated along with phase, such as in orthogonal frequency division multiplexing (OFDM), and amplifier-related nonidealities might be better revealed [56],[3]. It is typically advisable to select the modulation type such that the frequency, amplitude, or phase of the signals changes rapidly, as this often improves the visibility of the fingerprints.

After the desired signal portion, that is, either transient or steady state, has been obtained, suitable features are extracted from it. Based on expert knowledge about transmitter structure, RF circuitry, antenna systems, and their nonidealities, the features are designed so that they vary along those nonidealities. There are several methods to compute features, and also different types of features. Many features can be obtained from the signal spectrum. For instance, the signal may be transferred to frequency domain via Fast Fourier Transform (FFT) or Discrete Wavelet Transform (DWT) as in [27], or [9] and [6], respectively. Even bispectrum, that is, third-order spectrum of the signal [46], can be estimated as in [12]. Possible power-spectrum-related time-frequency transforms that do not apply phase information include spectrogram and Choi-Williams distribution. Besides spectra, also amplitude characteristics, such as instantaneous amplitude, may be calculated, see [81]. Both DWT and amplitude characteristics may be employed in addition to phase information as well, as in [20]. Also various kinds of error metrics, i.e. differences between the received, impaired signal and an ideal signal, can be calculated and considered as features, see [8] and Table 2.1. The features may even consist of matched filter outputs as in [17]. As nonidealities often appear as nonlinearities, and e.g. FFT is a linear operation, one may need to include some features and their computational methods that are sensitive to nonlinearities in the feature selection. Different features are collected in Table 2.1.

When the features have been extracted, the feature vector, that is, the RF fingerprint, is ready for classification. Naturally, the ability to classify it requires that the classifier is already trained, i.e. it has been presented fingerprints similarly built along with the correct labels. The training phase is essential since the classifier should generalise well and not overlearn the training set. After the training, decision

Table 2.1: Different features for feature-based RF fingerprinting. $I(t)$ and $Q(t)$ are I and Q branches of the signal, respectively. Other features include basis function expansion, short-time Fourier transform (STFT), variational mode decomposition, and fitting polynomial to power level. The references column shows applications of the features in RFF.

Feature	Computing Method	References
Spectral Components	FFT	[27]
Wavelet Coefficients	DWT	[9], [6], [20]
Amplitude Profile	Hilbert Transform	[81]
Instantaneous Amplitude	$\sqrt{(I(t))^2 + (Q(t))^2}$	[20]
Instantaneous Phase	$\tan^{-1}(\frac{Q(t)}{I(t)})$	[20]
Phase Error	Difference wrt ideal angle	[8]
Magnitude Error	Difference wrt magnitude	[8]
Error Vector Magnitude	Difference wrt ideal vector	[8]
I/Q Origin Offset	Difference wrt ideal I/Q Origin	[8]
Frequency Error	Difference wrt ideal carrier frequency	[8]
SYNC Correlation	Correlation wrt ideal synchronisation signal	[8]
Closeness	Matched Filter	[17]
Bispectrum	Bispectrum (3rd-order spectrum) [46]	[12]

boundaries for each class are found such that along the boundary of classes a and b, the probability of a sample to belong to a is equal to its probability of belonging to b. Later, the decision boundary may be utilised to determine the amount of sufficient features discussed below, and to also find these features. [36]

Traditionally, when designing feature-based fingerprints for different transmitters, it has been advisable to select an appropriate number of features. At least if limitations in memory or computing capacity available raise an issue, it is beneficial to reduce the dimensionality of the feature vector, that is, the number of different features in a fingerprint. In order to eliminate highly correlated features, one could apply e.g. principal component analysis (PCA). However, it is crucial not to perform too sparse a selection, as the features essential for distinguishing the different transmitters should naturally remain in the vector. [81] On the other hand, features that vary a lot within the same emitter complicate the classification and should be removed [20]. Methods for the selection process of features utilised in radio-frequency fingerprint identification (RFFI) include PCA [81], Euclidean distance and unsupervised learning such as k-nearest neighbour (k-NN) clustering [20], and genetic algorithms [76]. Alternatively, feature selection may be performed with the help of probabilistic neural networks (PNN), even though they may not have been utilised in it in the context of RF fingerprints [23]. Algorithms for feature selection have also been collected in Table 2.2.

Table 2.2: Different feature selection algorithms.

Feature selection method	Reference
Principal component analysis (PCA)	[81]
Euclidean distance and clustering	[20]
Genetic algorithms	[76]
Probabilistic neural network (PNN)	[23]

2.2.2 Data-driven methods using raw signals

The recent research no more relies on feature-based identification, that is, methods that require tedious engineering of suitable features prior classification. Instead, observed signal sequences are fed to a classifier as they are, or right after ad-hoc preprocessing operations such as normalisation operations and outlier rejection. Thus, the RF fingerprint equals a piece of raw signal instead of a list of features extracted from it. This is feasible since the modern classifiers based on deep neural networks allow vaster amounts of input data, as well as more parameters when compared to former years due to the recent growth of computational power. In the classification, different ML techniques can be exploited. A survey including discussion about deep-learning-based RF fingerprinting is found in [24].

As the raw-signal-based methods use as the fingerprint the piece of raw signal itself, only the classification step is needed and thus the methods are more thoroughly described in Section 2.3. The features are extracted by the machine learning model without engineering expertise about the underlying dependencies.

Of the various machine learning techniques utilised in RF fingerprinting, neural networks (NN) are by far the most popular. Studies about convolutional neural networks (CNN) in RFFI include [90], [63], [62], and [71]. Recurrent neural networks (RNN) with long short-term memory (LSTM) approach are applied in [68] and [84]. However, there exist methods that utilise other NN frameworks as well. Namely, generative adversarial networks (GAN), that are each built upon two neural networks, the generator and the discriminator, have been employed in verification of transmitter identity in [60]. Typical techniques are listed in Table 2.3.

Deep learning (DL), which involves deep neural networks, has proven great success in the recent past. For instance, it has been successfully utilised in computer vision, such as image classification, object detection, and natural language processing (NLP). DL has many properties that are suitable for RFFI as well. In deep learning, there are several hierarchical layers in a neural network, hence they are called deep neural networks. These layers can automatically extract those features that the traditional methods calculated from the data. The deeper the layers are in the NN, the more evolved features they can observe. There are numerous different architectures, or structures, for deep neural networks. [37]

When compared to feature-based RFF methods, the data-driven raw signal based methods are more straightforward to use, which often makes them a more tempting alternative. That is because they do not involve tedious derivation and selection of the features, nor the process for finding them, by the user via separate

calculations. Thus, also less domain-specific engineering expertise is needed, as long as there is access to labelled training data, which will be later further discussed. The machine learning model classifies the signals based on the features that it finds, and the process within is not visible outside. Naturally, this seemingly easy procedure leads to black-box programming, and a risk arises that the ML model eventually makes wrong classifications, if the model or input data is somehow corrupt. For instance, in [90], the different hardware impairments enabling RF fingerprinting are studied. It is stated that carrier frequency offset (CFO) and IQ imbalances in receivers should be calibrated before the identification process so that neither the temporal and unpredictable variation of CFO nor IQ imbalances of the receiver side will degrade the fingerprinting process. Thus, although no knowledge about features and impairments is needed for performing raw-signal-based RFFI, it is still required for obtaining appropriate results.

2.2.3 Challenges in RF fingerprinting

It is not always straightforward to find and extract RF fingerprints. Challenges for RF fingerprinting concern e.g. signal quality, utilised receiver, or collected data. The signal quality is affected by its environment. The environment includes the state of the electromagnetic spectrum, other users of the shared spectrum, propagation environment, mobility, and temperature. These determine the channel and noise level discussed below. Environmental factors may render the received signal difficult to observe or significantly different.

One well-known challenge in RFF is the effect of the unknown radio channel on the signal quality. A *radio channel* is combined of media and path that a signal propagates over when travelling from a transmitter to a receiver. [70] The channel influences the signal that finally attains the receiver: the different media may attenuate it and it may travel through multiple paths leading to interference among different delayed versions of it, and consequently fading. The radio channel can be employed e.g. in indoor localisation of a device, as the channel alters with respect to the location and the upcoming signal paths. However, the variation due to the channel poses a problem for RF fingerprint identification, since the extracted fingerprints may be more affected by the channel than the analogue impairments, and thus the RFF method may end up recognising channels instead of emitters. Nevertheless, in case of a line-of-sight (LOS) setup where the signal can proceed straightforwardly from the transmitter to the receiver, the influence of the channel will decrease. Different methods have been developed to tackle the effect of the channel as well. For instance, the received IQ data is equalised before classification in [2] and [63]. Artificial channel-invariant impairments, that are added to the signal at the transmitter, are introduced in [62].

Noise level also greatly affects the signal quality in RF fingerprinting. Noise is a random and unobservable disturbance present in any physical measurement. It is commonly modelled using probability distributions such as Gaussian distribution. In practice, noise contains thermal noise formed in the electrical circuits and components in the devices, ambient noise from the atmosphere or space, and transmissions from

irrelevant transmitters or electrical systems [31]. Since noise makes the signal more vague, sometimes it may be challenging to even detect the signal, let alone the fingerprint. The longer is the distance from the emitting device, the weaker the perceived signal will be. The noise also prevents the signal from arriving as fully deterministic to the receiver but presents some randomness to it. Since noise weakens the signal quality, it also makes the fingerprints more uncertain by adding the variance of the observations and consequently the variance of features or raw signal. The relation between signal and noise levels is called signal-to-noise ratio (SNR) and it illustrates the visibility of the signal in the presence of noise. It is typically defined in a logarithmic scale and described quantitatively using decibel units [31]:

$$\mathrm{SNR[dB]} = 10 \cdot \log_{10} \frac{\sigma^2_{signal}}{\sigma^2_{noise}}, \tag{2.1}$$

where σ^2_{signal} and σ^2_{noise} are the variances of signal and noise, respectively. All in all, one may not be able to extract a fingerprint from a device if it is far away from the receiver or if there is otherwise much noise present in the signal.

An especially challenging task is to obtain fingerprints from emitters that do not wish to be found. There exist low probability of detection (LPD) and low probability of intercept (LPI) signals that intentionally have very low power spread over a broad bandwidth, hence making the detection and subsequent fingerprinting difficult. Typical examples of these signals are broadband pulse-compression radar code signals in radars and spread spectrum signals in wireless communications and satellite navigation systems.

Receiver imperfections pose another challenge to RF fingerprinting. Ideally, an extracted RF fingerprint is a property that is able to connect a signal to exactly one transmitter by being a sufficiently distinct or dichotomising representation of the analogue impairments in that transmitter. Nonetheless, the impairments in components of receivers show in fingerprints as well, even if transmitters' variations are the ones of interest. Hence, an RF fingerprint is not only unique to a transmitter, but also to the receiver that recorded it. There are a few cases where the receiver-dependency of a fingerprint can be ignored. Since impairments are more common in low-end than high-end equipment, it is possible to perform RFF identification by first extracting the fingerprints by a high-end receiver and then later distributing them to various, more low-end receivers. Another option is to use many different low-end receivers and hope that the receiver differences average out while the transmitter differences remain.

A crucial challenge in RFF is to have proper training data. That is, sufficiently large amount of representative data should be obtained for successful training, since RF fingerprinting is based on supervised learning and labelled training data. The fingerprint identification algorithms will have poor performance if they have been given too small a number of training data from any class, since the fingerprints are hard to be found and learned if there is not enough data. What's more, the amount of available data alone is not enough as training data should exhibit all the variation that is expected from the actual data for the classification. That is, the data needs to be representative and cover different transmitters, signal quality values

such as SNR, and environmental conditions, to name a few. Naturally, the sufficient variety of data is determined based on the application. Especially, training data from every transmitter type in question must be available, and thus it is not possible to identify a device type that has not been presented in the training data. On the other hand, recording, managing, and storing a vast amount of data is both time- and resource-consuming, to say nothing of its processing and classification in RFFI. There is no culture of sharing RFF training data, which underlines the challenge of forming a large and representative training data set for method development and comparison. For actual model training, data must always be collected from the specific emitters that should be identified. Hence, a balance must be found such that the RF fingerprints are created from a large enough amount of, and diverse enough, signal samples, while keeping the size of the data set in practical limits.

2.3 Classification of RF fingerprints

Table 2.3: Different classification methods for RFFI.

Classification method	References (feature-based)	References (data-driven)
Support vector machine (SVM)	[6],[8]	
T^2 statistics	[21]	
K-nearest neighbour (k-NN)	[27],[8]	
Artificial neural network (ANN)	[9]	
Probabilistic neural network (PNN)	[81]	
Multilayer perceptron (MLP)	[57]	
Convolutional neural network (CNN)		[90],[62],[63],[71]
Recurrent neural network (RNN)		[68],[84]

Classification is a procedure where one or more instances are to be categorised into proper classes. It is commonly assumed that there are a finite number of classes. For example, when classifying handwritten number characters, there are 10 classes for the digits of 0 to 9. Often the classification algorithm performs supervised learning in which the correct class for each training data instance, as well as the number of possible classes, is known. However, some methods utilise unsupervised learning and hence may start without knowing how many classes there are. In that case, classes are formed by clustering similar instances into groups. It is also possible to attach the uncertainty associated with the classification result into it, making the process more robust. In the context of this thesis, there is always at most one correct class for each instance.

There are two distinct classification tasks in RF fingerprinting. The one aims to find out whether a device is the one it claims to be, in which case a fingerprint of a transmitter is only compared to the identifier of the known, certified device the former claims to be. This method is called *(identity) verification*. On the other hand, when one wants to find out which of the several known transmitter types is

transmitting, one needs to perform *identification*. [10],[6] In the following, the focus lies on the identification task, but similar principles apply for verification as well.

Classification is a vital phase in RF fingerprinting. Whether the obtained fingerprint is a feature vector or raw signal, it must be classified in order to identify the transmitter, i.e. the origin of the received signal. This may mean identification of a particular type of transmitter or even recognising an individual radio transmitter. Some previous knowledge, that is, annotated training data, from every possible transmitting device type is required to train the classifier. Such training data has to be labelled or annotated so that the correct class is known for each sample in the training data. First fingerprints must be collected of those transmitters that will be considered as potential classes in the actual identification phase. Next the classifier needs to be trained with those fingerprints. Later the fingerprints that come from an unknown emitter will be compared to the existing ones. The success of identification requires that the unknown emitter be one of the transmitter classes that were included in the training phase.

In the training phase of a classifier, the data is divided to training, validation, and test data sets. First, the training data is employed to create a model that contains dependencies and hierarchy of the training data. Next, the validation and test sets are used to ensure that no overfitting takes place and that thus the classification generalises well, not only to familiar training data but to unseen data as well. Cross-validation, where combinations of results from different validation sets are used for validating the accuracy of the model, is one example of such approach [78]. The model accuracy can be measured based on classification accuracy or loss function value.

There are various different classifiers. Traditional feature-based RFF techniques use statistical or machine-learning methods, whereas modern data-driven raw-signal-based methods solely rely on neural networks. Statistical methods such as support vector machine (SVM) or T^2 statistics have been employed in [6] and [21], or [21], respectively. Also M-ary hypothesis testing or logistic regression may be used. Even k-NN algorithm, which is a classical clustering algorithm based on unsupervised learning, has been utilised as in [27] and [8]. Machine learning in RFF with neural networks, also called artificial neural networks (ANN) as in [9], include probabilistic neural network (PNN) [81], multilayer perceptron (MLP) [57], convolutional neural network (CNN) [90], [62], and recurrent neural network (RNN) including long short-term memory networks (LSTM) [68], [84]. More precisely, often the applied neural networks are deep neural networks (DNN), meaning that they have more than one layer of artificial neurons [52]. In this thesis, the focus lies on NNs, see Section 3. Different classification methods are presented in Table 2.3.

Although neural networks are popular in state-of-the-art RF fingerprinting, they have their limitations. Most recent RFF research has tackled problems that occur when using CNNs in radio-frequency fingerprint identification. Namely, convolutional neural networks only accept signal inputs of constant length. Besides, the identification may not perform well at low signal-to-noise ratios (SNR), as occurs in many signal processing applications. Consequently, new machine-learning approaches have been presented, such as Flatten-Free CNN, variants of recurrent neural networks

Table 2.4: RF fingerprinting datasets available online. All the presented datasets use SigMF as data format.

RX	TX	Number of emitters	Size	Reference
USRP B210	USRP X310	16	>320e6 samples	[62]
USRP B210	Lopy, Fipy, LoRa	25	1.2TB (16300 files)	[15]
USRP N210	USRP N210/X310	20	not mentioned	[2]
USRP X310	DJI M100 UAV	7	>13e3 samples	[71]
USRP B210	USRP X310	4	120e6 samples	[58]

(RNN), and transformers, that work for variable-sized inputs. [68]

The phase information plays an important role in RFF, which the classification system should take into account. Namely, the raw IQ data contains both I and Q branches which makes it a complex-valued observation that can be further processed e.g. with machine learning techniques. The I branch is represented by the real, and Q branch by the imaginary part of a sample vector. When dealing with complex-valued data, both amplitude and phase information are contained in it. It is common to present complex-valued data as bivariate real-valued vectors. However, complex-valued representation is very compact, captures the behaviour of phase information more explicitly and facilitates convenient processing both in time and frequency domains. This is where complex-valued signal processing comes at hand.

2.4 Publicly available RFF data sets

One major challenge in developing algorithms for RF fingerprint identification is the need for vast enough, and representative enough, collections of labelled signal data. It is time-consuming and necessitates equipment to construct such a comprehensive selection of measured real-world signals. Hence, it turns out practical for many to utilise data sets created and shared by other researchers. RF fingerprinting datasets that are publicly available online include those employed in [62], [15], [2], [71], and [58]. Characteristics of these datasets are to be found in Table 2.4.

Using of existing data sets enables fair comparison of different methods. However, it must be noted that the ready-made data sets are only helpful in a method's development phase. When in action, RFF identifiers will need data from the specific emitters or emitter types that need to be identified in a particular application or scenario. Another option for development of methods is to produce simulated RF data, but this does not suit practical applications. Besides, simulated data is often limited in the sense that it does not exhibit all the variation that the real-world data does.

It is noteworthy that plenty of RF fingerprinting data exists, however, it is often not publicly available. Such data that cannot be accessed include those produced by wireless operators, safety and security organisations, or the military. Unfortunately, there is no culture of sharing this kind of data because of business and security reasons.

3 Machine learning

Machine learning (ML) is a computational framework which allows solving problems that would be too complicated to construct using rigorous mathematical models or impossible to model beforehand, using the concept of learning and thus development by the algorithms. Machine learning is an attractive solution when there is modelling or algorithmic deficit in solving the problem but plenty of data available. It belongs to the wider concept of artificial intelligence (AI), or more precisely to narrow AI. It starts by having training data of which some pattern or rules should be extracted. For instance, the aim can be to predict or classify future, yet unknown data samples, but there are other tasks as well. The data is learned by iteratively training parameters for the model that is to be found. [89],[25]

There are various tasks for which machine learning can be applied, including classification, and regression, also known as prediction. In classification, labels or classes are assigned to input samples. In training, the system tries to classify known data, obtaining feedback about the success of the classification and learning dependencies between instances and their classes. When training is completed, unknown samples can be input to the classifier for determining classes for them. The learning performance depends heavily on the quality and size of the training data, although e.g. how representative the data is, the size of the model, and the number of its parameters that have to be trained affect as well. [43]

The three well-known different approaches of machine learning are supervised, unsupervised, and reinforcement learning. In the most common approach, *supervised learning*, the model learns patterns of the data based on previously seen examples with a correct class label, whereas *unsupervised learning* requires the model to find patterns of the data or clusters on its own. Another option is *reinforcement learning* where an agent learns by trial and error and performs sequential decision making by observing the state of its environment and choosing appropriate actions to maximise its rewards over time. However, other approaches have been defined as well. When ML algorithms imitate biological evolution of species, they are related to *evolutionary learning* [41]. In some listings, also *semi-supervised learning*, that is, approach that takes after both supervised and unsupervised learning, is mentioned [64]. In the following, the focus lies on the supervised learning and classification task, since this is the relevant one in RF fingerprint identification. [25], [41], [64]

Supervised learning is the base for many ML algorithms. In it, the task is to approximate an unknown function that leads from inputs to outputs. Supervised learning mainly consists of two types of models, namely classifiers and regression models [45]. Classifiers can be further divided into e.g. probabilistic, linear, and other types of classifiers [64]. Following this classification, probabilistic classifiers include Naïve Bayes, Bayesian Network, and Maximum Entropy Classifiers. Examples of linear classifier methods are Support Vector Machine (SVM), Logistic Regression (LR), Decision Trees (DT), and Neural Networks (NN). Nevertheless, NNs can be employed in regression tasks as well [73]. The other types of classifiers include for instance quadratic classifiers and boosting. Boosting is a procedure where results of any classifier are improved by making the learning focus on more challenging

instances [66]. However, besides ML algorithms, other classification and pattern recognition methods exist as well that are purely statistical. Examples, that all represent unsupervised learning, include k-nearest neighbour (k-NN) algorithm [32], clustering [85], principal component analysis (PCA) [13], and self-organising map (SOM) [29]. In this thesis, the interest lies on neural networks.

Neural networks (NN) are artificial counterparts of the animal brain. They consist of neurons, or nodes, and connections among them. The connections are described by weights, also known as network parameters, and updating these weights leads to actual learning of the NN. Nodes form layers such that nodes are interconnected to the nodes of previous and successive layers. The dimensions of a NN are defined by its numbers of nodes and layers. If there are multiple layers, then the NNs are typically called deep neural networks. [51]

Perceptron, the first, and also the most simple, neural network, dates back to 1958 [41]. It contains one layer of neurons, each of which having one set of weights $\mathbf{w_i}$ that are applied to an input $\mathbf{x}$, after which a nonlinear activation function h is employed in order to attain the output $\mathbf{y}$:

$$y_i = h(\sum_j w_{ij} x_j), \quad i \in 1, ..., N, j \in 1, ..., M, \tag{3.1}$$

where N is the number of neurons, and thus the length of the output, and M is the length of the input. [41]

Since the perceptron, neural networks have immensely evolved. By increasing the number of successive layers, where the input of a layer is the output of its predecessor, a multilayer perceptron (MLP), which is similar to a multi-layer feed-forward (MLF) neural network, is obtained. MLPs, also known as feed-forward neural networks, are such that all nodes in adjacent layers are connected to each other [4],[59]. Nowadays, a wide variety of different neural networks exist, such as artificial (ANN), probabilistic (PNN), convolutional (CNN), and recurrent (RNN), including long short-term memory (LSTM) [22], recursive, or unsupervised pretrained neural networks. [41], [51], [74], [88]

Deep learning (DL) is encountered when NNs with multiple layers are utilised. When neural networks have a lot of parameters and multiple layers, they may be called deep neural networks. In recent years, the growth in computing power, access to reasonably affordable hardware such as graphics processing units (GPU), and amount of available data has allowed the use of deep NNs, making DL approaches more and more popular. Well known DL architectures that both involve CNNs include GoogLeNet [75] and AlexNet [34]. These two architectures are mainly used to classify image data, and convolutional networks in general are often used for this purpose as well. [5]

A simplified example of a feedforward deep neural network is presented in Figure 3.1.

3.1 Training phase in supervised learning

In all supervised learning, the model is trained by feeding it with known, labelled data. This is called a *training set*. When there is much training data, the training

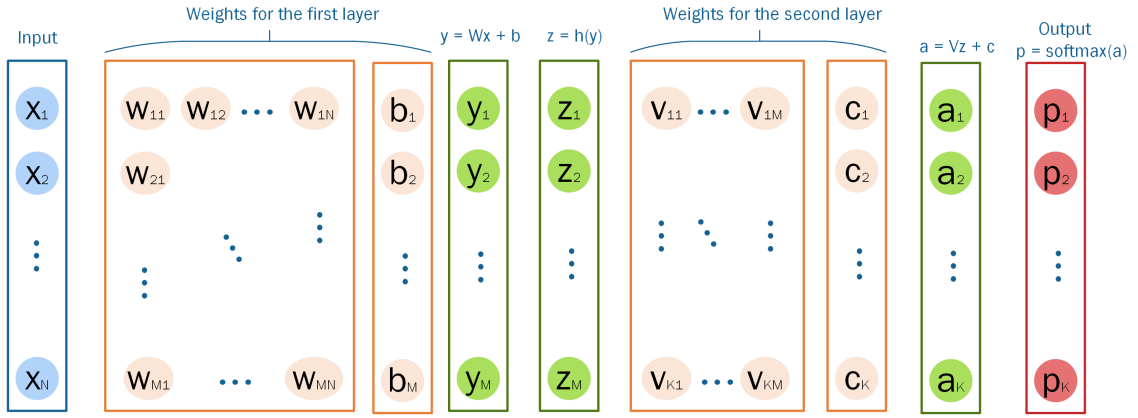

Figure 3.1: Example of a simple feedforward deep neural network. The input $\mathbf{x}$ is either a data or feature vector, and the output vector $\mathbf{p}$ provides probabilities of $\mathbf{x}$ belonging to different classes, of which a suitable class label is deduced. Neurons in the NN are presented with green circles, where $\mathbf{z}$ is the output of the hidden layer involving both linear and nonlinear parts. The weights to be adjusted are presented with light orange circles, and consist of matrices $\mathbf{W}$ and $\mathbf{V}$, as well as vectors $\mathbf{b}$ and $\mathbf{c}$. Function h is a nonlinear activation function.

set is often batched so that not all of the data is utilised simultaneously on every training round. The observed data may be pre-processed, such as normalised, before the training. In the actual training, the input samples are fed to a machine learning model which applies functions to them.

The model assigns different weights that represent importance to different parts of the input. In the beginning, the weights are typically randomly initialised. The goal of the training process is to find weights that make the model illustrate the data of the problem. In other words, the model should learn the dependencies between the data and their corresponding labels. To illustrate this task, a simple curve-fitting problem is next presented. The example is relatively distant from modern ML problems, but the idea can be further extended.

A simple example of this kind of a task is to find coefficients to a function that should fit into a set of points. Illustration is provided in Figure 3.2, where orange points represent the training data. A function should be found that passes every point as close as possible. Figure 3.2 a) presents a poor result where a most simple function, a straight line, may fit relatively well to some points but contains no information about the points far below it. On the other hand, Figure 3.2 b) shows a perfect fit to the same points. However, it can be seen that the blue function curve has now a remarkably more complicated form and fluctuates more than the line in the left picture. That is, it needs many parameters in order to be defined.

Alike the example above, a training process of a supervised learning problem can be seen as fitting a multidimensional curve to a set of data points in multidimensional space. The task presented thus works as a simplification of various of such problems. However, adding more parameters till the model fits flawlessly to the data is rarely optimal. That is, the final aim is not to represent the training data as a beautiful

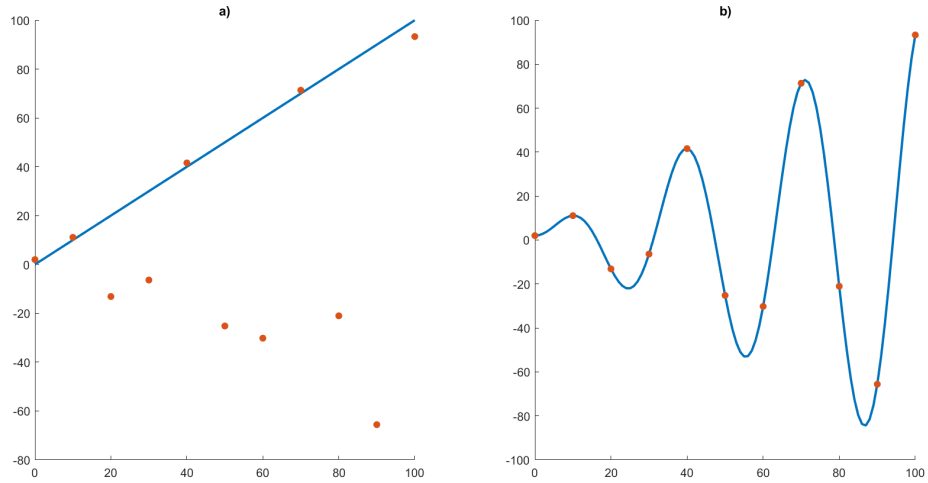

Figure 3.2: Example of curve fitting into a set of points.

curve but to create a model into which new unseen samples can be input, and which still fits nicely enough. The situation where the model fits perfectly to its training data but is of no use when considering new samples is called *overfitting*, or sometimes *overlearning*. Such model may not generalise well to new data in the actual classification phase. To summarise, the aim is to train the model so that it generalises well to unseen data.

The training itself is conducted by applying the model using first initial network weights and then after each iteration adjusting the weights a little. This adjustment is based on minimising the loss, that is, the error between the model and the training data. There are different methods for computing the loss, but e.g. in the curve fitting example above, a loss criterion could be the least square error between the data points and the blue curve. In each iteration in the training process, the weights are updated so that the loss decreases. In a favourable scenario, this finally leads to convergence between the data and the model. On the other hand, in some methods, a sufficiently optimal solution is already found after the first iteration and weight adjustment.

In neural networks in general, training process is performed in sequential *epochs*, though some methods only use one epoch. In multi-epoch NNs, the model tries to get better by adjusting its parameters, and in the end of each epoch, it is facing a choice: either to quit or continue the training. The criterion for quitting is normally some of the following:

- The maximum number of epochs, n_{max}, has been attained.
- The model has achieved the targeted performance level.
- The model is not learning anymore.

An epoch is a part in the training during which the whole training set is fed to the NN. Usually, the training data is handled in smaller batches so that the

learning speeds up, and an epoch consists of training with all the batches. Without batching, the whole training set would be fed to the NN simultaneously, which would require huge amounts of time and memory. However, also exceedingly small batches decelerate the learning as the amount of batches grows large.

3.2 Operation of feedforward neural networks

Feedforward neural networks involve operations in two directions. First, the NN is fed with an input, and the functions of the layers of the NN are applied to it successively. This is called the forward path. The direct path from input to output distinguishes feedforward NNs from recurrent NNs (RNN) that loop the data during the forward path [11]. After the output has been obtained, the neurons are backpropagated and the network parameters updated. Thus, the backpropagation path is followed.

3.2.1 Forward path

Feedforward neural networks are models that for instance classify samples. The samples are fed to the first layer, and as the name suggests, the operation flows forward in the NN.

The NN dimensions, and thereby also the number of weights, are affected by several decisions made by the designer of the network. The number of layers, as well as the numbers of neurons in hidden layers, are freely chosen by the designer, though increase in them naturally leads to more resource-consuming learning. The numbers of neurons in input and output layers, however, are determined by the size of the input data and the number of possible classes, respectively.

The neural networks like the one in Figure 3.1 only have fully connected (FC) layers, which means that every neuron is affected by all the neurons of the preceding layer. However, this approach leads to an extensive amount of trainable parameters within few layers, and thus resources are vastly consumed. A more parameter-efficient alternative is to form new neurons of a small collection of the neurons of the preceding layer only. The corresponding weights are called a kernel, and the operation of linear transformation between the kernel and the selected neurons is a convolution. Hence, layers that apply convolutions are called convolutional layers, and NNs that include those are convolutional neural networks (CNNs).

Architecture of a CNN layer differs from that of a fully-connected layer. In Figure 3.1, changing the first fully connected layer to a convolutional layer would replace the matrix $\mathbf{W}$ by a smaller matrix, that is, the kernel. CNNs are often used in image processing where the kernels are two-dimensional. However, in RF fingerprinting, one-dimensional kernels are typically suitable. A one-dimensional kernel of length λ is convoluted with $\mathbf{x}$ such that it is matrix multiplicated with the first λ elements of the input. The input can be either $\mathbf{x}$, or if the padding parameter p_{pad} is more than 0, $\mathbf{x}$ with p_{pad} zeros padded before and after the original vector for the one-dimensional case. Next, the kernel proceeds by s elements where s is a stride hyperparameter, and multiplies the next λ elements of the input by the kernel. Each of these products is the value of a neuron in the following layer. The size of the following layer is

thus determined by the size of its precedent layer, as well as kernel size, stride and padding hyperparameters. [89]

Besides linear layers, neural networks also include nonlinear activation functions that are applied to the outputs of the linear transformations. In fact, the activation functions form an essential part of a neural network. Popular activation functions include rectified linear unit (ReLU), sigmoid function, and hyperbolic tangent (tanh) which are shown below and in Figure 3.3 as f_{ReLU}, $f_{sigmoid}$ and f_{tanh}, respectively: [89]

$$f_{ReLU}(x) = max(0, x) = \left\{ \begin{array}{ll} x & \text{if } x \geq 0 \\ 0 & \text{if } x < 0 \end{array} \right\} \tag{3.2}$$

$$f_{sigmoid}(x) = \frac{1}{1 + e^{-x}} \tag{3.3}$$

$$f_{tanh}(x) = \frac{1 - e^{-2x}}{1 + e^{-2x}} \tag{3.4}$$

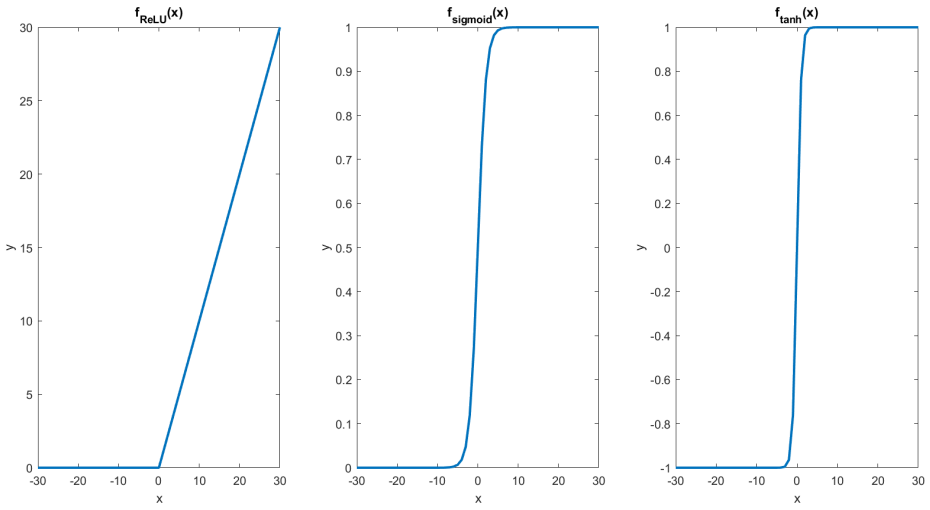

Figure 3.3: Three commonly used activation functions.

Activation functions are essential for the learning process of a feedforward neural network since with linear layers only, the whole NN could be simplified to one linear transformation. Especially, ReLU has been found to improve the neural network training [1].

In classification task, the NN should output probabilities of the sample belonging to each of the possible classes. Therefore, softmax function is applied as the last activation function in order to transform network outputs into values in the range of [0, 1]. The softmax function is defined as follows: [89]

$$\{f_{softmax}(\mathbf{o})\}_i = \frac{e^{o_i}}{\sum_j e^{o_j}}, \tag{3.5}$$

where i is the index of the output neuron in question and $\mathbf{o}$ is the output vector. The probabilities offer a measure of how certain the model is about its outcome. However,

in classification, a class should be addressed to each of the samples and thus the class corresponding to the highest probability is chosen, leading to a discrete-valued output.

3.2.2 Backpropagation

Backpropagation is an approach for updating parameters where the layers in a feedforward NN are gone through in a backward direction. In backpropagation, error of the NN classification result is given as a function of the network weights, which allows for optimising the weights such that the error is minimised. A visualisation of backpropagation process is shown in Figure 3.4.

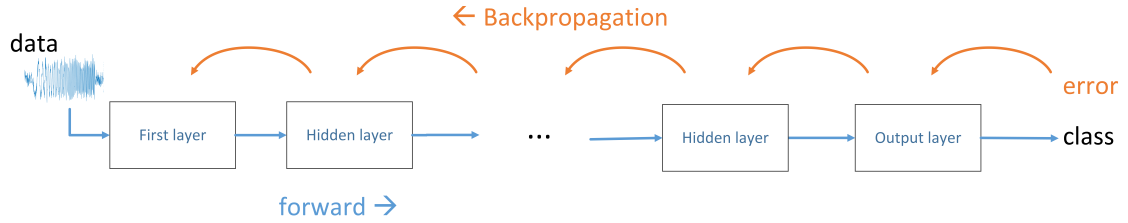

Figure 3.4: Illustration of backpropagation in a feedforward neural network.

The error of the NN classification is measured using a loss function. There are various different loss functions, however, most NNs use cross entropy loss that is also called log loss. Cross entropy loss is conventionally utilised with application of softmax function [42]. Another examples of loss functions, though at least in classification only marginally used, include mean absolute error using L_1 norm, mean squared error using squared L_2 norm, expectation, regularised expectation, Chebyshev, hinge, squared hinge, cubed hinge, squared log, and Tanimoto losses, and Cauchy-Schwarz divergence. [26] Also when using cross entropy loss, multiple variations exist that employ e.g. label smoothing, dropout, logit penalty and logit normalisation, to name a few [30].

The backpropagation step involves differentiating the loss of the NN output with regard to network parameters, that is, weights. Thereby, the weights can be optimised and new parameter values obtained. The backpropagation procedure is more precisely described in the example in Subsection 3.2.5.

3.2.3 Validation

After the model parameters have been updated, it is useful to check whether the new model is able to classify samples, or whether it is any better than the previous one. The model should not overlearn the training set but instead generalise well for new test data. This step is called the validation phase.

Typically, a *validation set* is formed from data that is not in the training set. This approach is called train-and-test, and it is recommended when the amount of data available is considered sufficient [78]. Another option is to use the same data for both training and validation, and hence obtain larger, though overlapping, data

sets, but this demands computational resources. A commonly used such method is cross-validation in which the data is split into training and validation sets multiple times, and the mean of the classification errors is used as the validation error. It is important that the training and validation sets, if they are small, are different every time in order to prevent overfitting as the model should not overlearn the training set. Furthermore, instead of cross-validation, e.g. resubstitution or bootstrapping can be employed in order to virtually expand the number of samples available [78].

When the validation set is formed, its data is propagated into the network and loss or accuracy is calculated to measure the learning performance of the model. Loss is obtained as in the training phase, and can thus be for instance mean absolute error (MAE) or root mean squared error (MSE) [78]. The validation accuracy of the model is defined as the proportion of validation samples that are classified correctly by the model and is also known as percent good (PG) classification [78].

After the validation, the training has proceeded one epoch, and the ending criterion is checked. If the criterion is met, the NN is considered trained. On the other hand, if the ending criterion is not fulfilled, another epoch is needed and the training continues. A block diagram of one epoch, that is, training and validation, is presented in Figure 3.5. [78]

3.2.4 Testing

When the training has finally been completed, the model may be tested with previously unused data in order to find out how well its predictions fit the data. This yet unseen data is called a *test set*. Nevertheless, there seems to exist some confusion between the terms *validation* and *testing* and they are sometimes used interchangeably. Strictly speaking, the test data should be such that it is only fed to the model after all parameters are fixed, and not before. [89]

For classification task, the NN performance is usually measured by PG classification of the test set. The result is also known as NN accuracy:

$$\text{acc} = \frac{\text{test set samples classified correctly}}{\text{test set samples}} \tag{3.6}$$

Neural networks can achieve good results with regard to accuracy of classification, however, it is also essential to know how reliable they are. Some kind of probability estimate of the confidence of the results should be available to explain the classification outcome and its trustworthiness. High network accuracy, i.e. test set accuracy, does not guarantee, or even necessarily correlate with the results being certain. Hence, the networks should often be better calibrated, and confidence intervals used instead of mere probabilities or binary variables as outputs. An example of a calibration method is temperature scaling where the NN outputs have an added amount of entropy without decrease in overall accuracy [18]. Alternative methods include the use a conformal loss function [14], or soft quantiles [50].

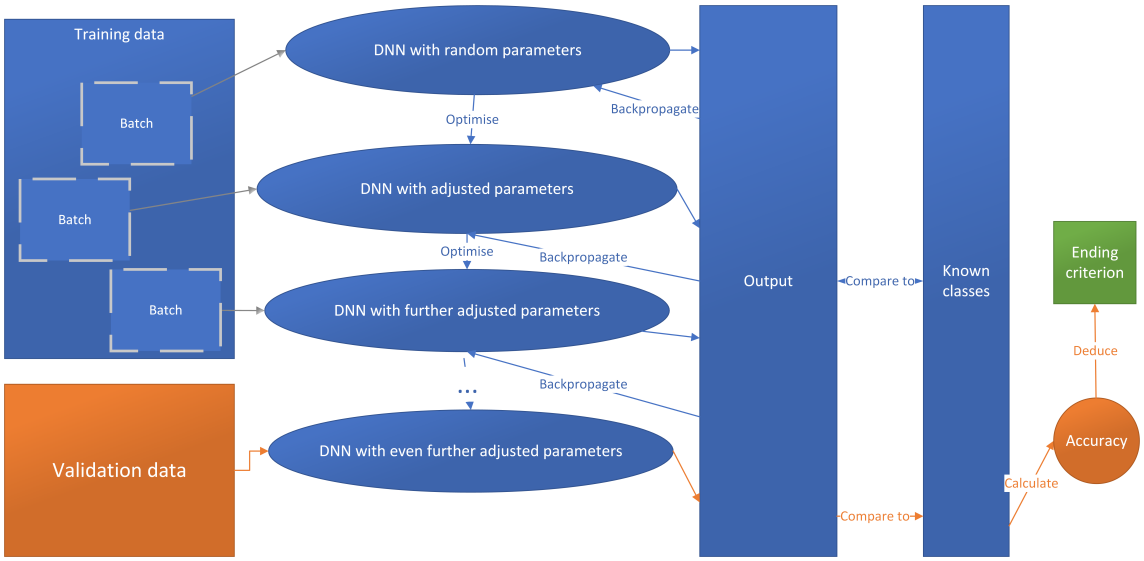

Figure 3.5: Phases of an epoch illustrated. The blue and grey arrows represent training, where batches of the training set are input to a deep neural network (DNN). With the help of the outputting results and the known correct classification, the loss is calculated, and the backpropagation is performed through the DNN, yielding adjusted network parameters via optimisation. Next, another batch of training data is input to the NN with these adjusted parameters, and the procedure is repeated successively. Finally, after going through all the batches, NN with optimised parameters is obtained, and it is tested by putting samples of validation data into it. The orange arrows represent the validation phase. Here, the validation approach is train-and-test and the validation metric is validation accuracy. The ending criterion determines whether the training is finished or the whole procedure is repeated.

3.2.5 Example of an epoch

In this section, the training procedure in an epoch is presented with the simple neural network in Figure 3.1 as an example. When considering deeper neural networks, the processus for one layer, consisting of both linear and nonlinear parts, should be repeated until the end of the network is reached.

Training data is first subdivided into batches, each containing β input vectors, where β is a hyperparameter for batch size. The input samples are then fed into a neural network where there are one or several layers. In Figure 3.1, the original input of length N is represented by vector $\mathbf{x} = (x_1, x_2, ..., x_N)$. Output of the linear part of the first layer is obtained by multiplying $\mathbf{x}$ by a weight matrix $\mathbf{W}$ and adding bias $\mathbf{b}$: $\mathbf{y} = \mathbf{W}\mathbf{x} + \mathbf{b}$, thus making a linear transformation of the input.

In the beginning of the training process, the weights are random. That is, the coefficients $w_{ij} \in \mathbf{W}$ and constants $b_i \in \mathbf{b}$ of the functions are initially drawn from some distribution. In the preceding, $i \in \{1, 2, 3, ...M\}$, $j \in \{1, 2, 3, ..., N\}$, and M is the number of neurons in the first hidden layer. The weights are often called parameters, and their values are repeatedly updated during training. Referring to Figure 3.1, next a nonlinear activation function h is applied to the vector $\mathbf{y}$. In

Figure 3.1, the output of the activation function, to be marked with $h(\mathbf{y})$, is the vector $\mathbf{z}$: $\mathbf{z} = h(\mathbf{y})$.

Next, the second layer is considered. In case of classification the number of neurons in the last layer, that is, the dimension of $\mathbf{a}$, is the number of output classes K. This requires that a properly sized weight matrix, denoted here by $\mathbf{V}$, be applied to $\mathbf{z}$. Here this linear transformation is performed via matrix $\mathbf{V}$, that has the size of its first dimension equal to K, such that $\mathbf{a} = \mathbf{V}\mathbf{z} + \mathbf{c}$.

As an activation function, softmax function is applied to the vector $\mathbf{a}$, yielding probabilities of the original input belonging to each of the considered classes. The probabilities form a vector $\mathbf{p}$, where $p_i \forall i \in \{1, 2, ..., K\}$ is the probability that $\mathbf{x} \in i$. Next, the outputs $\mathbf{p}$ are compared to the known correct vector, $\mathbf{d}$, where $d_i = 1$ if $i = j$ and $d_i = 0$ if $i \neq j$ and the jth class is the correct one. Loss L is computed using a loss function l: $L = l(\mathbf{p}, \mathbf{d})$. The loss is an auxiliary variable that is needed in the training.

Here, cross entropy loss is employed. It is defined as follows: [42]

$$L(\mathbf{p}, \mathbf{d}) = -\sum_i d_i \log p_i, \tag{3.7}$$

where d_i and p_i are the ith elements of $\mathbf{d}$ and $\mathbf{p}$, respectively.

After the calculation of the loss, the parameters $\mathbf{W}, \mathbf{V}$ and $\mathbf{b}, \mathbf{c}$ can be adjusted so that the model works better. Backpropagation and parameter update are used for this. First, the derivatives of the loss function are calculated with regard to the parameters of the last layer, since the goal is to find directions where the loss would decrease by adjusting the weights in the direction of negative gradient. Thereby, the gradient $\nabla L = (\frac{\partial L}{\partial v_{ij}} \forall i, j)$ with regard to the network parameters is to be obtained. Thus, L is written as a function of v_{ij} so that the differentiation can be performed, where v_{ij} is the element of V in ith row and jth column. Hence, the route from the end of the NN to the last linear layer must be calculated backwards step by step, thus, it is to be backpropagated. [61] Towards this goal the chain rule of differentiation is utilised. First, the partial derivative of the loss with regard to the last layer output is calculated: i.e. $\frac{\partial L}{\partial \mathbf{p}}$. This result is then used when chain rule is employed and the partial derivative of the loss with respect to the output of the previous layer,

$$\frac{\partial L(\mathbf{p}(a_u))}{\partial a_u} = \sum_i \frac{\partial L}{\partial \mathbf{p}_i} \frac{\partial \mathbf{p}_i}{\partial a_u}, \tag{3.8}$$

where $u, i \in \{1, 2, ..., K\}$, K is the length of $\mathbf{a}$ and $\mathbf{p}$ from Figure 3.1, and $\mathbf{p}(\cdot)$ is the last layer of the NN that is dependent on its previous layer given as parameter, is calculated. The chain-rule mechanism can be further applied till the desired derivative, that is, $\frac{\partial L}{\partial v_{ij}}$, is obtained. An illustration of how backpropagation proceeds is provided in Figure 3.4.

Once the gradient $\nabla L = (\frac{\partial L}{\partial v_{ij}} \forall i, j)$ is calculated, the procedure continues by updating the layer parameters with some optimisation algorithm. One alternative for this is Stochastic Gradient Descent (SGD), which takes one step to the direction of negative gradient. The size of the optimisation step is defined by the *learning rate*

η. Using SGD as the optimisation algorithm, the new parameter values are given as follows:

$$v_{ij}^{*} = v_{ij} - \eta \nabla L(v_{ij}), \tag{3.9}$$

where v_{ij} is the old parameter value, η is the learning rate, and $\nabla L(v_{ij})$ is the gradient of L with regard to the old value of v_{ij}. Sometimes the learning rate is decayed between epochs in order to achieve the minimum faster in the beginning and more precisely in later epochs. If this is the case, the learning rate is now multiplied with the decay parameter γ to obtain the new learning rate for the next epoch:

$$\eta_1 = \gamma \eta_0, \tag{3.10}$$

where η_1 is the new, and η_0 the original learning rate. [61], [89], [44]

4 Complex-valued neural networks

4.1 Why to consider complex-valued signals?

Many signals observed by sensors are complex-valued, which supports the idea of performing complex-valued computations for them. Another, yet common alternative is to convert the complex-valued signals into bi-variate real-valued signals in order to apply real-valued calculations that are often more straightforward. However, some crucial information about the signals may be missed. Use cases where complex-valued signals are to be encountered include radar, sensor array processing, time series analysis when dealing with frequency domain processing, Fourier analysis, spectra, and medical imaging, such as magnetic resonance imaging. Additionally, in some cases, for instance in statistical shape analysis, it might actually be more straightforward to handle the signal data as complex-valued even though it doesn't first appear in this form. [47]

4.2 Circularity

Circularity is an essential measure when studying complex-valued signals. A complex random variable $z = x + yj$, where $j = \sqrt{-1}$, is said to be circular if it is distributed identically with $e^{j\theta}z \ \forall\theta \in \mathbb{R}$. The circular distributions include for example complex elliptically symmetric (CES) distributions. If z is circular, the real part x and the imaginary part y are statistically uncorrelated. If a complex random variable is not circular, it is called non-circular. When $\mathbf{z}$ is multivariate, it is marked with $\mathbf{z} = \mathbf{x} + \mathbf{y}j$ and its circularity can be defined via its second-order moments, and hence they imply second-order circularity. These moments are covariance matrix $\mathbb{E}[\mathbf{z}\mathbf{z}^H]$ that is defined by [47]

$$\mathbb{E}[\mathbf{z}\mathbf{z}^H] = \mathbb{E}[\mathbf{x}\mathbf{x}^T] + \mathbb{E}[\mathbf{y}\mathbf{y}^T] + j(\mathbb{E}[\mathbf{y}\mathbf{x}^T] - \mathbb{E}[\mathbf{x}\mathbf{y}^T]) \tag{4.1}$$

and pseudo-covariance matrix, sometimes called complementary covariance matrix, $\mathbb{E}[\mathbf{z}\mathbf{z}^T]$ that is in turn defined by [47]

$$\mathbb{E}[\mathbf{z}\mathbf{z}^T] = \mathbb{E}[\mathbf{x}\mathbf{x}^T] - \mathbb{E}[\mathbf{y}\mathbf{y}^T] + j(\mathbb{E}[\mathbf{y}\mathbf{x}^T] + \mathbb{E}[\mathbf{x}\mathbf{y}^T]). \tag{4.2}$$

The random variable $\mathbf{z}$ is said to be second-order circular, or equivalently, proper, if its pseudo-covariance matrix $\mathbb{E}[\mathbf{z}\mathbf{z}^T] = 0$. Using the definitions of the second-order moments above, it can be seen that the pseudo-covariance matrix is zero if and only if both its real and imaginary parts are zero. The real part being zero leads to $\mathbb{E}[\mathbf{x}\mathbf{x}^T] = \mathbb{E}[\mathbf{y}\mathbf{y}^T]$, and when the imaginary part equals zero, $\mathbb{E}[\mathbf{y}\mathbf{x}^T] = -\mathbb{E}[\mathbf{x}\mathbf{y}^T])$. Additionally, there exist multiple nonequivalent definitions of circularity, such as marginal, weak, strong, total, and moment circularities. [53] Examples of circular waveforms include quadrature amplitude modulation (QAM) along with 16QAM and 64QAM, and quadrature phase shift keying (QPSK) along with 8PSK and 16PSK, signals. On the other hand, inherently non-circular waveforms include amplitude modulation (AM), binary phase shift keying (BPSK), and offset quadrature phase shift keying (OQPSK). [47], [49], [53], [83]

It is essential to know whether a complex random variable, or a signal, is circular or not, since the methods for optimal detection and estimation of complex-valued signals vary based on that [55]. Actually, methods for non-circular signals work for circular ones as well, but employing them without need consumes computing resources excessively. For instance, commonly used covariance matrix $\mathbb{E}[\mathbf{z}\mathbf{z}^H]$, the definition of which was covered in Equation (4.1), does not contain information about the relationship of real and imaginary parts of a complex signal, and is thus not sufficient for non-circular signals [54]. Besides normal covariance matrix $\mathbb{E}[\mathbf{z}\mathbf{z}^H]$, pseudo-covariance matrix $\mathbb{E}[\mathbf{z}\mathbf{z}^T]$, defined in Equation (4.2), should thus be used.

The circularity of the signal can be quantised by a circularity quotient that can be defined by several different ways. In [47], the circularity matrix ϱ for a random vector $\mathbf{z}$ is defined for instance in the following manner:

$$\varrho(\mathbf{z}) = (\mathbb{E}[\mathbf{z}\mathbf{z}^H])^{-1}\mathbb{E}[\mathbf{z}\mathbf{z}^T]. \tag{4.3}$$

After obtaining the circularity matrix, the circularity quotients are given as square roots of eigenvalues of the matrix $\varrho\bar{\varrho}$, where $\bar{\varrho}$ indicates complex conjugate of ϱ. [47]

A special case for the circularity quotient is the one-dimensional circularity quotient ϱ, which is defined as follows[47]:

$$\varrho = \frac{\tau}{\sigma^2}, \tag{4.4}$$

where τ is pseudovariance

$$\tau(z) = \mathbb{E}[(z - \mathbb{E}[z])^2], \tag{4.5}$$

and σ^2 is the conventional variance.

It is also possible to examine whether a signal is circular or not via generalised likelihood ratio test (GLRT), which is derived in [48]. Nonetheless, this kind of implementation of GLRT contains normality assumption which makes it slightly unpractical in real life. Therefore, an adjusted version of the test is presented in [47] and [16]. Sometimes, it is also essential to know, if the signal is non-circular, the degree of non-circularity, and in this problem correlation coefficients are used for properness and mth-order circularity quotients for the actual circularity. [16]

For a second-order circular, or proper, signal, as stated above, the pseudo-covariance matrix $\mathbb{E}[\mathbf{z}\mathbf{z}^T]$, or in one-dimensional case, the pseudovariance τ, equal to zero. From the forms of the circularity quotient formulas, in Equations (4.3) and (4.4), respectively, it can be seen that the condition for second-order circularity implies that the circularity quotient must be zero. It has been common to casually assume circularity also for those cases that may not be it, without justification or proof. This is because derivations of algorithms for circular signals are more straightforward than those for non-circular signals. Nevertheless, a more secure way is to always utilise pseudo-covariance matrices or pseudovariances, since data processing methods for non-circular signals apply for circular signals as well. [47], [16]

A signal, even though it would contain a circular waveform, will often in practice be rendered non-circular when it is being transmitted and passing through the analogue parts of the signal processing. This is due to the impairments in the

analogue circuitry of a device, such as a radio transmitter. For example, the real and imaginary parts may become correlated or coupled via some nonlinear function. This is yet another reason to employ methods for non-circular signals.

4.3 Complex-valued calculus

When processing complex-valued data, it is common to consider complex-valued numbers as real-valued vectors and then apply functions and tools developed for real-valued data. However, this approach is not always correct, especially when performing differentiation, gradient-based optimisation or statistical inference. For example, in order to employ complex $\mathbb{C}$ derivatives, a function should be $\mathbb{C}$-*differentiable*, that is, *complex analytic*, or *holomorphic*.

A complex function $f(z) = u(x, y) + jv(x, y)$, where z is a complex random variable $z = x + jy, x, y \in \mathbb{R}$, is holomorphic if and only if

$$\exists \text{ continuous } f'(z), \tag{4.6}$$

which is equivalent with the effectiveness of the Cauchy-Riemann conditions

$$\frac{\partial u}{\partial x} = \frac{\partial v}{\partial y} \text{ and } \frac{\partial v}{\partial x} = -\frac{\partial u}{\partial y} \tag{4.7}$$

and the condition that $f(z)$ has a convergent power series and all of its derivatives exist [33]. There are many common functions that do not satisfy these conditions, e.g., none of the real-valued non-constant functions is holomorphic. Thus, special differentiation rules must be obeyed in that case. [33]

Among the $\mathbb{R}$ differentiable functions that have $\mathbb{R}$ derivative, the holomorphic ones are a special case. As is stated in the so-called Wirtinger calculus, also known as $\mathbb{CR}$-calculus, given that a function is holomorphic and thus its $\mathbb{C}$ derivative exists, its $\mathbb{R}$ and $\mathbb{C}$ derivatives must equal each other. [33], [42] A $\mathbb{R}$-differentiable function has two partial derivatives, namely $\mathbb{R}$-derivative and conjugate $\mathbb{R}$-derivative, which are obtained by taking complex $\mathbb{C}$-derivatives with respect to the complex number z or its complex conjugate $\bar{z}$, respectively. Thus, a holomorphic function, for which $\mathbb{R}$-derivative = $\mathbb{C}$-derivative, must have its conjugate $\mathbb{R}$-derivative equal to 0. [33]

With holomorphic functions, it works well to utilise the complex-valued samples as a vector of two real numbers, $z = (x, y)^T, z \in \mathbb{C}, x, y \in \mathbb{R}$, and correct results are obtained. A function $f(z)$, let it be complex- or real-valued, may be presented as $f(z) = f(z, \bar{z})$ where $\bar{z}$ is the complex conjugate of z. The $\mathbb{R}$-derivative of f is thus defined as derivative of f with regard to z, that is, [33]

$$\frac{\partial f(z, \bar{z})}{\partial z} = \frac{1}{2}\left(\frac{\partial f}{\partial x} - j\frac{\partial f}{\partial y}\right). \tag{4.8}$$

Correspondingly, the conjugate $\mathbb{R}$-derivative of f is defined as derivative of f with regard to $\bar{z}$, that is, [33]

$$\frac{\partial f(z, \bar{z})}{\partial \bar{z}} = \frac{1}{2}\left(\frac{\partial f}{\partial x} + j\frac{\partial f}{\partial y}\right). \tag{4.9}$$

When differentiating a function, derivatives are calculated by the $\mathbb{R}$ and conjugate $\mathbb{R}$ derivatives. When doing so, the results are correct no matter whether the function is holomorphic or not.

Wirtinger calculus also states that the chain rule used for differentiation of composite functions differs from that used in real-valued calculus. Namely, the derivatives are calculated as follows for function $f(g(z,\bar{z}))$ [33],[42]:

$$\frac{\partial f(g)}{\partial z} = \frac{\partial f}{\partial g}\frac{\partial g}{\partial z} + \frac{\partial f}{\partial \bar{g}}\frac{\partial \bar{g}}{\partial z} \tag{4.10}$$

$$\frac{\partial f(g)}{\partial \bar{z}} = \frac{\partial f}{\partial g}\frac{\partial g}{\partial \bar{z}} + \frac{\partial f}{\partial \bar{g}}\frac{\partial \bar{g}}{\partial \bar{z}} \tag{4.11}$$

4.4 Comparing complex- and real-valued neural networks

Complex-valued neural networks (CVNN) are a special type of neural networks where the input data, neurons and weights are complex-valued and the backpropagation is gradient-based using rigorous complex differentiation based on Wirtinger calculus. [42] This allows preservation of important phase information, for example in IQ data acquired from radio frequencies or data observed in medical imaging devices. When dealing with non-circular signals, it is essential to be sensitive to the phase information and all the distortions on both the phase and amplitude simultaneously, and thus it is preferable to utilise CVNNs instead of real-valued NNs. In fact, some studies about CVNNs already exist, however, they do not usually follow the complex differentiation rules by Wirtinger calculus, see e.g. [77].

In this thesis, a complex-valued neural network presented in [42] is studied. Thus, the CVNN differs from real-valued neural network (RVNN) in several aspects, however, they possess many similarities in structure as well. In CVNN, the neurons and parameters are complex-valued with the exception of last output layers that are real-valued. Besides, the input data fed to the network is complex-valued. Complex-valued differentiation and Wirtinger calculus are applied in backpropagation of the network, and thus it is assumed that the functions in the layers, that is, both linear transformations and the nonlinear activation functions, need to be $\mathbb{R}$-differentiable. Thus, the derivatives are calculated according to equations (4.8) and (4.9), and the chain rule is applied according to equations (4.10) and (4.11). [42]

Like the fully connected neurons in RVNNs, presented in Equation (3.1) and Figure 3.1, the fully connected neurons in CVNNs can be presented as follows:

$$y_i^l = h(\sum_j w_{ij}^l y_j^{l-1} + b_i^l), \quad i \in 1,...,N^l, j \in 1,...,N^{l-1}, \tag{4.12}$$

where superscripts are used to indicate the layer index, $\mathbf{y^l} = (y_1^l, y_2^l, ..., y_{N^l}^l) \in \mathbb{C}^{N^l}$ is the output of the lth layer consisting of linear part with coefficients $w_{ij}^l \in \mathbb{C}$ and bias $b_i^l \in \mathbb{C}$, as well as of nonlinear part with activation function $h(z) \in \mathbb{C}$. Besides, N^l is the number of neurons in layer l.

The complex linear transformations are holomorphic and in that sense straightforward. On the other hand, nonlinear activation functions may not be holomorphic and

thus their calculations are more complicated. There are different types of complex activation functions, however, they are all $\mathbb{R}$-differentiable. Some of them are purely complex-valued, whereas others are e.g. split-real-and-imaginary functions, which means that the function handles real and imaginary parts separately. For instance, the complex ReLU function (CReLU) is formed for a complex variable z in the following manner and is a split-real-and-imaginary activation function [42]:

$$\mathrm{CReLU}(z) = \max(\mathrm{real}(z), 0) + j * \max(\mathrm{imag}(z), 0) \tag{4.13}$$

When it comes to the NN output and classifiers, softmax function like in RVNNs can be applied. However, softmax requires real input, which means that the complex output of the last layer must be converted into real domain. A function must thus be put between the last complex-valued layer and the softmax function. In [42], absolute value $abs(\cdot)$ is used for this purpose. This function also works as the last nonlinear activation function of the neural network. [42]

In the backpropagation and parameter updating phase of CVNNs, proper differentiation rules must be obeyed. Thus, as the very last layers of the NN are real-valued, real-valued differentiation suffices there. However, as the complex-valued layers require complex-valued differentiation and corresponding chain rules based on Wirtinger calculus, it is a more straightforward choice to treat all the layers as complex-valued. [42]

5 Developed RF Fingerprinting Methods

The aim of this thesis is to find out whether it is beneficial in an RF fingerprint classification task to utilise complex-valued neural networks over real-valued ones. The hypothesis suggests that the complex-valued NN have an edge over real-valued neural networks at least with non-circular signals due to the fact that they utilise phase information. Many of the nonidealities present in radio transmitters turn the ideal circular signals into non-circular ones. The complex-valued neural network employed is more thoroughly described in [42], and thus it is based on the rigorous complex-valued differentiation known as Wirtinger calculus.

Neural networks have a vast number of different architectures, and several parameters affect their operation. Furthermore, parameter initialisation, forming of training set, and data batching bring additional factors that may influence the outcome. Thus, it is a real challenge to take into account all the essential variables in order to ensure that the different developed NN structures are compared in a fair manner. Furthermore, it is challenging to test all the relevant conformations and hyperparameter combinations when looking for the best possible neural network. Hyperparameters are such NN parameters that their values are chosen before the training and kept constant during learning. The problems of selecting suitable hyperparameter values are addressed e.g. in [87].

5.1 Employed Neural Network Structures

The neural networks used in this thesis are convolutional neural networks. More precisely, they contain both convolutional and fully connected layers. The activation function applied in both NNs is rectified linear unit (ReLU). However, whereas the RVNN employs the traditional form given in equation (3.2), the CVNN has to use the complex ReLU from equation (4.13).

The neural networks were implemented and run using Python programming and computing resources within the Aalto University School of Science "Science-IT" project. The real-valued neural network was built using Pytorch modules in Python, whereas the complex-valued neural network was the one presented in [42] and programmed from scratch using Python. The structures of the NNs are presented in Figure 5.1.

Both NNs consist of two convolutional and two fully connected layers. The sizes of the convolution kernels are 1 x 7 for the first and 1 x 5 for the second one. No padding of zeros is applied, which means that the samples become shorter due to the convolutions. Stride, denoted by s as in Section 3, of one is used and thus the convolution kernel moves one by one through the sample. The initial weights of the convolution kernels and the fully connected layers, as well as the initial bias terms, are drawn from distributions. CVNN is initialised with a scaled normal distribution and RVNN using a uniform distribution. The samples fed to the NNs have been normalised in the preprocessing phase and thus no batch normalisation is applied.

The neural networks can have one or more channels, which form another dimension to the weight matrices or kernels. The channels are also occasionally called filters.

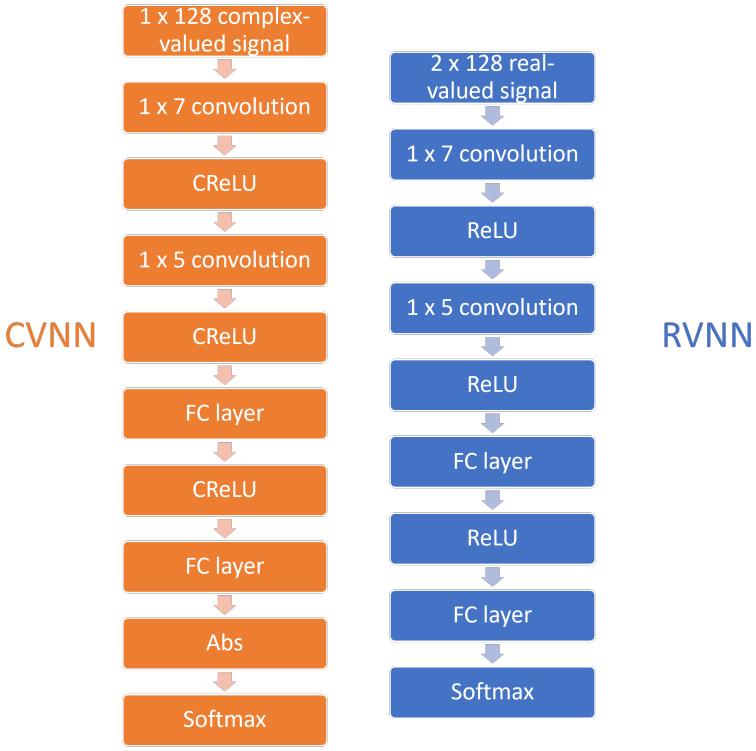

Figure 5.1: Structures of the neural networks run in this thesis.

In order to examine how the number of trainable parameters of an NN affects the performance, the number of filters is chosen as a varying parameter. Therefore, the filter sizes are in the following also called net sizes. Furthermore, in order to study the performance of CVNNs and RVNNs in a fair manner, the number of trainable parameters should be as equal as possible. Hence, the number of filters for CVNN is set as half of the value of RVNN for the second convolutional layer. The employed filter sizes are presented in Table 5.1.

Table 5.1: Values for filter sizes, also called net sizes, used in this thesis. The first filter size corresponds the number of output channels for the first convolutional layer, the second is the number of output channels for the second convolutional layer, and the last is the number of output channels for the first fully connected layer.

Size index	Number of filters for CVNN	Number of filters for RVNN
1	8, 4, 8	8, 8, 8
2	16, 8, 16	16, 16, 16
3	32, 16, 32	32, 32, 32
4	64, 32, 64	64, 64, 64

During training, the network parameters are optimised using Adam optimiser [28] for RVNN and complex Adam [65] for CVNN. The learning rate is determined as 0.01 in the first epoch, so according to the symbols in Equation (3.10), $\eta_0 = 0.01$. After that, the learning rate will decay with factor $\gamma = 0.95$ every epoch, allowing

Table 5.2: Values for hyperparameters in neural networks used in this thesis.

Hyperparameter	Symbol	Value
Learning rate, initial	η_0	0.01
Learning rate, decay coefficient per epoch	γ	0.95
Patience for validation	p_{val}	10
Maximal number of epochs	n_{max}	200
Padding for convolutional layers	p_{pad}	0
Stride for convolutional layers	s	1
Kernel size for convolutional layer 1	1 x λ_1	1 x 7
Kernel size for convolutional layer 2	1 x λ_2	1 x 5
Batch size	β	1024

the optimum to be reached more precisely.

In the NNs, the learning process is monitored by calculating validation accuracy at the end of each epoch. *Early stopping criterion* is utilised, which makes the training stop if the selected criterion, here thus validation accuracy, has not improved in a period determined by *patience* parameter p_{val}, measured in epochs. For the NNs described here, $p_{val} = 10$. On the other hand, the maximal number of epochs $n_{max} = 200$, which means that in case the early stopping criterion has not finished the training in that number of epochs, the stopping takes place then.

The neural network architectures are formed by a vast collection of hyperparameters. Thus, it is challenging to choose a network configuration, since optimising the network performance subject to one hyperparameter may decrease the performance with regard to another hyperparameter. There are typically so many different hyperparameter combinations that all of them cannot be tested, nor can all the dependencies between hyperparameters be modelled. Thus, fully understanding the effect of every hyperparameter on the learning performance is often not feasible. For the sake of reproducibility, values of hyperparameters used here are presented in Table 5.2.

5.2 Data utilised

An essential part of training a neural network is the data fed to the model. In this thesis, RF fingerprint data was simulated using Matlab. QPSK-modulated communications signal with random payload was generated and then filtered with raised cosine filter using samples-per-symbol ratio of 8 and rolloff factor of 0.22. To simulate RF fingerprints, impairments inherent to simulated hardware were added to these signals. Plain IQ-imbalance was considered as an impairment because of its simplicity and inherent non-circularity, and it was added according to [82]:

$$\mathbf{y} = K_1\mathbf{x} + K_2\bar{\mathbf{x}} \tag{5.1}$$

$$K_1 = \frac{1 + ge^{-j\phi}}{2} \tag{5.2}$$

$$K_2 = \frac{1 - ge^{j\phi}}{2}, \tag{5.3}$$

where $\mathbf{x}$ is the unimpaired signal, $\mathbf{y}$ is the impaired signal, g is a coefficient determining amplitude imbalance, ϕ is a coefficient determining phase imbalance, and $\bar{\mathbf{x}}$ is the complex conjugate of $\mathbf{x}$.

When generating the data, 5 different IQ imbalances were chosen, corresponding to 5 simulated emitters. The values for the coefficients g and ϕ presented in Equations (5.1) - (5.3) were set for values given in Table 5.3. The aim was to create signals with different levels of noncircularity associated with different emitter fingerprints even though pure QPSK is a circular waveform. The circularity of the signals can be modelled by using the circularity coefficients defined in [47], page 46, for the generated data. The circularity coefficients of the simulated signals are presented in Table 5.3, and values close to zero indicate circularity, whereas circularity coefficients near to one imply high noncircularity. Some of the employed amplitude and phase imbalance coefficients are unusually large for a practical system but they are used to study the sensitivity of CVNNs and RVNNs to noncircularity,

Table 5.3: IQ imbalance coefficients for the 5 simulated emitters, as well as the resulting circularity coefficients for the generated data with SNR = 15 dB. Circularity coefficients near 0 indicate circularity, whereas values close to 1 indicate high noncircularity.

Emitter index	0	1	2	3	4
ϕ (degrees)	2	4	10	22	45
g	1.6	0.8	2.0	0.5	0.3
Circularity coefficient	0.45	0.24	0.63	0.67	0.92

The data length for each sample was chosen as 128. After the IQ imbalance, additive white Gaussian noise (AWGN) was added to the signals. The power of noise was determined such that signals of signal-to-noise ratios (SNR, see Equation (2.1)) of -5, 0, 5, 10, and 15 dB were obtained. The samples were normalised such that the maximal amplitude of each sample was 1.

The data was formed as inherently complex-valued. However, when fed to the real-valued NN, the real and imaginary parts of the signals were stacked so that a matrix of size 128 x 2 was utilised. On the other hand, the complex-valued neural network naturally allowed complex-valued input and thus complex vectors of size 128 x 1 were fed to it.

The signal samples were split into 3 groups: training, validation, and test sets. In order to study the effect of the amount of training data available, different data sizes were used. The sizes of respective datasets are listed in Table 5.4. Samples

Table 5.4: The data set sizes utilised in this thesis. The sizes are in signal samples of length 128. The given amounts were taken from each class and each SNR.

Size index	Training data	Validation data	Test data
1	400	100	500
2	800	200	1000
3	4000	1000	5000

with different SNR values were mixed such that the NNs would be trained with all of them. Naturally, the NNs were also trained with all of the classes, that is, samples from different simulated emitters.

6 Simulation Results

In this chapter, results comparing complex-valued (CVNN) and real-valued (RVNN) neural networks are presented. The classification performance of the NNs was quantised using mean accuracy. That is, for each run, the PG classification [78] was calculated according to Equation (3.6). The Monte Carlo method [69] was applied such that every hyperparameter combination was run 60 times with different random seeds, and averages of accuracies of these runs were used for results. Also misclassifications were examined.

In the simulations, the simulated RF fingerprint data described in section 5.2 was classified using one complex-valued (CVNN) and one real-valued neural network (RVNN) architecture. The NN structures are described in section 5.1. Both the CVNN and RVNN were run using different net sizes as listed in Table 5.1, as well as feeding them with different amounts of training data, listed in Table 5.4.

The effect of SNR on the classification was examined as well. The SNR values (see Equation (2.1)) of -5 dB, 0 dB, 5 dB, 10 dB, and 15 dB were taken into consideration. The NNs were trained with batches of signals of different SNR values mixed, but the final results were derived separately for each SNR.

In RF fingerprinting applications, the amount of data is critical since it tends to be sparsely available and hard to acquire, and there is no culture of sharing such data among the users or practitioners. Hence, it is often not possible to increase sample sizes in training even though that would improve the classification performance. Instead, the amount of NN parameters can be made larger as long as appropriate computational resources are available. Thus, the amount of training data is often not as large as it should be, and the network size needs to be adjusted to obtain as good results as possible.

6.1 Assumptions

In the simulations, many simplifying assumptions were made. In a real-world scenario, these assumptions would often not hold.

It was assumed in the simulations that the unknown emitter is always one of the known 5 transmitters used in the training data. Naturally, in practical use, also signals transmitted by emitter types that are not in the training set could be observed. One could consider additional classes for emitters that do not belong to the training set as well as to the noise-only data. Data from unknown emitter types could be analysed and then added to the training set if a label can be attached to it. On the other hand, this poses challenges in training as it would be very hard to collect a comprehensive set of unexpected and unknown signals.

Additionally, it was assumed that the signal is received line-of-sight (LOS), and that no multipath due to reflections occurs. In an open space without objects between the transmitter and the receiver, and with directed and potentially well elevated antennas, this could be a fairly reasonable assumption, but it hardly applies to all circumstances such as urban environments with streets and higher buildings.

Another assumption made was that the impairments in the signal are wholly

represented by IQ imbalance. In reality, various analogue components cause imperfections to the signal, for example power amplifier nonlinearities and antenna couplings, and it is a challenging task to model all these factors mathematically. This is also a reason why machine learning models seem so appealing in RFF, since they involve no engineering expertise about all the dependencies between the components in the circuits and the signal, but learn them from the observed data. By simulating data, it was possible to simplify the model thus that the structure of fingerprints was known. This helped understanding the model at the cost of decreasing connections to the real world.

Besides, it was assumed that the noise attached to the signal was purely white Gaussian noise. This is a widely used assumption, however, in vicinity of other transmitters and their interfering emissions, for instance, it might not hold. Especially in dense spectrum use scenarios and demanding propagation environments, interference may be a dominating factor. The RF signal to be classified should be separated from the interference before the classification task.

Furthermore, the communications data utilised was simulated, and its payload was random as it did not need to convey a message. Payload data is typically source coded and all redundancies are removed, hence it has white-noise-like properties. On the other hand, error correcting coding may introduce additional redundancies in the data. The impact of payload data and channel coding were not considered in the simulations.

6.2 Classification Results

First the impact of the training set size was studied with three different sizes listed in Table 5.4, while averaging the results over different net sizes. After that, the classification accuracies of CVNN and RVNN were compared with different training set sizes, still averaging over net sizes. The misclassifications were considered with one of the training set size - net size combinations. Finally, the effect of the net size, and thus also that of the number of parameters, was examined with the six sizes in Table 5.1. All the results have been averaged over 60 random seeds, regardless of whether other averaging has been performed as well.

6.2.1 Effect of training set size

The effect of training data size is presented in Figure 6.1. In the figure, classification accuracies have been averaged over all used net sizes in Table 5.1. It can be seen that, as expected, the increase in the amount of training data leads to better classification accuracy. The increase is slightly more significant for CVNN. However, the RVNN accuracy levels remain well below 0.9 even at high SNR regime. CVNNs have accuracy levels above 0.9 at SNR regime from 5 dB upwards for the largest training data set, at SNR regime from 10 dB upwards for the second largest training set, and at SNR = 15 dB for the smallest training data set as well. Hence, best performance is achieved with CVNNs and a large training data set of signals at high SNR regime.

The average accuracies of the neural networks for the smallest utilised training

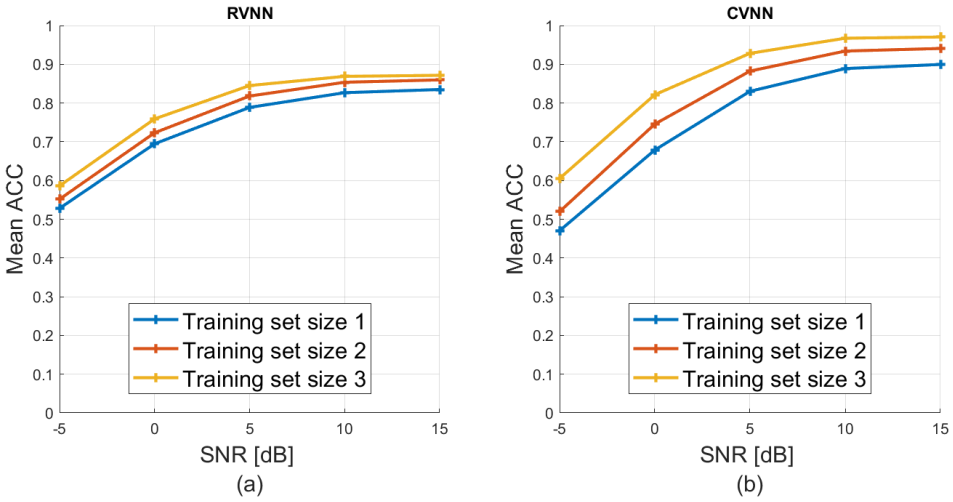

Figure 6.1: Mean classification accuracy performance of CVNN and RVNN for different training set sizes averaged over 6 net sizes, as a function of SNR of signals in the testing set.

set size are presented in Figure 6.2. It can be seen that the CVNN offers better results than RVNN at the higher SNR regime, whereas the order is reversed for low SNRs. However, the performances of the NNs show to be quite similar.

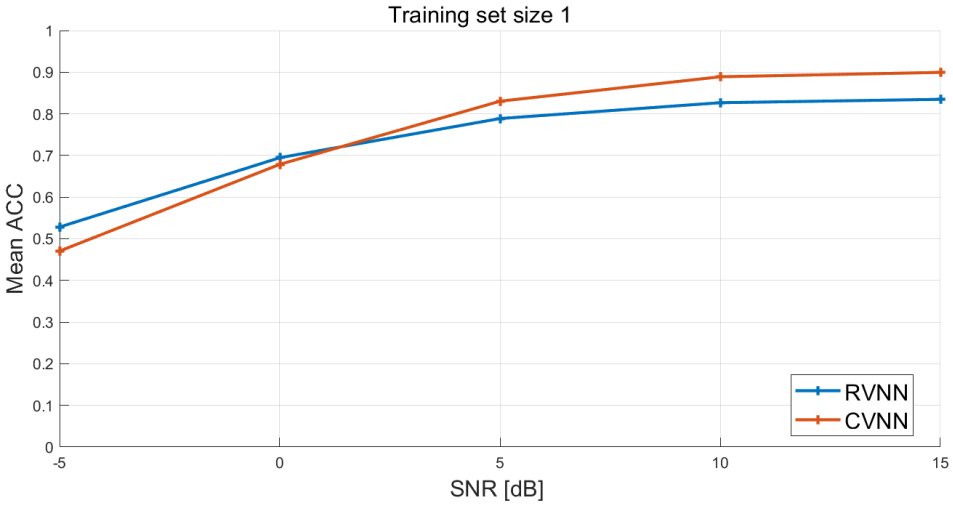

Figure 6.2: Mean classification accuracy performance of CVNN and RVNN when training set size 1 in Table 5.4 is used, and the performance is averaged with regard to 6 net sizes, as a function of SNR of signals in the testing set.

Figure 6.3 presents the performance of the networks for the largest studied training set size, which is ten times the value for training set size 1 in Table 5.4. It can be seen that the CVNN performs better than RVNN in all SNR levels. However, this scenario demands access to a vast collection of training data, which might make it a challenging one. The difficulty of collecting large amounts of labelled training data

was addressed earlier in Section 2.4.

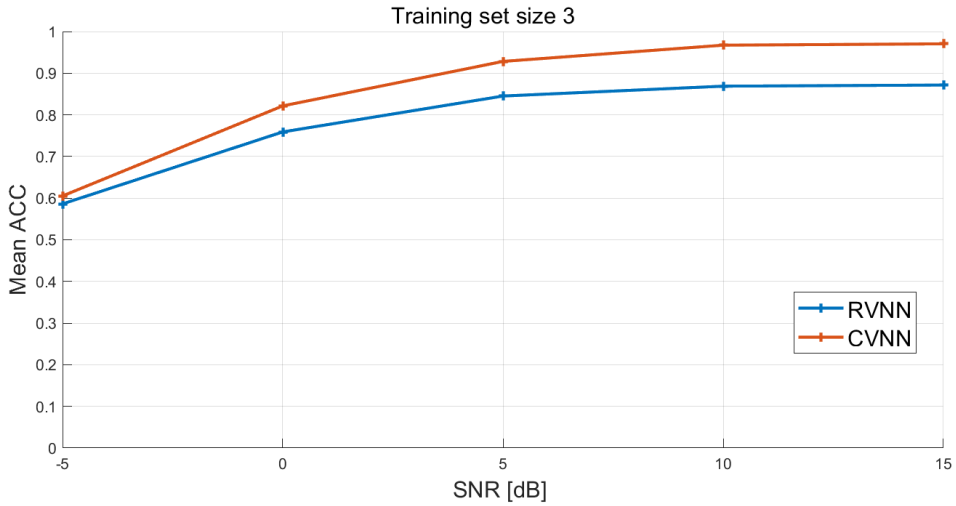

Figure 6.3: Mean classification accuracy performance of CVNN and RVNN when training set size 3 in Table 5.4 is used, and the performance is averaged with regard to 6 net sizes, as a function of SNR of signals in the testing set.

From Figures 6.2 and 6.3, it can be seen that the CVNN works best with regard to RVNN when there is much training data. The larger is the amount of training data, the more competitive option the use of CVNN is instead of RVNN. CVNN is also more favourable at high than low SNRs.

6.2.2 Misclassifications

The classification with net size 4 in Table 5.1 and training data size 1 in Table 5.4 is presented by a confusion matrix in Figure 6.4. In a confusion matrix, each cell contains the number of samples from the class indicated by the row, that were classified to the class indicated by the column. Therefore, a perfect classification would lead to a diagonal matrix where non-zero values were only found from the diagonal. Here, the classifications from 60 runs were summed, and the cell values were normalised such that the sum of each row is 1. From Figure 6.4, it can be seen that in this case the percentages on the CVNN matrix diagonal are larger than on the RVNN diagonal. Thus, CVNN has performed more correct classifications. The mean accuracies for this hyperparameter combination are for CVNN and RVNN 0.95 and 0.83, respectively. From the figure, misclassifications are seen. Both CVNN and RVNN have classified around 10% of class 0 samples to class 2, and approximately 10% vice versa. Additionally, RVNN has classified several samples incorrectly to classes 2 and 3. Classes 0 and 2, according to Table 5.3, both have g values (amplitude imbalances) greater than 1, whereas all the other classes have g values below 1. Thus, the amplitude might dominate the classification and thus the CVNN confuses classes 0 and 2. RVNN, on the other hand, should be less applicable to classify non-circular data, and might therefore make many incorrect classifications.

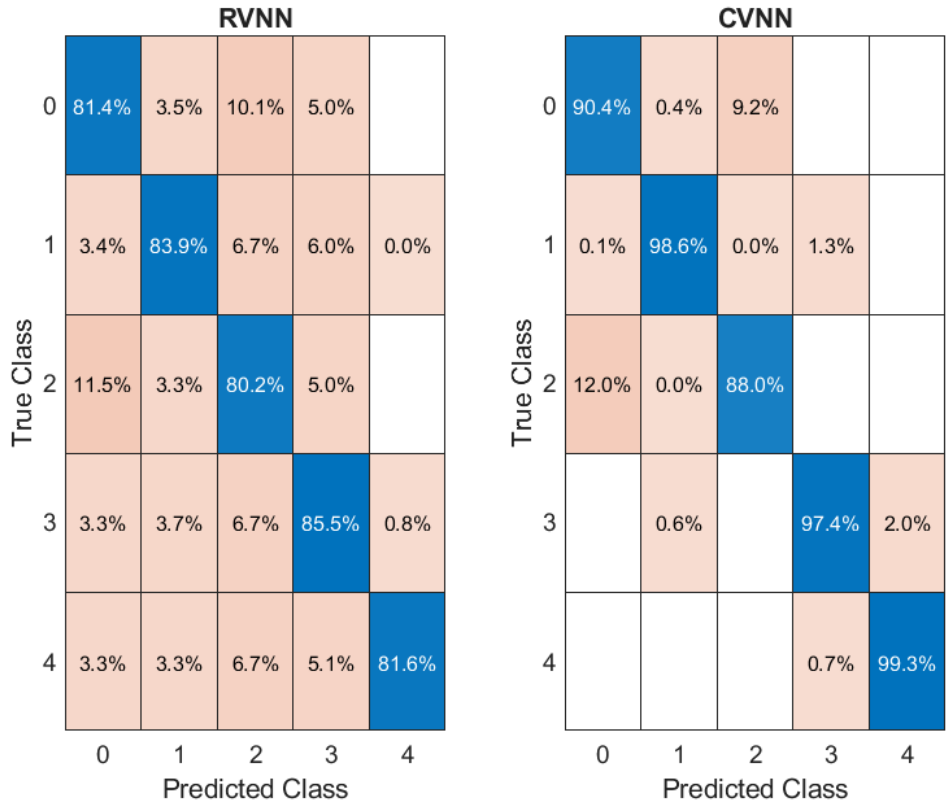

Figure 6.4: Confusion matrices describing classification by RVNN and CVNN. The training set size 1 in Table 5.4 and net size 4 in Table 5.1 are used, and the testing set signals have had SNR of 15 dB. Empty cells indicate that their value is 0.

6.2.3 Effect of net size

The number of parameters of a neural network largely affects its ability to adjust itself to the training data. Figure 6.5 shows the RVNN and CVNN performances for different net sizes. In the figure, training set size 3 in Table 5.4 is used, but similar trends are observed with other training set sizes as well. It can be seen that the performance of the RVNN is highly dependent on the net size, the smallest ones surprisingly yielding best results. On the other hand, CVNN is more robust when the net size is varied. As can be seen from Figure 6.5, at very low SNR levels where noise dominates the data, the performance of the best RVNN net size is better than CVNN with any of the net sizes. This may be due to the fact that noise is complex circular whereas the signal of interest is noncircular in this case. However, for higher SNRs, the best CVNNs outperform the RVNN. When the number of parameters, i.e. net size, is increased, the CVNN performance improves until it decreases a little at largest net sizes. Still, CVNN has accuracy above 0.9 for all network sizes at SNR regime above 5 dB. RVNN, on the other hand, gets accuracy above 0.9 only for the 3 smallest network sizes.

From Figure 6.5 it can be seen as well that for the two smallest net sizes, the

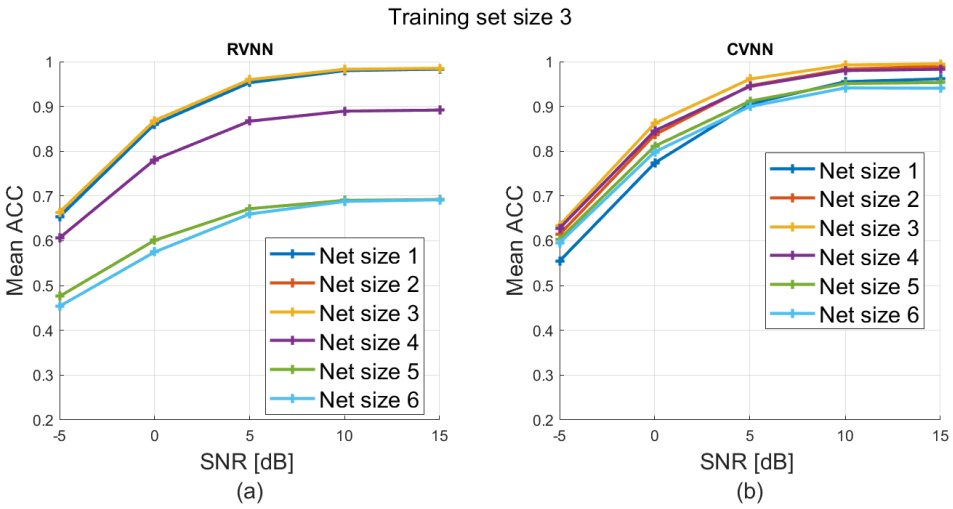

Figure 6.5: Mean classification accuracy performance of CVNN and RVNN when training set size 3 in Table 5.4 is used for different net sizes, as a function of SNR of signals in the testing set. For RVNN, the curve corresponding the net size 2 in Table 5.1 is not visible as it is located under the curve for net size 3.

RVNN performs better than CVNN, although the difference is small for net size 2 in Table 5.1. This result is quite opposite to the hypothesis which suggests that the CVNN performs better than RVNN with a small number of parameters. According to the simulations of this thesis, the performance of CVNN only surpasses that of RVNN with the corresponding net size when the network sizes are large enough. This is at least partly due to the fact that for large network sizes, the learning process of RVNNs becomes unreliable. Many runs end up in a situation where the RVNN classifies all the samples to the same class. However, the only predicted class is not constant but varies between runs. The runs that collapse in that manner lower the average used in results. The same applies to all training data sets tested. On the other hand, the performance of CVNNs is more robust to randomness. As the variability of CVNN performance is much less than that of RVNN, the CVNN user can expect good and predictable performance regardless of the net size.

6.3 Discussion

There are multiple factors that had impact to the simulation results. As neural networks are complicated models with a huge number of design choices and hyperparameters, the listing of factors discussed in the following is by no means comprehensive. However, they try to represent some of the most contributing ones.

The learning performance in all machine learning algorithms is greatly affected by the data fed to the model. The classification accuracy depends on whether there are distinguishable features in the data, and whether or not the training and test sets resemble each other. In this thesis, the training, validation and test sets were all sampled randomly from the same simulated data, which means that the training and test sets were quite similar. On the other hand, it may not be realistic to assume

that the signals fed to the NN in actual RF fingerprinting have been collected in an identical setup than the training data. There were also significant differences between the IQ imbalance values of some of the simulated emitters, which made the classification task easier. Moreover, the training data used here contained uniformly different SNR values, and the most noisy samples probably confused the NNs as any impairments were difficult to extract from them. However, this approach was chosen since in practical situations, the SNR of the unknown signal is not known beforehand to the receiver, but is defined by e.g. transmitting power, propagating channel, and interfering signals.

The amount of the data is of crucial importance as well. The degree of homogeneity among the samples of the same class and the degree of heterogeneity among different classes determine which training data size is sufficient for the NN to be able to classify them. For classes that are easy to be distinguished, small amount of training data should suffice. In this thesis, different training data sizes listed in Table 5.4 were examined, and it was shown in Section 6.2.1 that the classification accuracy improved by increasing the amount of training data. This is an intuitive result, as large amount of training data helps the model to both learn the data and prevent overfitting.

Neural networks can take various forms depending on the character and ordering of their layers. In this thesis, the NN architectures follow in general the one presented for simulations in [42]. Namely, both CVNN and RVNN consist of two convolutional layers, followed by two fully connected layers and softmax classifier. The used kernel sizes presented in Table 5.2 have been previously used in [63]. By adding or removing layers, the depth of the NNs and thus their ability to extract features would change. There exist also methods that help to prevent overfitting, such as max-pool layers or dropout, where the amount of information gathered by the NN is reduced, and that were not used in this thesis. However, their influence would probably be relatively small, as overlearning of the training set was not observed here.

The effect of number of trainable network parameters was examined in this study by running NNs with different filter sizes listed in Table 5.1. The number of trainable parameters is directly proportional to filter and kernel sizes, as well as the amount of layers. Based on the results presented in Section 6.2.3, the size of the neural network affects the results more for RVNNs than for CVNNs. For large RVNNs, a significant amount of runs ends up learning nothing, regardless of the training data set size used, and thus the average accuracies used in results are poor. Furthermore, this study only examined cases where the filter sizes were equal for every RVNN layer as well as first and third filter size for CVNN, and where the middlemost filter size in CVNN was half of that. By changing the proportions of filter sizes in different layers, the layers could be weighted in an alternative manner.

At the parameter update, an optimisation method must be used. In the field of optimisation, there is a numerous amount of different methods. Common alternatives include stochastic gradient descent (SGD) and Adam [28], the latter of which is used in this thesis for RVNN. Adam is used as well in e.g. [63]. The complex-valued variant of Adam [65] is utilised for the CVNN. As Adam is a stochastic optimisation algorithm, its performance is affected by random seeds of the software. However,

stochastic optimisation methods require usually less computations than deterministic ones. Adam is considered robust, but is also known to converge poorly with gradients of large variance. This might lead to the cases where the RVNN does not learn anything with large net sizes, observed in Section 6.2.3, since large filter sizes lead to very small initial parameter values, and computations with tiny numbers may be erroneous. [89]

The parameter initialisation determines from which distribution the initial values of the network parameters, that is, weight matrices of convolution kernels and fully connected layers, along with their bias terms, are drawn. In this thesis, RVNN was initialised with the default uniform distribution used by Pytorch [38], and CVNN used scaled normal initialisation further described in [42]. When different initialisation distributions were tested for this thesis, the two chosen led to the best classification accuracies for respective NNs.

The neural network architectures involve a huge number of hyperparameters that influence the learning process. Hyperparameter design has been studied in e.g. [87]. For instance, different batch sizes affect how much data the model sees simultaneously. However, when tested for this study, the performance was not largely affected by the batch size. The used batch size in Table 5.2 is also used in [63]. Beside batch size, the sample length affects the visibility of the data to the model. Here, sequence length of 128 as in [63] was used. Learning rate is another important hyperparameter that plays a crucial role in convergence of the optimisation algorithm. The initial learning rate, as well as its decay rate, that are both listed in Table 5.2, were chosen so that the optimisation algorithm first proceeds fast near the local optimum and then, due to the decay, approaches the optimum more precisely at every epoch. The initial learning rate is also the default for Pytorch [39]. The convergence is also affected by validation patience, since it determines whether or not the training will proceed. The used patience value in Table 5.2 is utilised also in [63]. The maximal number of epochs in Table 5.2 was chosen such that it would be large enough to allow stopping based on patience in most cases. Training should stop on maximal number of epochs only if the convergence takes unreasonably long.

It is not feasible to examine all the possible hyperparameter combinations in order to find one providing the best performance. On the other hand, since the main task here was to compare the performance of CVNN to that of RVNN, it suffices to ensure that the corresponding hyperparameter values are comparable, even though they might not be the ones that lead to best possible performance. However, such hyperparameter combinations have been sought that provide useful classification accuracies.

Neural networks are heuristic methods, and randomness is widely present. The random seeds affect e.g. the initial values of the network parameters sampled from distributions, the splitting of the data to training, validation, and test sets, and forming of batches in training. Besides, when run on separate computers, the results may vary a little. To minimise the effect of randomness, the simulations in this thesis have been run on 60 random seeds, and averages on these runs have been employed when plotting the actual results.

There are multiple differences between the CVNN and the RVNN although the

aim was to construct them such that they would be as comparable as possible. Due to the different dimensions of the input data, as well as the differences between complex- and real-valued calculus, the NN architectures could not be exactly identical. Additionally, the programming software for building the NNs were different, as the CVNN functions had been built from scratch whereas RVNNs are available in most NN software packages, and thereby the internal implementations of various functions and methods most probably differ from each other.

Based on the results of this thesis, CVNNs are a more reliable and less variable RF fingerprinting technique than RVNNs. Based on Figures 6.2-6.3 with results averaged over net sizes, the mean classification accuracy of CVNN surpasses that of RVNN at SNR regime above 5 dB, and it is often above 0.9 whereas RVNN accuracies remain below 0.9. However, RVNNs obtain good results with accuracy above 0.9 at SNR regime from 5 dB upwards, but only with small network sizes, as shown in Figure 6.5. It is difficult to know in advance the optimal network size, and therefore it is recommended to use CVNN as it has shown its robustness with regard to different network sizes. RVNN is more sensitive to parameter initialisation determined by random seeds and there is thus much variation it its classification accuracies for larger network sizes. As there exist numerous other NN architectures than the ones used, there are many uncertainties concerning the hyperparameter choices. Thus, the robustness of CVNN towards these design choices makes it a practicable method for RF fingerprint classification.

7 Conclusion

In this Master's thesis, neural networks were applied for classification of simulated RF fingerprint data. A conventional, real-valued neural network (RVNN) was compared to a complex-valued neural network (CVNN) that utilises complex-valued differentiation according to Wirtinger calculus. It was shown that the complex-valued neural network classified the data in average better than the real-valued one, especially in high SNR region. However, it was also discovered that the different hyperparameters have a great influence on the classification accuracy, and therefore there is much variation in NN performances as well. Nonetheless, the results show that change in the number of NN parameters affects the CVNN less than RVNN. On the other hand, increasing the training data size improves the CVNN more than it does RVNN. When averaging the accuracies over different network sizes, CVNNs overall showed better performance than RVNNs with accuracy levels above 0.9 at high SNR regime.

As the practical RF fingerprinting scenarios often have a limited amount of training data, it is more essential to be able to choose an appropriate network size than a data size. Since the CVNN is more robust with regard to the NN size, it offers a more reliable option for RF fingerprint classification.

It remains for future work to examine how different kinds of fingerprints, rather than the ones with IQ imbalance as the impairment employed here, get classified. Non-simulated signals from publicly available RF fingerprinting datasets could thus be used with the developed methods. Many impairments, such as amplifier nonidealities or turn-on transients, work usually better in creating differences between emitters than IQ imbalance. Different neural network architectures, such as recurrent neural networks (RNN) and in particular long short-term memory (LSTM) suitable for time-series data, remain to be explored. There are also multiple hyperparameters, the effect of which on the results should be further studied. Additionally, there exist methods that combine results from different neural networks and thus the great variability and poor predictability of classification accuracies of RVNN could possibly be compensated, but this demands further studies as well. A neural network could even be trained to compensate nonidealities present in signals by learning the inverse function of the impairments.

References

[1] Abien Fred Agarap. Deep Learning using Rectified Linear Units (ReLU), 2019.

[2] Amani Al-Shawabka et al. Exposing the fingerprint: Dissecting the impact of the wireless channel on radio fingerprinting. *Proceedings of IEEE Conference on Computer Communications (INFOCOM)*, 2020.

[3] P Preenu Ann and Renu Jose. Comparison of PAPR reduction techniques in OFDM systems. In *2016 International Conference on Communication and Electronics Systems (ICCES)*, pages 1–5. IEEE, 2016.

[4] Pierre Baldi and Roman Vershynin. The capacity of feedforward neural networks. *Neural Networks*, 116:288–311, 2019.

[5] Pedro Ballester and Ricardo Araujo. On the performance of GoogLeNet and AlexNet applied to sketches. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 30, 2016.

[6] Crystal Bertoncini et al. Wavelet fingerprinting of radio-frequency identification (RFID) tags. *IEEE Transactions on Industrial Electronics*, 59(12):4843–4850, 2011.

[7] Daniele Borio et al. Impact and detection of GNSS jammers on consumer grade satellite navigation receivers. *Proceedings of the IEEE*, 104(6):1233–1245, 2016.

[8] Vladimir Brik et al. Wireless device identification with radiometric signatures. In *Proceedings of the 14th ACM International Conference on Mobile Computing and Networking*, pages 116–127, 2008.

[9] Howard C Choe et al. Novel identification of intercepted signals from unknown radio transmitters. In *Wavelet Applications II*, volume 2491, pages 504–517. SPIE, 1995.

[10] Boris Danev, Davide Zanetti, and Srdjan Capkun. On physical-layer identification of wireless devices. *ACM Computing Surveys*, 45(1), dec 2012. ISSN 0360-0300. doi: 10.1145/2379776.2379782. URL https://doi.org/10.1145/2379776.2379782.

[11] Giovani Dematos et al. Feedforward versus recurrent neural networks for forecasting monthly japanese yen exchange rates. *Financial Engineering and the Japanese Markets*, 3:59–75, 1996.

[12] Lida Ding et al. Specific emitter identification via convolutional neural networks. *IEEE Communications Letters*, 22(12):2591–2594, 2018.

[13] George H Dunteman. *Principal Components Analysis*, volume 69. Sage, 1989.

[14] Bat-Sheva Einbinder et al. Training uncertainty-aware classifiers with conformalized deep learning. *Advances in Neural Information Processing Systems*, 35: 22380–22395, 2022.

[15] Abdurrahman Elmaghbub and Bechir Hamdaoui. LoRa device fingerprinting in the wild: Disclosing RF data-driven fingerprint sensitivity to deployment variability. *IEEE Access*, 9:142893–142909, 2021.

[16] Jan Eriksson, Esa Ollila, and Visa Koivunen. Essential statistics and tools for complex random variables. *IEEE Transactions on Signal Processing*, 58(10): 5400–5408, 2010. doi: 10.1109/TSP.2010.2054085.

[17] Ryan Gerdes et al. Device identification via analog signal fingerprinting: A matched filter approach. 2006.

[18] Chuan Guo et al. On calibration of modern neural networks. In *Proceedings of the 34th International Conference on Machine Learning*, pages 1321–1330, Sydney, 2017. PMLR.

[19] Jeyanthi Hall, Michel Barbeau, and Evangelos Kranakis. Detection of transient in radio frequency fingerprinting using signal phase. *Wireless and Optical Communications*, pages 13–18, 2003.

[20] Jeyanthi Hall, Michel Barbeau, and Evangelos Kranakis. Enhancing intrusion detection in wireless networks using radio frequency fingerprinting. *Communications, Internet, and Information Technology*, 1, 2004.

[21] Jeyanthi Hall, Michel Barbeau, and Evangelos Kranakis. Radio frequency fingerprinting for intrusion detection in wireless networks. *IEEE Transactions on Defendable and Secure Computing*, 12:1–35, 2005.

[22] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

[23] Andrew Hunter. Feature selection using probabilistic neural networks. *Neural Computing & Applications*, 9:124–132, 2000.

[24] Anu Jagannath, Jithin Jagannath, and Prem Sagar Pattanshetty Vasanth Kumar. A comprehensive survey on radio frequency (RF) fingerprinting: Traditional approaches, deep learning, and open challenges. *Computer Networks*, 219: 109455, 2022.

[25] Jithin Jagannath et al. Deep learning and reinforcement learning for autonomous unmanned aerial systems: Roadmap for theory to deployment. In *Deep Learning for Unmanned Systems*, volume 984 of *Studies in Computational Intelligence*, pages 25–82. Springer International Publishing, Cham, 2021. ISBN 978-3-030-77939-9. doi: 10.1007/978-3-030-77939-9_2. URL https://doi.org/10.1007/978-3-030-77939-9_2.

[26] Katarzyna Janocha and Wojciech Marian Czarnecki. On loss functions for deep neural networks in classification. *arXiv preprint arXiv:1702.05659*, 2017.

[27] Irwin O Kennedy et al. Radio transmitter fingerprinting: A steady state frequency domain approach. In *2008 IEEE 68th Vehicular Technology Conference*, pages 1–5. IEEE, 2008.

[28] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[29] Teuvo Kohonen. The self-organizing map. *Proceedings of the IEEE*, 78(9): 1464–1480, 1990.

[30] Simon Kornblith et al. Why do better loss functions lead to less transferable features? *Advances in Neural Information Processing Systems*, 34:28648–28662, 2021.

[31] Jyri Kosola and Tero Solante. *Digitaalinen taistelukenttä. Informaatioajan sotakoneen tekniikka.* Kolmas painos. Maanpuolustuskorkeakoulu, Sotatekniikan Laitos, Helsinki, 2013. 491 pages.

[32] Oliver Kramer. K-nearest neighbors. In *Dimensionality Reduction with Unsupervised Nearest Neighbors*, volume 51 of *Intelligent Systems Reference Library*, pages 13–23. Springer, Berlin, Heidelberg, 2013.

[33] Ken Kreutz-Delgado. The complex gradient operator and the $\mathbb{CR}$-calculus. *arXiv preprint arXiv:0906.4835*, 2009.

[34] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in Neural Information Processing Systems*, 25, 2012.

[35] Päivi Lakka (Yle). Ulkoministeri Valtonen Venäjän aiheuttamista GPS-häiriöistä: ”Mittakaava on laajentunut ja häirintä on muuttanut muotoaan”. https://yle.fi/a/74-20086939, 2024. Accessed: 6.5.2024.

[36] Chulhee Lee and David A Landgrebe. Feature extraction based on decision boundaries. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(4):388–400, 1993.

[37] Shutao Li et al. Deep learning for hyperspectral image classification: An overview. *IEEE Transactions on Geoscience and Remote Sensing*, 57(9):6690–6709, 2019.

[38] The Linux Foundation. Pytorch documentation: Conv1d. https://pytorch.org/docs/stable/generated/torch.nn.Conv1d.html#torch.nn.Conv1d, . Accessed: 14.5.2024.

[39] The Linux Foundation. Pytorch documentation: Torch.optim. https://pytorch.org/docs/stable/optim.html, . Accessed: 14.5.2024.

[40] Jarmo Lundén and Visa Koivunen. Deep learning for HRRP-based target recognition in multistatic radar systems. In *2016 IEEE Radar Conference (RadarConf)*, pages 1–6. IEEE, 2016.

[41] Stephen Marsland. *Machine Learning: An Algorithmic Perspective*. CRC press, 2015.

[42] Maarit Melvasalo and Visa Koivunen. Complex-valued neural networks for radio frequency machine learning. *Submitted*, 2023.

[43] Mehryar Mohri, Afshin Rostamizadeh, and Ameet Talwalkar. *Foundations of Machine Learning*. MIT press, 2018.

[44] Kevin P. Murphy. *Probabilistic Machine Learning: An Introduction*. MIT Press, 2022. URL probml.ai.

[45] Vladimir Nasteski. An overview of the supervised machine learning methods. *Horizons - International Scientific Journal (Series B)*, 4:51–62, 2017.

[46] Chrysostomos L Nikias and Mysore R Raghuveer. Bispectrum estimation: A digital signal processing framework. *Proceedings of the IEEE*, 75(7):869–891, 1987.

[47] Esa Ollila. *Contributions to Independent Component Analysis, Sensor Array and Complex Valued Signal Processing*. Doctoral thesis, 2010. URL http://urn.fi/URN:ISBN:978-952-60-3031-9.

[48] Esa Ollila and Visa Koivunen. Generalized complex elliptical distributions. In *Processing Workshop Proceedings, 2004 Sensor Array and Multichannel Signal*, pages 460–464. IEEE, 2004.

[49] Esa Ollila et al. Complex elliptically symmetric distributions: Survey, new results and applications. *IEEE Transactions on Signal Processing*, 60(11): 5597–5625, 2012. doi: 10.1109/TSP.2012.2212433.

[50] Sangwoo Park, Kfir M. Cohen, and Osvaldo Simeone. Few-shot calibration of set predictors via meta-learned cross-validation-based conformal prediction. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 46(1):280–291, 2024. doi: 10.1109/TPAMI.2023.3327300.

[51] Josh Patterson and Adam Gibson. *Deep Learning: A Practitioner's Approach*. O'Reilly Media, Inc., 2017.

[52] SJ Pawan and Jeny Rajan. Capsule networks for image classification: A review. *Neurocomputing*, 509:102–120, 2022.

[53] Bernard Picinbono. On circularity. *IEEE Transactions on Signal Processing*, 42(12):3473–3482, 1994.

[54] Bernard Picinbono. Second-order complex random vectors and normal distributions. *IEEE Transactions on Signal Processing*, 44(10):2637–2640, 1996.

[55] Bernard Picinbono and Pascal Chevalier. Widely linear estimation with complex data. *IEEE Transactions on Signal Processing*, 43(8):2030–2033, 1995.

[56] Yasir Rahmatallah and Seshadri Mohan. Peak-to-average power ratio reduction in OFDM systems: A survey and taxonomy. *IEEE Communications Surveys & Tutorials*, 15(4):1567–1592, 2013.

[57] Saeed Ur Rehman et al. Radio frequency fingerprinting and its challenges. In *2014 IEEE Conference on Communications and Network Security*, pages 496–497. IEEE, 2014.

[58] Guillem Reus-Muns et al. Trust in 5G open RANs through machine learning: RF fingerprinting on the POWDER PAWR platform. In *IEEE Globecom 2020-IEEE Global Communications Conference*. IEEE, 2020.

[59] Brian D Ripley. *Pattern Recognition and Neural Networks*. Cambridge university press, 2007.

[60] Debashri Roy et al. Detection of rogue RF transmitters using generative adversarial nets. In *2019 IEEE Wireless Communications and Networking Conference (WCNC)*, pages 1–7. IEEE, 2019.

[61] David E Rumelhart et al. Backpropagation: The basic theory. *Backpropagation: Theory, Architectures and Applications*, pages 1–34, 1995.

[62] Kunal Sankhe et al. ORACLE: Optimized radio classification through convolutional neural networks. In *IEEE INFOCOM 2019-IEEE Conference on Computer Communications*, pages 370–378. IEEE, 2019.

[63] Kunal Sankhe et al. No radio left behind: Radio fingerprinting through deep learning of physical-layer hardware impairments. *IEEE Transactions on Cognitive Communications and Networking*, 6(1):165–178, 2019.

[64] Renuka Saravanan and Pothula Sujatha. A state of art techniques on machine learning algorithms: a perspective of supervised learning approaches in data classification. In *2018 Second International Conference on Intelligent Computing and Control Systems (ICICCS)*, pages 945–949. IEEE, 2018.

[65] Andy M Sarroff. *Complex Neural Networks for Audio*. PhD thesis, Dartmouth College, 2018.

[66] Robert E Schapire. A brief introduction to boosting. In *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence*, volume 99, pages 1401–1406. Citeseer, 1999.

[67] D. Shaw and W. Kinsner. Multifractal modelling of radio transmitter transients for classification. In *IEEE WESCANEX 97 Communications, Power and Computing. Conference Proceedings*, pages 306–312, 1997. doi: 10.1109/WESCAN.1997.627159.

[68] Guanxiong Shen et al. Toward length-versatile and noise-robust radio frequency fingerprint identification. *IEEE Transactions on Information Forensics and Security*, 18:2355–2367, 2023. doi: 10.1109/TIFS.2023.3266626.

[69] Yu A Shreider. *The Monte Carlo Method: The Method of Statistical Trials*, volume 87. Elsevier, 2014.

[70] Bernard Sklar. *Digital Communications. Fundamentals and Applications*. Prentice Hall PTR, New Jersey, USA, 2001.

[71] Nasim Soltani et al. RF fingerprinting unmanned aerial vehicles with non-standard transmitter waveforms. 2020.

[72] Naeimeh Soltanieh et al. A review of radio frequency fingerprinting techniques. *IEEE Journal of Radio Frequency Identification*, 4(3):222–233, 2020.

[73] Donald F Specht. A general regression neural network. *IEEE Transactions on Neural Networks*, 2(6):568–576, 1991.

[74] Daniel Svozil, Vladimír Kvasnicka, and Jirí Pospichal. Introduction to multi-layer feed-forward neural networks. *Chemometrics and Intelligent Laboratory Systems*, 39(1):43–62, 1997. ISSN 0169-7439. doi: https://doi.org/10.1016/S0169-7439(97)00061-0. URL https://www.sciencedirect.com/science/article/pii/S0169743997000610.

[75] Christian Szegedy et al. Going deeper with convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–9, 2015.

[76] J Toonstra and Wintold Kinsner. Transient analysis and genetic algorithms for classification. In *IEEE WESCANEX 95. Communications, Power, and Computing. Conference Proceedings*, volume 2, pages 432–437. IEEE, 1995.

[77] Chiheb Trabelsi et al. Deep complex networks. *arXiv preprint arXiv:1705.09792*, 2017.

[78] Janet M Twomey and Alice E Smith. Validation and verification. *Artificial Neural Networks for Civil Engineers: Fundamentals and Applications*, pages 44–64, 1997.

[79] Oktay Ureten and Nur Serinken. Bayesian detection of radio transmitter turn-on transients. In *NSIp*, pages 830–834, 1999.

[80] Oktay Ureten and Nur Serinken. Bayesian detection of wi-fi transmitter RF fingerprints. *Electronics Letters*, 41(6):373–374, 2005.

[81] Oktay Ureten and Nur Serinken. Wireless security through RF fingerprinting. *Canadian Journal of Electrical and Computer Engineering*, 32(1):27–33, 2007.

[82] Mikko Valkama, Markku Renfors, and Visa Koivunen. Advanced methods for I/Q imbalance compensation in communication receivers. *IEEE Transactions on Signal Processing*, 49(10):2335–2344, 2001.

[83] Xianpeng Wang et al. Polarization channel estimation for circular and non-circular signals in massive MIMO systems. *IEEE Journal of Selected Topics in Signal Processing*, 13(5):1001–1016, 2019. doi: 10.1109/JSTSP.2019.2925786.

[84] Qingyang Wu et al. Deep learning based RF fingerprinting for device identification and wireless security. *Electronics Letters*, 54(24):1405–1407, 2018.

[85] Rui Xu and Donald Wunsch. Survey of clustering algorithms. *IEEE Transactions on Neural Networks*, 16(3):645–678, 2005.

[86] Simon Yiu et al. Wireless RSSI fingerprinting localization. *Signal Processing*, 131:235–244, 2017.

[87] M. Todd Young et al. HyperSpace: Distributed bayesian hyperparameter optimization. *Proceedings (Symposium on Computer Architecture and High Performance Computing)*, September 2018. doi: 10.1109/CAHPC.2018.8645954.

[88] Yong Yu et al. A review of recurrent neural networks: LSTM cells and network architectures. *Neural Computation*, 31(7):1235–1270, 2019.

[89] Aston Zhang et al. Dive into Deep Learning. *arXiv preprint arXiv:2106.11342*, 2021.

[90] Junqing Zhang et al. Radio frequency fingerprint identification for narrowband systems, modelling and classification. *IEEE Transactions on Information Forensics and Security*, 16:3974–3987, 2021.