
UNIVERSITY OF THESSALY

SCHOOL OF ENGINEERING

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

# Neural Network Architectures for Jammer Signal Classification from Radio-Interferometric Data

Diploma Thesis

**Christos Andreas Ntemkas**

**Supervisor:** Antonios Argyriou

July 2024


UNIVERSITY OF THESSALY

SCHOOL OF ENGINEERING

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

# Neural Network Architectures for Jammer Signal Classification from Radio-Interferometric Data

Diploma Thesis

**Christos Andreas Ntemkas**

**Supervisor:** Antonios Argyriou

July 2024


ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ

ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ

ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

# Αρχιτεκτονικές Νευρωνικών Δικτύων για Ταξινόμηση Σημάτων Παρεμβολέων από Ραδιο-Συμβολομετρικά Δεδομένα

Διπλωματική Εργασία

**Χρήστος Ανδρέας Ντέμκας**

**Επιβλέπων:** Αντώνιος Αργυρίου

Ιούλιος 2024

Approved by the Examination Committee:

Supervisor **Antonios Argyriou**
Associate Professor, Department of Electrical and Computer Engineering, University of Thessaly

Member **Dimitrios Katsaros**
Associate Professor, Department of Electrical and Computer Engineering, University of Thessaly

Member **Gerasimos Potamianos**
Associate Professor, Department of Electrical and Computer Engineering, University of Thessaly

# Acknowledgements

To my family

# DISCLAIMER ON ACADEMIC ETHICS AND INTELLECTUAL PROPERTY RIGHTS

«Being fully aware of the implications of copyright laws, I expressly state that this diploma thesis, as well as the electronic files and source codes developed or modified in the course of this thesis, are solely the product of my personal work and do not infringe any rights of intellectual property, personality and personal data of third parties, do not contain work / contributions of third parties for which the permission of the authors / beneficiaries is required and are not a product of partial or complete plagiarism, while the sources used are limited to the bibliographic references only and meet the rules of scientific citing. The points where I have used ideas, text, files and / or sources of other authors are clearly mentioned in the text with the appropriate citation and the relevant complete reference is included in the bibliographic references section. I also declare that the results of the work have not been used to obtain another degree. I fully, individually and personally undertake all legal and administrative consequences that may arise in the event that it is proven, in the course of time, that this thesis or part of it does not belong to me because it is a product of plagiarism».

The declerant

Christos Andreas Ntemkas

Diploma Thesis

**Neural Network Architectures for Jammer Signal Classification from Radio-Interferometric Data**

**Christos Andreas Ntemkas**

# Abstract

The study aims to develop a Deep Neural Network (DNN) capable of accurately classifying the presence of one or more RF jammers in radar signals measured with an array of antennas. The Radar signals undergo processing to extract target elevation and azimuth information with a software interferometer that calculates the 2D Discrete Fourier Transform (DFT) of the cross-correlation between the antenna signals. RF jammers are treated as anomalies in the DFT data and for this reason we deploy an LSTM Autoencoder that is specialized for anomaly detection in time series data.

To evalue our idea we produce an imaging dataset of the 2D-DFTs is produced by yielding 60 images that replicate 60 seconds of radar signals. When tested in a scenario where two jamming signals are present, the Deep Learning model trained on the dataset achieves an accuracy of almost 98% under the optimal Signal-to-Noise Ratio (SNR). Nevertheless, the model shows decreased precision in differentiating jammers at intervals of less than 10 seconds and beyond 45 seconds. The model exhibits good detection accuracy of jammers in the remaining time intervals.

**Keywords:**

RADAR, Deep Learning, Long Short-Time Memory, jammer, Direction Of Arrival, Uniform Rectangular Array, Pulse RADAR

Διπλωματική Εργασία

**Αρχιτεκτονικές Νευρωνικών Δικτύων για Ταξινόμηση Σημάτων Παρεμβολέων από Ραδιο-Συμβολομετρικά Δεδομένα**

**Χρήστος Ανδρέας Ντέμκας**

# Περίληψη

Στόχος της μελέτης είναι η ανάπτυξη ενός βαθύ νευρωνικού δικτύου (DNN) που θα μπορεί να ταξινομεί με ακρίβεια την παρουσία ενός ή περισσότερων παρεμβολέων RF σε σήματα ραντάρ που μετρώνται με μια συστοιχία κεραιών. Τα σήματα ραντάρ υποβάλλονται σε επεξεργασία για την εξαγωγή πληροφοριών ανύψωσης και αζιμούθιου του στόχου με ένα παρεμβολόμετρο λογισμικού που υπολογίζει τον δισδιάστατο διακριτό μετασχηματισμό Fourier (DFT) της διασταυρούμενης συσχέτισης μεταξύ των σημάτων των κεραιών. Οι παρεμβολείς RF αντιμετωπίζονται ως ανωμαλίες στα δεδομένα DFT και για το λόγο αυτό αναπτύσσουμε έναν αυτόματο κωδικοποιητή LSTM που είναι εξειδικευμένος για την ανίχνευση ανωμαλιών σε δεδομένα χρονοσειρών.

Για να αξιολογήσουμε την ιδέα μας παράγουμε ένα σύνολο δεδομένων απεικόνισης των 2D-DFTs παράγεται με την παραγωγή 60 εικόνων που αναπαράγουν 60 δευτερόλεπτα σημάτων ραντάρ. Όταν δοκιμάζεται σε ένα σενάριο όπου υπάρχουν δύο σήματα παρεμβολής, το μοντέλο Deep Learning που εκπαιδεύεται στο σύνολο δεδομένων επιτυγχάνει ακρίβεια σχεδόν 98% υπό τον βέλτιστο λόγο σήματος προς θόρυβο (SNR). Ωστόσο, το μοντέλο παρουσιάζει μειωμένη ακρίβεια στη διαφοροποίηση των παρεμβολέων σε διαστήματα μικρότερα των 10 δευτερολέπτων και πέραν των 45 δευτερολέπτων. Το μοντέλο παρουσιάζει καλή ακρίβεια ανίχνευσης των παρεμβολέων στα υπόλοιπα χρονικά διαστήματα.

**Λέξεις-κλειδιά:**

Ραντάρ, Βαθειά Μάθηση, Μακρά Μάθηση Σύντομου Χρόνου, Παρεμβολέας, Κατεύθυνσης Άφιξης, Ομοιόμορφη Ορθογώνια Διάταξη, Παλμικό ραντάρ

# Table of contents

# List of figures

# List of tables

# Abbreviations

| | |
|---|---|
| RADAR | RAdio Detection And Ranging |
| CW | Continuous Wave |
| FMCW | Frequency Modulated Continuous Wave |
| RF | Radio Frequency |
| LFM | Linear Frequency Modulated |
| PRI | Pulse Repetition Interval |
| ULA | Uniform Linear Array |
| URA | Uniform Rectangular Array |
| DOA | Direction of Interval |
| AWGN | Additive White Gaussian Noise |
| SNR | Signal to Noise Ratio |
| DFT | Discrete Fourier Transform |
| ML | Machine Learning |
| DL | Deep Learning |
| DNN | Deep Neural Network |
| RNN | Recurrent Neural Network |
| LSTM | Long Short-Term Memory |
| CNN | Convolution Neural Networks |
| BPTT | Backpropagation Through Time |
| RTRL | Real-Time Recurrent Learning |
| FFNN | Feed-Forward Neural Network |
| CEC | Constant Error Carousel |
| SSE | Sum of Square Errors |
| OFDM | Orthogonal Frequency Division Multiplexing |
| LiDar | Light Detection and Ranging |

# Chapter 1

# Introduction

The spread of radar applications has increased dramatically in recent years due to the quick development of autonomous systems like unmanned aerial vehicles and self-driving cars. To improve their autonomy and safety features, these technologies mainly rely on radar sensors for navigation, object detection, and environmental sensing. The increasing need for resilient radar systems that can reliably identify and categorize objects in intricate and ever-changing settings highlights the vital significance of creating sophisticated signal processing and machine learning methods. In order to overcome these obstacles, our work focuses on creating a Deep Neural Network (DNN) that can identify and categorize anomalies in radar operations, such as jamming signals. This research attempts to greatly improve the effectiveness, dependability, and security of radar-based systems in a variety of areas by utilizing contemporary machine learning techniques.

## 1.1 Thesis Subject

The purpose of this thesis is to create a Deep Neural Network (DNN) that can identify malicious activity attempts by jammers on different kinds of radar systems. Currently found in all forms of transportation (vehicles, boats, and aircraft), radar systems are vulnerable to malevolent tampering, which could result in disastrous events like aircraft crashes or abrupt vehicle stops. This paper suggests using a DNN, more precisely an LSTM Autoencoder, trained on image sequences obtained from real RADAR signals. The goal is to distinguish between the presence and absence of jammers in RADAR operations with a high detection rate. This research attempts to strengthen the security and dependability of RADAR-enabled

applications across several sectors by improving the capacity of RADAR systems to detect and mitigate such threats.

### 1.1.1 Contribution

The contribution of this thesis are stared as follows:

1. First implementation of LSTM Autoencoder for RADAR jammer detection
2. High accuracy on finding the number of the jammer hitting in a time frame of 60 seconds and above average accuracy for most positions in this time frame.

## 1.2 Structure

The thesis is structured into five chapters. In Chapter 2, the reader is introduced to RADAR systems through an explanation of their working principles and a discussion of several methodologies relevant to this study. The fundamental underpinnings of deep learning are covered in Chapter 3, along with neural network topologies, training techniques, and pertinent theoretical ideas that are important to comprehend the chapters that follow. The case study and the full implementation of the suggested DNN application for RADAR system jammer detection are covered in detail in Chapter 4. The experimental setup, data pretreatment methods, model architecture design, and deployment considerations are all explained in this chapter. Last but not least, Chapter 5 offers a thorough examination of the results derived from the trials carried out in Chapter 4. This chapter contains performance comparisons, evaluation metrics, and an extensive discussion of the results. Furthermore, Chapter 5 offers an analysis of the findings' consequences, their importance concerning RADAR security, and possible directions for further investigation.

# Chapter 2

# Introduction to RADAR

## 2.1 Introduction

## 2.2 Meaning and Importance

RADAR stands for RAdio Detection And Ranging and is an electromagnetic system for the detection and location of objects. It operates by transmitting a particular type of waveform, a pulse-modulated sine wave for example, and detects the nature of the echo signal. Radar is used to extend the capability of one's senses for observing the environment, especially the sense of vision. The value of radar lies not in being a substitute for the eye, but in doing what the eye cannot do ... Radar cannot resolve detail as well as the eye, nor is it capable of recognizing the" color" of objects to the degree of sophistication of which the eye is capable. However, radar can be designed to see through those conditions impervious to normal human vision, such as darkness, haze, fog, snow, even non-conductive materials, such as walls, to detect and locate objects on the other side. In addition, radar has the advantage of being able to measure the distance, range, azimuth and elevation of the target. This is what is makes RADAR so useful in a variety of applications.

## 2.3 Applications

It is widely agreed [8],[9] that the RADAR systems have been invented for military reasons. Even though it had and has a pivotal role in military, RADAR has to offer more for us:

- **Remote Sensing** The most general purpose of RADAR. In the past, it was used to ”take photos” of the moon and other planets. Similarly, weather analysis and prediction has been a use for it as well
- **Air Traffic Control and Aircraft Navigation** Worldwide, radar systems are used for safe air traffic control, both in and around airports. At large airports, aircraft and ground vehicle traffic are monitored by high-resolution radar. Radar is used by Ground-Controlled Approach (GCA) systems to direct airplanes toward safe landings in inclement weather. The widely utilized Air Traffic Control (ATC) radar-beacon system and the microwave landing system are also supported by radar technology. Furthermore, in aircraft navigation, weather-avoidance radar enables pilots to detect precipitation. Radar also facilitates terrain avoidance and terrain following
- **Ship Safety** By identifying navigation markers and providing a warning of possible collisions with other ships, particularly in low visibility, radar improves maritime safety. In terms of numbers, this is a major use for radar, although it is not very large or expensive. Moreover, it is among the most dependable radar systems. Plot extractors, which are automatic tracking and detection devices, are sold commercially for the purpose of preventing collisions. In addition, harbor navigation and surveillance are aided with reasonably precise shore-based radar.

## 2.4 Monostatic RADAR

Monostatic RADAR systems [1] are characterized by the use of a single antenna for both transmitting and receiving signals. This antenna alternates between sending out a pulse and then receiving the echo that the target reflects. Since the antenna system only needs to be installed in one place, monostatic RADAR’s main benefit is its simplicity and convenience of use. The time it takes to transition between transmitting and receiving modes has certain intrinsic obstacles, too, and this can reduce the minimum range at which targets can be detected.

The time delay between the broadcast pulse and the received echo is measured in monostatic RADAR to calculate the target’s range. The following gives the distance $R$ to the target:

$$R = \frac{c \cdot \Delta t}{2} \tag{2.1}$$

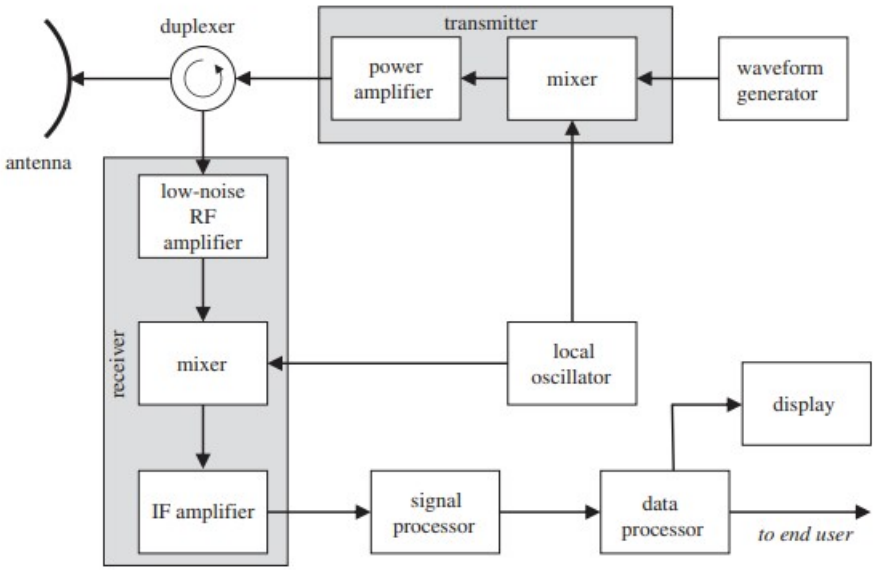



Figure 2.1: Monostatic RADAR diagram [1]

where $c$ is the speed of light and $\Delta t$ is the time delay. The factor of 2 accounts for the fact that the signal travels to the target and back.

Below you can check the description of each component shown in the Figure:

**Transmitter:** A power amplifier, a mixer, and a waveform generator make up the transmitter part. The original RADAR signal is produced by the waveform generator and combined in the mixer with a signal from the local oscillator. This signal is amplified before transmission by the power amplifier.

**Duplexer:** The duplexer is a switch that allows the same antenna to be used for both transmitting and receiving. It directs the high-power transmitted signal to the antenna and protects the sensitive receiver components from damage by the transmitted signal.

**Antenna:** The antenna emits the transmitted signal into the environment and receives the reflected echoes from targets.

**Receiver:** A mixer, an IF amplifier, and a low-noise RF amplifier make up the receiving section. The low-noise RF amplifier initially amplifies the received echo signals. The signals are then converted to an intermediate frequency (IF) by the mixer by combining them with a signal from the local oscillator. These signals are amplified even further by the IF amplifier.

**Local Oscillator:** The local oscillator generates a stable frequency that is used in both the

transmitter and receiver mixers to convert signals to and from the intermediate frequency.

**Signal Processor:** The signal processor processes the IF signals to extract useful information about the targets, such as their range and velocity.

**Data Processor:** The information from the signal processor is further processed by the data processor so that the final user may view and understand it.

**Display:** The display shows the processed data in a human-readable format, allowing the end user to interpret the RADAR information.

## 2.5 Radar Equation

An essential tool for the analysis and design of radar systems is the radar equation [8], [10]. It connects the radar range to the properties of the target, surroundings, antenna, transmitter, and receiver. This equation sheds light on radar operations and helps calculate the maximum distance at which a radar can identify a target.

A radar transmitter with power $P_t$ is considered. The power density (measured in watts per unit area) at a distance $R$ from the radar can be calculated by dividing the transmitter power by the surface area of a sphere with radius $R$ if this power is uniformly radiated in all directions using an isotropic antenna:

$$\text{Power density from isotropic antenna} = \frac{P_t}{4\pi R^2} \quad (1.3) \tag{2.2}$$

In order to concentrate the radiated power $P_t$ in a particular direction, radars usually use directive antennas. When compared to an isotropic antenna, the gain $G$ of an antenna measures the increase in power radiated towards the target. The maximum radiation intensity from the directive antenna divided by the maximum radiation intensity from an isotropic antenna with the same input power is known as the gain, or $G$. As a result, the power density of an antenna with gain $G$ at the target is:

$$\text{Power density from directive antenna} = \frac{P_t G}{4\pi R^2} \quad (1.4) \tag{2.3}$$

A portion of this incident power is intercepted by the target, which then reradiates it in different directions. The amount of incident power that is intercepted by the target and rera-

diated back towards the radar is represented by the radar cross section $\sigma$. It has the following definition:

$$\text{Power density of echo signal at radar} = \frac{P_t G^2 \sigma}{(4\pi)^2 R^4} \quad (1.5) \tag{2.4}$$

The target's size as perceived by the radar is represented by the radar cross section $\sigma$, which has units of area. A portion of the echo power is captured by the radar antenna. Given $A_e$, which represents the effective area of the receiving antenna, the power $P_r$ that the radar receives can be calculated as follows:

$$P_r = \frac{P_t G^2 \sigma A_e}{(4\pi)^2 R^4} \quad (1.6) \tag{2.5}$$

## 2.6 RADAR Waveform Types

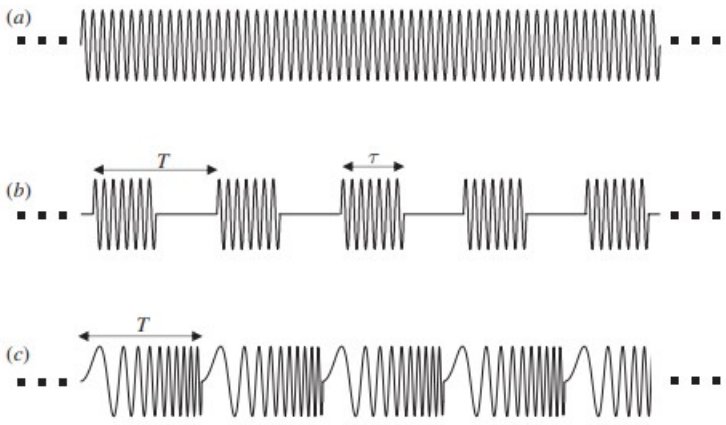



Figure 2.2: Three of the major classes of radar waveforms. (a) Continuous wave (**CW**) (b) **Pulsed** (c) Frequency-modulated continuous wave (**FMCW**)[[1]]

A radar typically transmits a waveform that can be modeled as follows:

$$\overline{s}(t) = a(t)\sin(\Omega t + \theta(t)) \tag{2.6}$$

In this equation, the term $\Omega$ represents the carrier radar frequency (RF) in radians per second. The function $a(t)$ signifies the amplitude modulation of the RF carrier. In the context of pulsed radar, $a(t)$ is usually a rectangular function that modulates the waveform, effectively turning it on and off. The term $\theta(t)$ accounts for any phase or frequency modulation of the

carrier. This phase term can be zero, a nonzero constant, or a more complex function. The overbar on $\overline{s}(t)$ indicates that the signal is on a carrier and has not yet been demodulated.

Figure 2.2 illustrates three common waveform types in pulsed radar. The first type, the simple pulse, is characterized by a constant-amplitude burst at the RF frequency. The second type, the linear frequency modulated (LFM) pulse, exhibits a frequency that increases at a constant rate during the pulse duration. LFM pulses can also exhibit decreasing frequency during the pulse. The third example is a binary phase-coded pulse, where the frequency remains constant, but the absolute phase of the waveform alternates between zero and $\pi$ radians multiple times within the pulse. In this case, $\theta(t)$ alternates between the constants zero and $\pi$ at specific intervals within the pulse.

### 2.6.1 CW Radar

Continuous Wave (CW) radar systems operate by transmitting a continuous sine wave, which may be unmodulated or modulated. These systems frequently utilize a bistatic configuration to maintain uninterrupted transmission. Unmodulated CW radars can determine the relative velocity of a target through the Doppler shift observed between the transmitted waveform and the received echoes. However, since target range is typically measured by the two-way time delay, single-frequency unmodulated CW radars are unable to measure the target range directly. By employing modulated CW waveforms, it is possible to introduce timing marks at both transmission and reception, enabling range measurement.

Modulation of CW waveforms can occur in amplitude , frequency , or phase , with frequency-modulated continuous wave (FMCW) radar being the most extensively studied in literature. While some sophisticated CW radars are employed in semi-active missile tracking systems, CW radars are predominantly utilized in cost-effective, simpler applications such as police speed radar, automotive radar, and radar altimeters.

### 2.6.2 Pulse Radar

Pulsed radar systems release one single pulse or a succession of finite-duration pulses, in contrast to continuous wave radar. When using a monostatic architecture, the receiver is turned on just before to the next pulse being transmitted, and the transmitter is turned off right after the pulse has been transmitted. The equation 2.6 is used also for pulse RADAR.

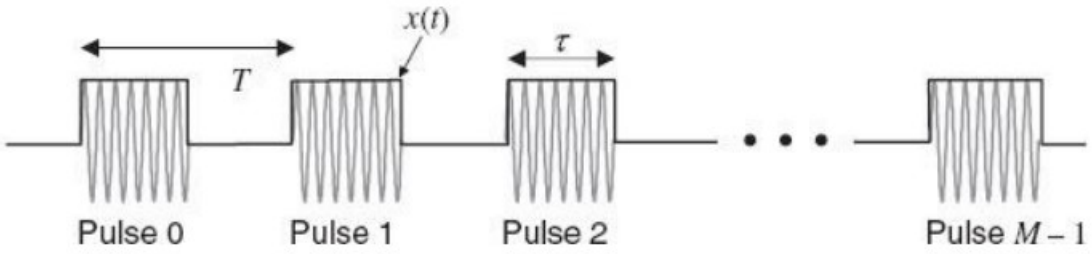



Figure 2.3: Pulse RADAR Waveform [1]

**Basic Function**

The main function of a pulse RADAR, as all types of RADAR, is to calculate the range between the Radar and the target. The main equation is shown in 2.1. However in a real scenario, there are some limitations. A pulse radar sends a single pulse in a series and then waits for its return. For simplicity, assume the system sends only one pulse. The sum of the time for transmitting and waiting for the return is called PRI (Pulse Repetition Interval), denoted as $T$ in Figure 2.3. Consider a target located within the period when the waveform is zero immediately after Pulse 0. Here, the system detects the return and calculates the range. However, for targets that are either too close or too far, if a return signal arrives while the radar is transmitting a pulse, it will be ignored because the radar, being monostatic, cannot transmit and receive simultaneously. Therefore, the radar cannot measure ranges smaller than $\frac{c \cdot \tau_{\text{p}}}{2}$, where $\tau_{\text{p}}$ is the pulse width.

Now, consider the scenario where a return signal arrives after $T =$ PRI. In such cases, the radar cannot distinguish from which pulse (whether the first, second, or subsequent) the return signal originated, leading to inaccurate measurements. Hence, the maximum unambiguous time $t_{\text{max}}$ is PRI $- \tau_{\text{p}}$, and the maximum unambiguous range is:

$$R_{\text{unamb}} = \frac{c \cdot (PRI - t)}{2} \tag{2.7}$$

## 2.7 Phased Arrays

It is commonly acknowledged that antennas, which are in charge of transforming electrical impulses into electromagnetic radiation and the opposite, are what allow radars to function. As previously mentioned, radars use their antenna to emit a signal, receive a return, and

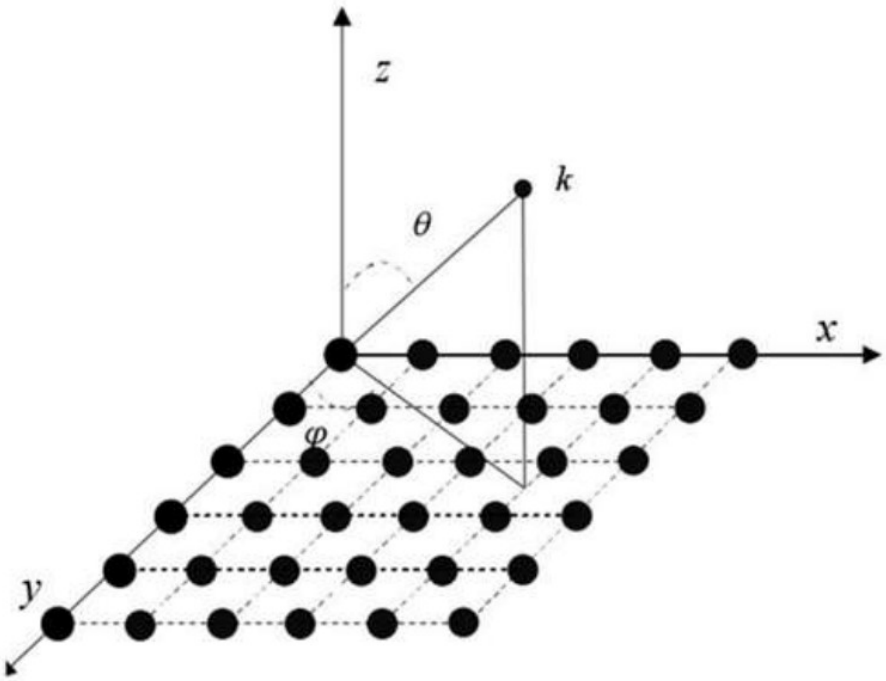



Figure 2.4: Uniform Rectangular Array[2]

process the signal as needed. Since the radiation from most antennas spreads in a 360-degree circle, they are said to be isotropic. Nevertheless, this is not totally true because antennas are equipped with devices that let them concentrate more on some angles and less on others.

Achieving the previously described concentration involves placing antennas near to one other and electronically guiding them. Phased array is the term for this method. Phased array radar antennas come in two primary varieties: Uniform Linear Array (ULA) and Uniform Rectangular Array (URA). Antennas are arranged in two different configurations: a 2D rectangular grid for the URA and a 1D linear array for the ULA. Gain, directivity, and overall performance are all enhanced in this arrangement. It also opens up additional capabilities that are essential to our study, like elevation and azimuth calculations.

### 2.7.1 URA and DOA

The most common use of URA is Direction of Arrival (DOA) in either active [2, 11, 12] or passive RADAR systems [13]. A model for RADAR signals that includes Doppler besides a ULA can be found in [13]. DoA can be used with either high resolution methods like MUSIC [14] while other prameters can be also simultaneously estimated [15]. In this subsection the calculation of azimuth and elevation angles will be described and is based on the DFT

algorithm instead of MUSIC or or other super-resolution methods. An electromagnetic plane wave signal $s(t)$ is received by an $M \times N$ Uniform Rectangular Array (URA) composed of $MN$ antenna sensor elements with equidistances $d_1$ and $d_2$ along the $x_1$- and $x_2$-axes, respectively. The signal $s_{n,m}(t)$ received by the $(n, m)$-th antenna element at time $t$ can be expressed as:

$$s_{n,m}(t) = s(t)e^{j\beta(md_1 \sin\theta\cos\phi + nd_2 \sin\theta\sin\phi)} \tag{2.8}$$

$$s_{n,m}(t) = s(t)a_{n,m}(\phi, \theta) \tag{2.9}$$

where the steering phase factor for the $(n, m)$-th antenna element is:

$$a_{n,m}(\phi, \theta) = e^{j\beta(md_1 \sin\theta\cos\phi + nd_2 \sin\theta\sin\phi)} \tag{2.10}$$

Here, $\phi$, $\theta$, and $\beta$ represent the azimuth angle, zenith angle, and wave number, respectively.

The signals received by the $N \times M$ antenna elements can be described by:

$$\begin{bmatrix} u_{0,0}(\phi,\theta,t) & \cdots & u_{0,M-1}(\phi,\theta,t) \\ u_{1,0}(\phi,\theta,t) & \cdots & u_{1,M-1}(\phi,\theta,t) \\ \vdots & \ddots & \vdots \\ u_{N-1,0}(\phi,\theta,t) & \cdots & u_{N-1,M-1}(\phi,\theta,t) \end{bmatrix}$$
$$= \begin{bmatrix} a_{0,0}(\phi,\theta) & \cdots & a_{0,M-1}(\phi,\theta) \\ a_{1,0}(\phi,\theta) & \cdots & a_{1,M-1}(\phi,\theta) \\ \vdots & \ddots & \vdots \\ a_{N-1,0}(\phi,\theta) & \cdots & a_{N-1,M-1}(\phi,\theta) \end{bmatrix} s(t) + \begin{bmatrix} \nu_{0,0}(t) & \cdots & \nu_{0,M-1}(t) \\ \nu_{1,0}(t) & \cdots & \nu_{1,M-1}(t) \\ \vdots & \ddots & \vdots \\ \nu_{N-1,0}(t) & \cdots & \nu_{N-1,M-1}(t) \end{bmatrix} \tag{2.11}$$

where $\nu_{n,m}(t)$ is the additive white Gaussian noise (AWGN) for the signal $s_{n,m}(t)$ incident on the $(n, m)$-th antenna element at time $t$.

The phase angles of the signals (noise-free) received by every antenna element in the URA can be expressed as the following N×M matrix, given equations 2.10 and 2.11:

$$B = [b_{n,m}(\phi, \theta, t)] =$$

$$\begin{bmatrix} \beta\sin\theta(0+0)+\phi(t) & \beta\sin\theta(d_1\cos\phi+0)+\phi(t) & \cdots & \beta\sin\theta((M-1)d_1\cos\phi+0)+\phi(t) \\ \beta\sin\theta(0+d_2\sin\phi)+\phi(t) & \beta\sin\theta(d_1\cos\phi+d_2\sin\phi)+\phi(t) & \cdots & \beta\sin\theta((M-1)d_1\cos\phi+d_2\sin\phi)+\phi(t) \\ \vdots & \vdots & \ddots & \vdots \\ \beta\sin\theta(0+(N-1)d_2\sin\phi)+\phi(t) & \beta\sin\theta(d_1\cos\phi+(N-1)d_2\sin\phi)+\phi(t) & \cdots & \beta\sin\theta((M-1)d_1\cos\phi+(N-1)d_2\sin\phi)+\phi(t) \end{bmatrix} \tag{2.12}$$

where $\phi(t)$ is the phase angle of the signal $u_{0,0}(t)$ received by the antenna element at the origin $(0,0)$ of the URA. The phase differences between neighboring entries of the above matrix $B$ along its rows and columns are:

$$x_m \approx \beta d_1 \sin\theta\cos\phi \quad \text{for } m = 1,\ldots,M-1 \tag{2.13}$$

$$y_n \approx \beta d_2 \sin\theta\sin\phi \quad \text{for } n = 1,\ldots,N-1 \tag{2.14}$$

We can take their mean values:

$$m_x = \frac{1}{M-1}\sum_{m=1}^{M-1} x_m \quad \text{(mean value of } x_m\text{)} \tag{2.15}$$

$$m_y = \frac{1}{N-1}\sum_{n=1}^{N-1} y_n \quad \text{(mean value of } y_n\text{)} \tag{2.16}$$

And use them to get the azimuth angle estimate $\hat{\phi}$ and the elevation angle estimate $\hat{\theta}$ as:

$$\hat{\phi} = \tan^{-1}\left(\frac{m_y}{m_x}\right) \tag{2.17}$$

$$\hat{\theta} = \sin^{-1}\left(\frac{m_y}{\beta d_2 \sin\hat{\phi}}\right) \tag{2.18}$$

# Chapter 3

# Introduction to Deep Learning

## 3.1 Definition

Deep learning (DL) is a subset of machine learning (ML) techniques that operates on the principle of extracting features from raw data by utilizing multiple layers to identify various relevant aspects of the input data[16]. Key deep learning methodologies include convolutional networks, recurrent neural networks, and deep neural networks. Deep learning predominantly employs artificial neural networks, particularly recurrent neural networks, which are used in this study's experiment.

In the past, machine learning's limited capacity to handle unprocessed data effectively limited its application. Because deep learning can handle massive amounts of data, it has been able to overcome this constraint and establish itself as a useful and successful machine learning technology. Hardware advancements in computers have also expedited the development of deep learning.

To convert raw data into a format that computer subsystems could recognize and classify, significant experience with feature extraction was previously required. This necessity has been lessened by deep learning, whose layered design automates the feature extraction procedure.

## 3.2 Deep Learning for RADAR

In the recent years there has been a need for integration of Deep Neural Networks (DNNs) for RADAR applications, as they solve some bottle necks in the signal processing stage. As

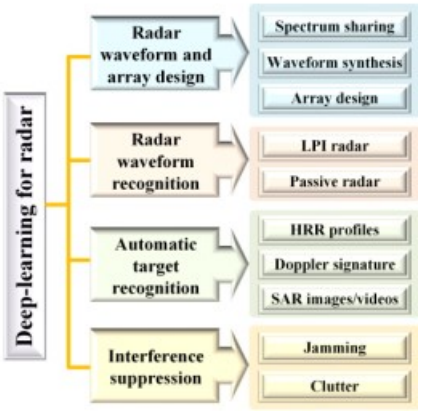



Figure 3.1: Deep Learning for Radar [3]

described in [3], the two main types of DNN that are used for RADAR system are Convolutional Neural Networks (CNN) and Long Short-Term Memory (LSTM) Networks. CNNs are particularly useful for finding patterns in images to recognize objects, classes, and categories, while LSTM systems are predominantly used to learn, process, and classify sequential data because these networks can learn long-term dependencies between time steps of data. Common LSTM applications include sentiment analysis, language modeling, speech recognition, and video analysis. Given that LSTM is capable of learning from time series and sequential data, it is an ideal choice for this study.

## 3.3 Recurrent Neural Networks

A Deep Neural Network (DNN) usually consists of multiple hidden layers that transfer information from one neuron to the next in the subsequent layer. On the other hand, a Recurrent Neural Network (RNN) [4]has the potential for the outputs of neurons to be fed back as inputs to neurons in preceding layers. Because of this feedback loop, RNNs are especially well-suited for tasks requiring time series, natural language processing, and other sequential data. They can also be used to simulate temporal sequences and relationships within the data.

### 3.3.1 Training

The most common methods to train recurrent neural networks are Backpropagation Through Time (BPTT) and Real-Time Recurrent Learning (RTRL) .The primary distinction between BPTT and RTRL lies in the method of computing weight updates. The initial formulation

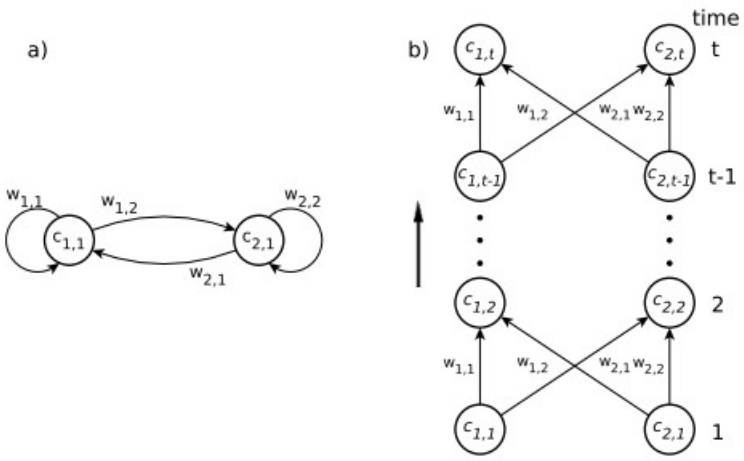



Figure 3.2: Figure a) shows a simple fully recurrent neural network with a two neuron layer. The same network unfolded over time with a separate layer for each time step is shown in Figure b). The latter representation is a feed-forward neural network [4]

of LSTM-RNNs utilized a blend of BPTT and RTRL, hence we will briefly discuss both learning algorithms

**BPTT**

The Back-propagation Through Time (BPTT) algorithm leverages the fact that, over a finite period, a feed-forward neural network (FFNN) can exhibit identical behavior to a recurrent neural network (RNN). To achieve this equivalence, the RNN must be unfolded in time. Figure 3.2a illustrates a simple, fully recurrent neural network with a single two-neuron layer, while Figure 3.2b depicts the corresponding FFNN, requiring a separate layer for each time step, all sharing the same weights. When the weights are identical to those of the RNN, both networks demonstrate the same behavior.

The unfolded network can be trained using the back-propagation technique. After a training sequence ends, the network is unfolded in time, and using a chosen error measure, the error is computed for the output units with the current goal values. After that, the network propagates the error backward, and weight updates are calculated for each time step. The sum of the deltas over all time steps is used to update the weights in the recurrent version of the network.

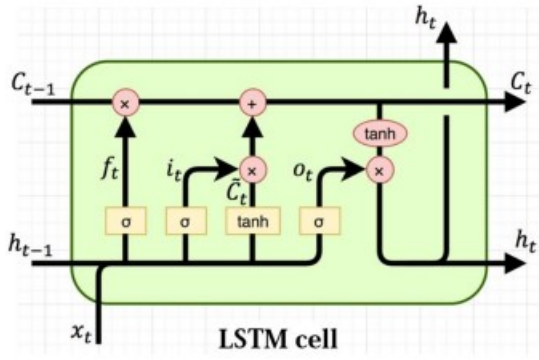
**RTRL**

The Real-Time Recurrent Learning (RTRL) algorithm doesn't require error propagation, since it gathers all the information needed for gradient computation while the input stream is being presented to the network. This removes the requirement for a specific training window. Nevertheless, it necessitates the storing of non-local data, or the output's sensitivity, and is computationally demanding. In additions, the size of the network determines the amount of memory needed, not the size of the input.

### 3.3.2 Vanishing Gradient Error

Over five to ten time steps, a conventional Recurrent Neural Network (RNN) finds it difficult to span. The back-propagated error signals, which have a tendency to either increase or decrease with each time step, are the cause of this limitation. These faults usually either explode or disappear over a large number of time steps. The learning process becomes unstable when the error signals explode because they cause the weights in the network to oscillate. On the other hand, when the error signals disappear, the network is unable to generate significant changes, which causes the learning process to become unfeasible or fail completely. The usefulness of ordinary RNNs in handling lengthy sequences is limited by this underlying problem.

## 3.4 LSTM




$$i_t = \sigma\left(x_t U^i + h_{t-1} W^i\right)$$
$$f_t = \sigma\left(x_t U^f + h_{t-1} W^f\right)$$
$$o_t = \sigma\left(x_t U^o + h_{t-1} W^o\right)$$
$$\tilde{C}_t = \tanh\left(x_t U^g + h_{t-1} W^g\right)$$
$$C_t = \sigma\left(f_t * C_{t-1} + i_t * \tilde{C}_t\right)$$
$$h_t = \tanh(C_t) * o_t$$

Figure 3.3: Structure of the LSTM cell and equations that describe the gates of an LSTM cell [5]

Long Short-Term Memory (LSTM) is a type of recurrent neural network (RNN) designed to address the vanishing gradient problem. It achieves this through several key mechanisms:

## Constant Error Carousel (CEC)

The Constant Error Carousel (CEC) is central to LSTM's ability to maintain long-term dependencies. It ensures that the error signal persists over time, enabling the network to remember information for extended durations. The CEC is implemented through self-connected units that allow error flow to remain constant, ensuring robust memory retention. In the context of a single unit $u$ with a self-connection, the local error backpropagation at time-step $\tau$ is given by

$$\vartheta_u(\tau) = f_0'(z_u(\tau))W[u,u]\vartheta_u(\tau+1).$$

To maintain a constant error flow through $u$, Equation 22 indicates the necessity for:

$$f_0'(z_u(\tau))W[u,u] = 1.0,$$

and through integration (Equation 23):

$$f_u(z_u(\tau)) = z_u(\tau)W[u,u].$$

From this, it follows that $f_u$ must be linear, ensuring $u$'s activation remains constant over time:

$$y_u(\tau+1) = f_u(z_u(\tau+1)) = f_u(y_u(\tau)W[u,u]) = y_u(\tau).$$

This constancy is achieved by using the identity function $f_u = \mathrm{id}$, with $W[u,u] = 1.0$. This mechanism, termed the Constant Error Carousel, underpins LSTM's capability for long-term memory storage.

## Memory Blocks

While the CEC maintains constant error flow in isolation, within a neural network, LSTM extends this concept by integrating input and output gates linked to the network and other memory cells. Figure 11 illustrates the standard architecture of an LSTM memory block.

The input gates, modeled as sigmoid threshold units with activation range [0, 1], regulate signals entering the memory cell by scaling them appropriately. When closed, these gates minimize activation, thereby protecting stored contents from irrelevant inputs. The activation of the CEC by the input gate defines the cell state.

Output gates control access to the memory cell's contents, safeguarding other memory cells from disruptions originating from $u$. This multiplicative gating mechanism allows the LSTM unit to selectively permit or deny the propagation of the constant error flow through the CEC, ensuring robust memory management in complex neural networks.

In an LSTM cell, additional gates—specifically the input, forget, and output gates—are employed to determine which signals propagate to the next node. The recurrent connection $W$ links the previous hidden layer to the current one, while $U$ represents the weight matrix connecting inputs to the hidden layer. $C_e$ denotes the candidate hidden state, calculated from the current input and the preceding hidden state. $C$ signifies the internal memory of the unit, derived from a blend of the prior memory modulated by the forget gate and the freshly computed hidden state modulated by the input gate. The equations governing the behavior of all gates within the LSTM cell are depicted in Figure 3.3.

### 3.4.1 Training

In a neural network, learning entails creating a mapping between inputs and matching outputs[5]. During training, (input, output) pairs are provided to the network. The network modifies its weights throughout training in order to generate precise outputs for the supplied inputs. The network should be able to accurately predict outputs for inputs that were not seen during training in order to demonstrate good generalization.

When the network can accurately predict each training input's proper output, training is considered to have ended. The cost or loss function is a metric used to quantify how near the desired result is to the projected output. This function measures the difference between expected and desired values, which helps to steer the network. Often employed as a cost function, the mean squared error is defined as follows:

$$\text{cost} = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y_i})^2,$$

where $n$ is the number of data points, $Y_i$ is the target value, and $\hat{Y_i}$ is the predicted value.

Weights are iteratively changed by backpropagation during training in order to minimize the loss function. Stochastic gradient descent optimization is made possible through backpropagation, which calculates the loss function's gradient with respect to the weights.

## 3.5 Autoencoder

An artificial neural network that uses backpropagation to generate an output vector that resembles the input vector is called an autoencoder. The incoming data is first compressed into a lower-dimensional space, and from this representation, the original data is reconstructed. The autoencoder learns the non-linear relationships in the data by using many layers and non-linear activation functions. Figure 3.4 depicts a basic diagram of the autoencoder model architecture. There are two primary stages to it: the encoder and the decoder. As per Equation (3.1), the main goal of the encoder phase is to decrease the dimensionality of the input data $\mathbf{X}$.

$$\mathbf{Z} = \sigma(\mathbf{W}\mathbf{X} + \mathbf{b}) \tag{3.1}$$

Here, $\mathbf{Z}$ represents the latent dimension, $\sigma$ denotes the activation function, $\mathbf{W}$ is the weight matrix, and $\mathbf{b}$ is the bias vector. Similarly, the decoding phase is governed by Equation (3.2) to reconstruct the output data that resembles the original input data but with potentially different biases, weights, and activation functions.

$$\mathbf{X}' = \sigma'(\mathbf{W}'\mathbf{Z} + \mathbf{b}') \tag{3.2}$$

The main objective of the autoencoder is to minimize the reconstruction error between the output vector and the original input space. One can calculate the reconstruction error by utilizing the sum of squared errors (SSE) or a cross-entropy function. To determine the reconstruction error in this work, we employ SSE which can be calculated as shown in Equation (3.3).

$$\text{SSE} = \sum_{i=1}^{n} (\mathbf{X}'_i - \mathbf{X}_i)^2 \tag{3.3}$$

The decoder part regenerates the initial data based on the encoder output. To achieve dimensional reduction and generate a compressed feature vector from the input data, the

code layer is used at the center of the autoencoder structure. This code layer can be employed for classification tasks or combined with another stacked autoencoder.

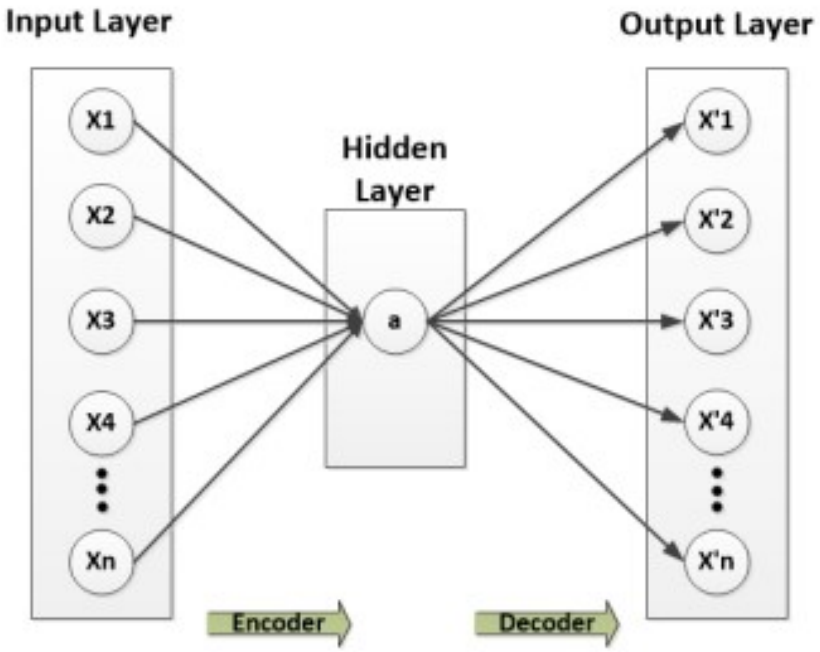



**Figure 2: An example of a single autoencoder.**

Figure 3.4: Illustration of the autoencoder model architecture [6]

## 3.6 LSTM Autoencoder

LSTM-Autoencoder combines the strengths of Long Short-Term Memory (LSTM) neural networks and autoencoders [17] by integrating LSTM networks within the encoder and decoder architectures of the autoencoder framework. This hybrid approach enables the model to effectively process and learn from sequential data with complex dependencies.

Sequences of high-dimensional input data are first sent into the LSTM-Autoencoder's encoder component. Through a succession of LSTM layers, each sequence—typically representing a time-series dataset—is converted into a fixed-size vector representation. Long-term relationships across several data points within each sequence can be captured by the LSTM because to its recurrent structure. Through incremental processing of the input data, the encoder gradually reduces the high-dimensional input vector representation to a lower-dimensional latent space. By retaining the important aspects of the original input data and

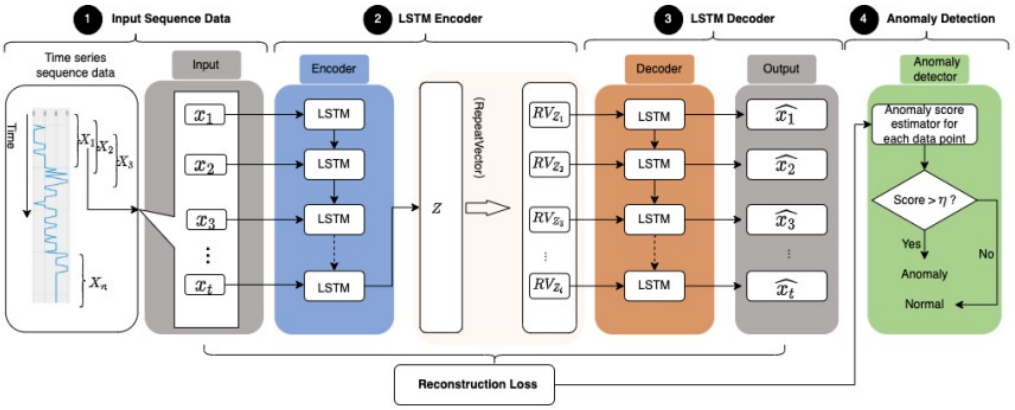



Figure 3.5: LSTM Autoencoder [7]

eliminating noise and extraneous details, this latent space functions as a compressed representation of the original data. This reduction in dimensionality aids in both effective data representation and the reconstruction procedure that follows.

The LSTM decoder uses the reduced representation kept in the latent space after the encoding stage to recreate the original input sequence. When compared to the encoder, the decoder works in reverse, taking the fixed-size vector from the latent space and expanding it back into the original dimensions' sequence. The difference between the original input and the rebuilt output is known as the error rate, which is used by the LSTM decoder to define a threshold for anomaly identification throughout this reconstruction process. When the reconstruction error is greater than a predetermined threshold, anomalies in the input data are detected. By allowing the model to discern between typical patterns and anomalous deviations in the data, this threshold plays a crucial role in anomaly detection systems.

Applications requiring anomaly detection in time-series data, such as industrial process monitoring systems, network traffic analysis, and healthcare monitoring, benefit greatly from the use of the LSTM-Autoencoder architecture. Using autoencoders for unsupervised learning and dimensionality reduction and LSTM networks for sequential data processing, the model may identify minute deviations or anomalies that might point to possible problems or irregularities in the systems it is monitoring. This hybrid technique improves the model's robustness in real-world applications where data may display non-linear and dynamic behaviors over time, in addition to improving the model's capacity to learn intricate patterns and connections in data.

All things considered, the LSTM-Autoencoder is a potent weapon in the toolbox of ma-

chine learning methods, providing a strong foundation for anomaly identification and sequential data processing. The model successfully tackles the difficulties of learning from high-dimensional, time-dependent data while offering interpretable and useful insights into the behavior of complex systems by using LSTM networks into the autoencoder architecture.

# Chapter 4

# Case Study and Implementation

## 4.1 Case Study

The objective of this research is to develop a Deep Neural Network (DNN) that can reliably identify whether one or more jammers are present in a series of radar signals. The selected DNN model—an LSTM Autoencoder—is very well-suited for time series data anomaly detection. Since the existence of a jammer is unusual for routine radar operations, it is regarded as an anomaly in the context of this study. Target azimuth and elevation are retrieved from radar data during preprocessing. An image dataset is then produced by applying a 2D Discrete Fourier Transform (DFT). The input used to train the LSTM Autoencoder DNN is this dataset. The study is aimed at assessing the classification accuracy of radar signals incorporating jammers by the LSTM Autoencoder. The effectiveness of the DNN in separating out radar signals influenced by jamming anomalies from regular signals is evaluated by analyzing the results.

## 4.2 Implementation Overview

The fundamental models and systems are explored that are crucial to our research in this chapter. First, we present the RADAR system, which is the foundation for acquiring images in our collection. Then, we carefully examine the process of creating datasets, emphasizing approaches and factors. Additionally, we clarify the subtleties of our neural network architecture, which is intended to make good use of the dataset. Lastly, we give a thorough explanation of the training schedule and assessment procedures used to confirm the functionality

of our model. These elements work together to provide the framework for our study, which aims to produce solid insights and advances in our profession.

## 4.3 RADAR System

A specialized system called the Uniform Rectangular Array (URA) Pulse Radar System is needed to generate the dataset for the Deep Neural Network (DNN). A phased array of 256 antennas, placed in a rectangular grid with dimensions of $16 \times 16$, makes up this system. Since $c$ is the speed of light and $f_c$ is the carrier frequency, which is fixed at 1 GHz, the distance $d$ between neighboring antennas is specified as $d = \frac{0.5 \cdot c}{f_c}$.

Pulses with a start delay of three seconds and a pulse width of four seconds are released by the radar system. The signals that are received by each of the 256 antennas are processed so that they can be seen as images. The neural network receives these images as inputs for additional analysis.

By ensuring that the system records a wide range of signals, the array architecture offers a solid dataset for neural network training. By using sophisticated signal processing techniques to convert these signals into pictures, high-quality, useful input is fed into the neural network. The creation of a comprehensive dataset is an essential step in the creation and training of a DNN model.

## 4.4 Dataset

Using the Python programming language, the dataset utilized in this research was created to simulate the RADAR responses of a legitimate target moving at random in the RADAR system's area. 2D Discrete Fourier Transform (DFT) representations of the signals received by the RADAR are the images used for model evaluation and training. These 2D DFT images have dimensions along the azimuth (x-axis) and elevation (y-axis). The azimuth dimension relates to the horizontal angle (left-right), whereas the elevation dimension shows the vertical angle (up-down) with respect to the RADAR. There are degrees involved in both dimensions. Greater yellow coloring denotes larger signal amplitudes in these visual representations, while greater blue coloring denotes lower signal amplitudes. Because of the color coding, it is easy to understand how strong the signals are inside the pictures. Figure 4.1

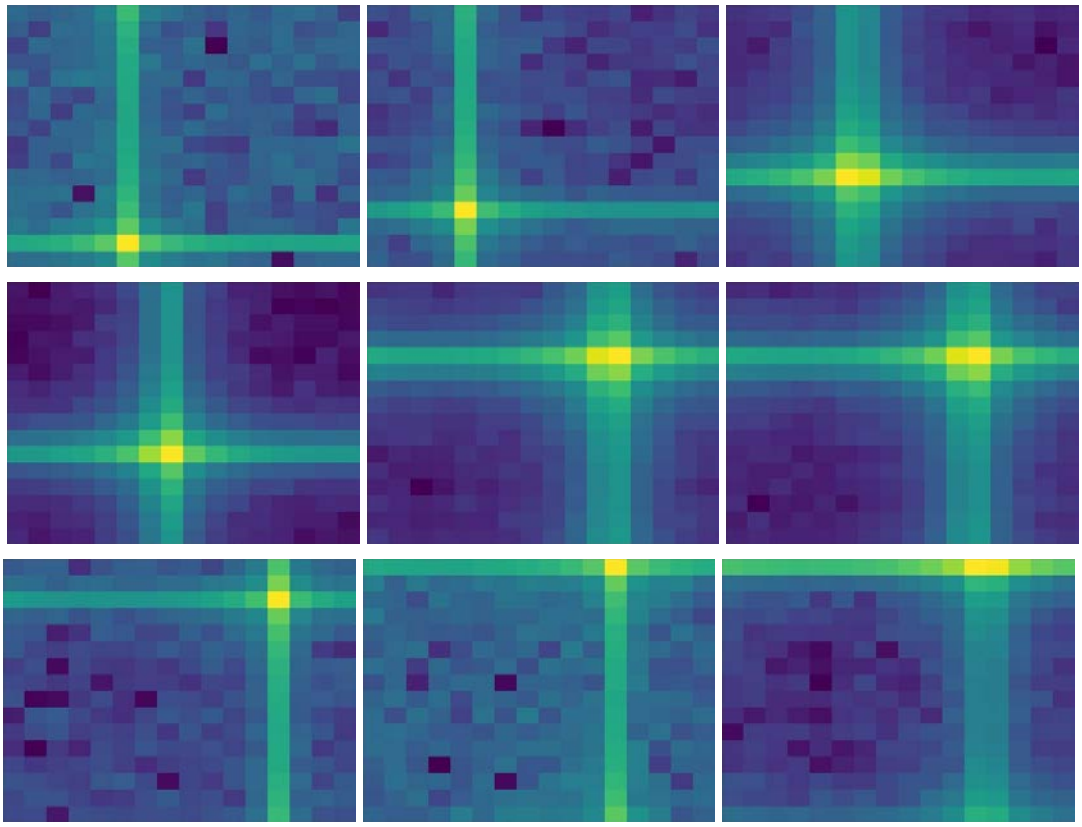


Figure 4.1: Dataset snippet without jammer, **SNR:** 20dB

provides an example, illustrating a target moving from the lower left to the upper right of the image. This movement is depicted by the changing signal amplitudes and their corresponding colorations within the 2D DFT representation. Moreover, it is composed of sixty 600 x 480 pixel images that replicate sixty seconds of RADAR answers. Figure 4.1 offers a sneak peek at these pictures. Every time the model needed to be evaluated or retrained, a fresh random dataset was created to ensure that overfitting would not occur and that the model could detect jammers effectively.

**Assumptions** that have taken place when creating the dataset:

- **Limitations in Azimuth and Elevation Representation:** The radar system's 256 antennas, which are grouped in a $16 \times 16$ uniform rectangular array, yield a comparatively low signal resolution. As a result, there are azimuth and elevation values that the Discrete Fourier Transform (DFT) cannot adequately represent. To be more precise, data outside of the range of -70 degrees to 70 degrees in azimuth and elevation are not well represented. This constraint results in a decreased capacity to distinguish minute details at extreme angles because of the finite number of antennas' inadequate angular

resolution. The overall accuracy and fidelity of the radar signal processing may be impacted as a result of the underrepresentation of signal components that correspond to these extreme values.

- **Jammer Location** Jammer Location is fixed in this study because in real life a jammer is usually jamming from a steady location and not moving. Also it hits for a specific amount of time, for example 2-3 seconds

The dataset depicted in Figure 4.1 is utilized for training purposes, enabling the model to learn the characteristics of normal activity. To evaluate the model's performance, it is essential to create a distinct dataset incorporating jammers. This is achieved by introducing interference signals. In Figure 4.2, jammers are introduced for two seconds in images 2 and 3. The presence of the jammer is identifiable by the yellow square in the top right corner of these images. This dataset is crucial for assessing the model's ability to detect anomalies and accurately differentiate between normal and jammed signals.

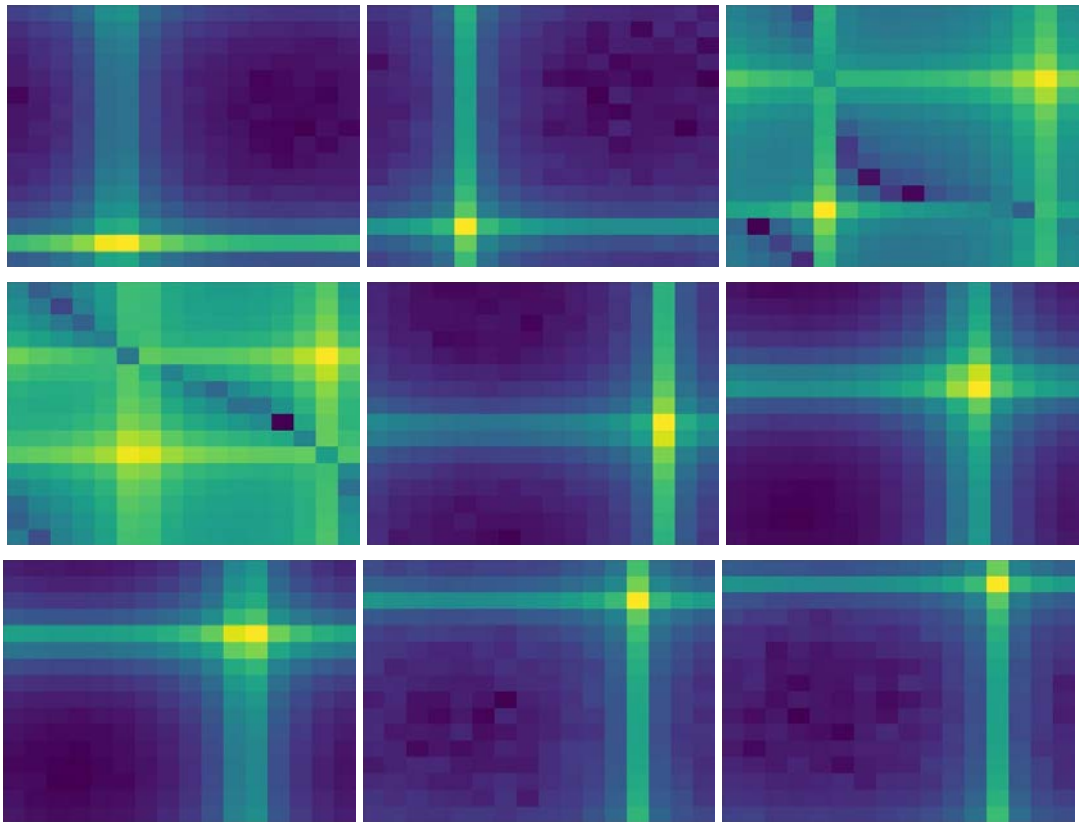


Figure 4.2: Dataset snippet with jammer, **SNR:** 30dB

More specifically, the dataset for training purposes it is constructed with these steps:

- Initialize the signal **x** with zeros and put ones starting on pulse_start and ending on pulse_start + pulse_width
- Get azimuth and elevation angle and then calcuate the steering vector:

$$\theta = \text{azimuth\_angle in rad}, \quad \phi = \text{elevation\_angle in rad}$$

$$\mathbf{a}_{\text{src1}} = \text{Beamforming on the initialized signal using the Uniform Rectangular Array antenna positions and the angles } \theta \text{ and } \phi. \tag{4.1}$$

  Steering vector needs to be on exponential form:

$$\mathbf{a}_{\text{src}} = \exp\left(-j \cdot \pi \cdot fc \cdot \mathbf{a}_{\text{src1}}\right)$$

- Multiply transposed $\mathbf{a}_{\text{src}}$ with transmit signal $\mathbf{x}$:

$$\text{signal} = \mathbf{a}_{\text{src}}^{T} \cdot \mathbf{x}$$

- Make the received signal more realistic by adding noise with power equal to signal2

$$\text{y} = \text{signal} + \text{noise}$$

- Calculate covariance matrix of signal y:

$$C_{ij} = \frac{1}{N-1} \sum_{k=1}^{N} (y_{\text{2D},k,i} - \bar{y}_i)(y_{\text{2D},k,j} - \bar{y}_j)$$

- Extract first row of C in another array and then reshape it into a 2D array with dimensions of the URA (16x16 in this case)

$$\text{acf} = \mathbf{C}_{(0,:)}$$

$$\text{acf\_2D} = \begin{bmatrix} C_{00} & C_{01} & C_{02} & \cdots & C_{0,15} \\ C_{16} & C_{17} & C_{18} & \cdots & C_{16,31} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ C_{240} & C_{241} & C_{242} & \cdots & C_{240,255} \end{bmatrix}$$

  where acf_2D is a 16x16 matrix reshaped from acf, with each element $C_{ij}$ corresponding to the elements of acf appropriately arranged.

- Calculate the DFT of autocorrelation

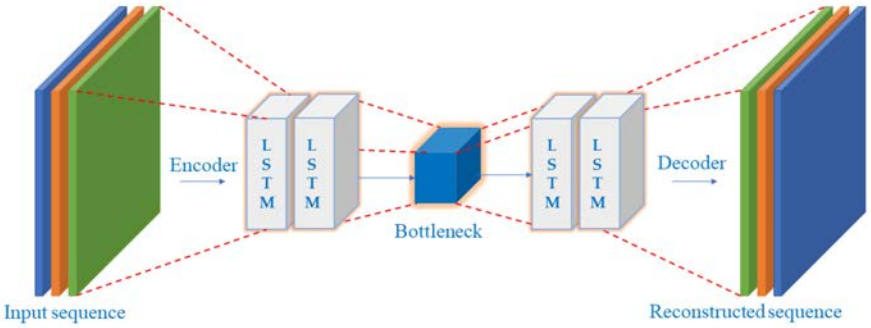



Figure 4.3: LSTM Autoencoder

## 4.5 NN Architecture

Different NN approaches have been used for classifying anomalous events. For example the authors in [18] detected anomalies in the application of wireless modulated signal classification with convolutional NNs from spectrogram images. In this work we focus on the time domain behavior of the jammer, thus the use of LSTMs. The neural network (NN) model used in this work is an LSTM Autoencoder, which is well-known for its effectiveness in anomaly identification. It is particularly good at spotting anomalies, such as jammers, in a regular image series. The input data is compressed into a condensed representation by the LSTM Autoencoder, which then reconstructs it to its original form. This procedure uses temporal dependencies to improve managing time series data accuracy. The model's ability to capture complex patterns across time enables it to detect deviations from the norm, which is important for tasks involving anomaly identification.

Enabling bidirectionality allows the LSTM to integrate data from both directions, resulting in a more thorough learning process and improved temporal pattern representation. This is especially useful for activities like anomaly detection, where knowing the background from several different perspectives can help spot abnormalities more accurately.

In order to successfully detect anomalies like jammers, the LSTM Autoencoder's higher layer depth enables the model to incorporate longer-term dependencies within sequential data. This level of detail helps the model identify irregularities from regular behavior by allowing it to pick up complex patterns and fluctuations over long periods of time. The LSTM Autoencoder's enormous 23,168,000 total number of model parameters highlights its complexity and potential. By allowing the model to accurately capture and comprehend the complex dynamics found in time-series data, this complexity plays a crucial role in attaining robust anomaly

identification.

| Layer | Input Image Size | Output Image Size |
|---|---|---|
| 1 (Encoder) | $128 \times 128$ | $64 \times 64$ |
| 2 (Encoder) | $64 \times 64$ | $32 \times 32$ |
| 3 (Encoder) | $32 \times 32$ | $16 \times 16$ |
| 4 (Decoder) | $16 \times 16$ | $32 \times 32$ |
| 5 (Decoder) | $32 \times 32$ | $64 \times 64$ |
| 6 (Decoder) | $64 \times 64$ | $128 \times 128$ |

Table 4.1: LSTM Autoencoder Layers and Image Sizes

As the autoencoder is trained, it acquires the ability to reconstruct a certain kind of image, usually one with a single, obvious target. But since the autoencoder hasn't seen anomalies like jammers during training, it is unable to replicate them accurately when they occur in the input data. The loss value for that specific instance increases significantly as a result of this failure. As a result, the loss value may be used as a signal to identify a jammer. What degree of spike in the loss values is suggestive of the presence of a jammer is a relevant question that emerges.

## 4.6 Training

The training process of the model is efficient and relatively quick due to its substantial parameter count of 23,168,000. The model can quickly get closer to being ready for jammer detection because each trip over the dataset takes around one to two minutes. The training dynamics are shown in Figure 4.4, where the mean loss is indicated by the yellow line, which is provided for illustrative purposes, and the average loss is represented by the red line.

During the initial stages of training, specifically within the first twenty iterations, the loss value exhibits a notable decrease from approximately 0.14 to around 0.05. This rapid decline signifies that the model undergoes significant learning progress early on. Following this initial drop, the loss value shows fluctuations, albeit maintaining an overall downward trajectory. Between iterations thirty and sixty, the loss stabilizes around 0.04. The model stabilizes with an average loss value of roughly 0.015 (second figure) while training continues, usually over

the course of three more cycles. This constancy in loss suggests that the model is ready to detect anomalies such as jammers and has successfully trained to reconstruct input data.

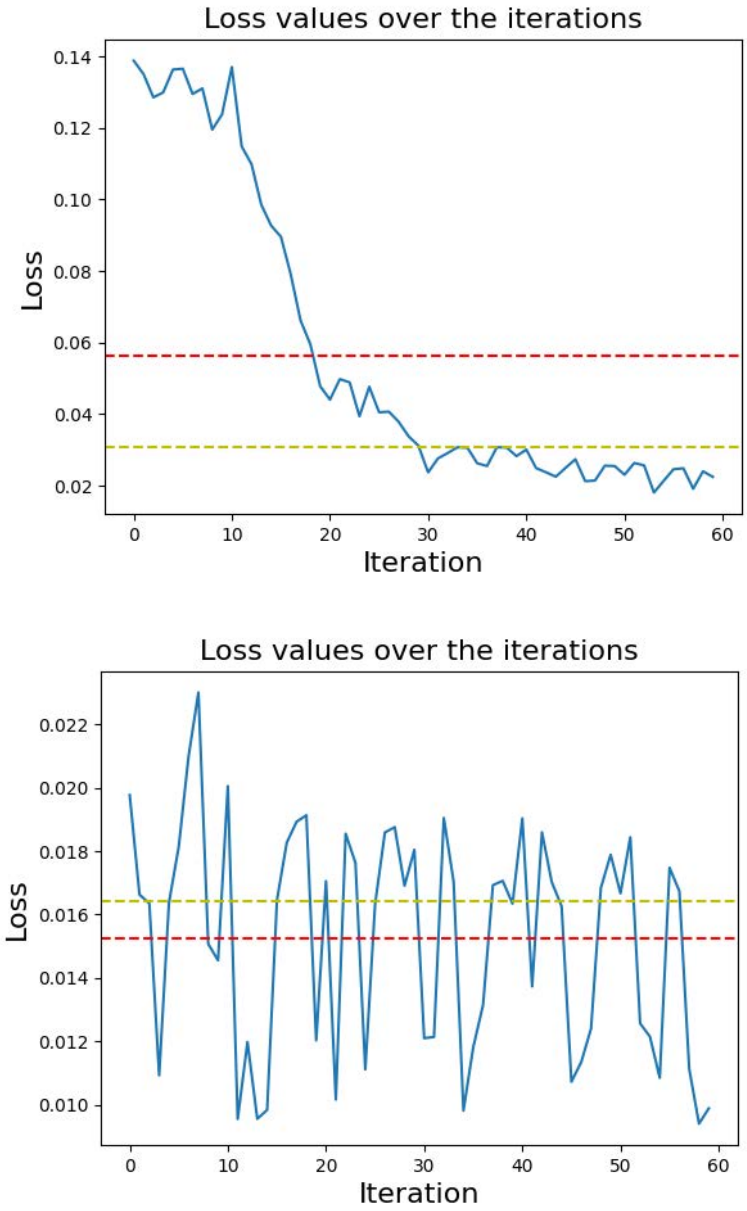



Figure 4.4: Loss vs iteration, **SNR:** 30dB

## 4.7 Evaluation

The key component of evaluating the model is determining how well it can identify jammers. During the assessment process, spikes in the loss values correspond to the times the model detects an abnormality, namely a jammer, when it is evaluated on datasets containing jammers. These spikes correspond to three jammers that have been discovered, as seen in Figure 4.5. With just a count of the loss value spikes, this graphic depiction makes it easy to categorize the quantity and location of jammers.

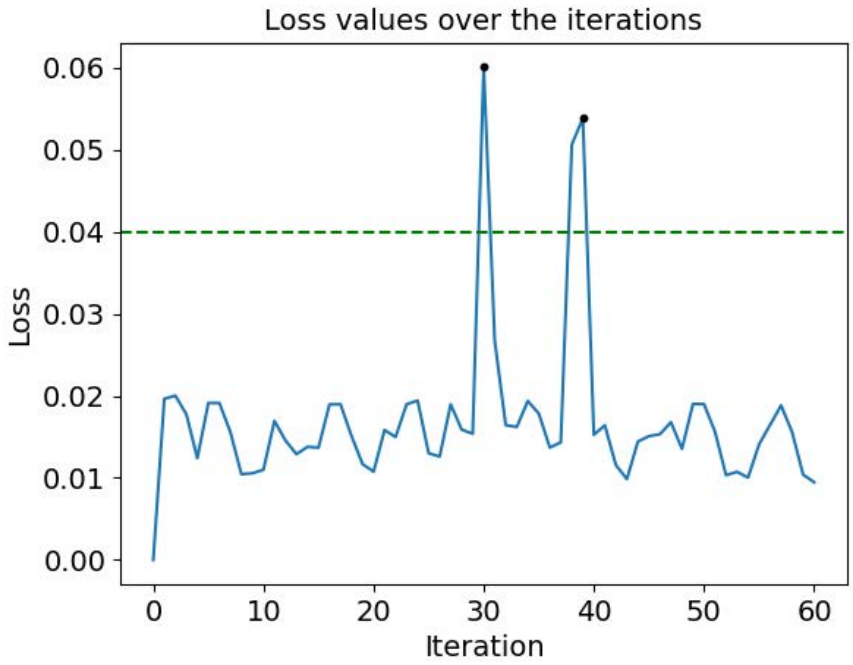



Figure 4.5: Loss vs iteration, **SNR:** 30dB

In detail, Figure 4.5 visualizes loss values during the training or evaluation of the LSTM autoencoder. The y-axis represents the loss value, reflecting the discrepancy between predicted and actual data. The blue line fluctuates across iterations, indicating the model's evaluation process. Notably, sharp spikes occur in the line—these are the detected anomalies, specifically jammers. Each spike signifies an abnormal interference event. Above the entire curve runs a dashed green threshold line, set just above 0.2 on the loss axis. Any spike surpassing this threshold is considered a significant jammer. This visualization aids in assessing model performance and quickly identifying potential jamming events. Last but not least, the two black circles in the spikes above the threshold line indicate the points that the model classifies as jammers.

Finding spikes in the loss values is important, but it is challenging due to noise and data volatility. Therefore, establishing a threshold is necessary for accurately identifying jammers. A perfect threshold has been meticulously built by careful model evaluation and accurate training. This threshold has been carefully measured to ensure that the output of the model closely resembles real-world instances of jammer activity and is set in a way that maintains the model's capacity to detect jammers while reducing the possibility of false positives.

In order to establish the threshold for anomaly detection, a graphical analysis was done. The findings revealed that several threshold multipliers evaluated against SNRs had differing accuracy. Details are demonstrated on Figure 4.6 Notably, the maximum average accuracy was obtained with a threshold multiplier of 1.7, which is shown in Figure 4.7. This provided guidance in developing the threshold computation, guaranteeing maximum sensitivity in detecting anomalies. Therefore, the threshold can be calculated by this formula:

$$\text{threshold} = \mu_{\text{error}} + 1.7 \times \sigma_{\text{error}} \tag{4.2}$$

where $\mu_{\text{error}}$ represents the mean error and $\sigma_{\text{error}}$ denotes the standard deviation of the error values.

The model's ability to estimate the number of jammers is evaluated through a classification approach. This classifier assesses how well the observed number of jammers matches the predictions made by the model. This assessment sheds light on how successfully the model detects the existence of jammers. These evaluations are essential for determining how well the model performs in real-world operating situations where efficiency and security depend heavily on the ability to identify abnormalities like jammers. The following criterion is used to measure accuracy:

$$\textbf{accuracy} = \begin{cases} \frac{\text{spikes}}{\text{total jammers}} \times 100 & \text{if number of spikes found} \leq \text{total jammers in the image} \\ 0 & \text{otherwise} \end{cases}$$

where:

- **spikes**: The number of spikes (predicted jammers) found by the model.
- **total jammers**: The actual number of jammers in the image, known from the data.
- **accuracy**: The percentage of correctly predicted jammers.

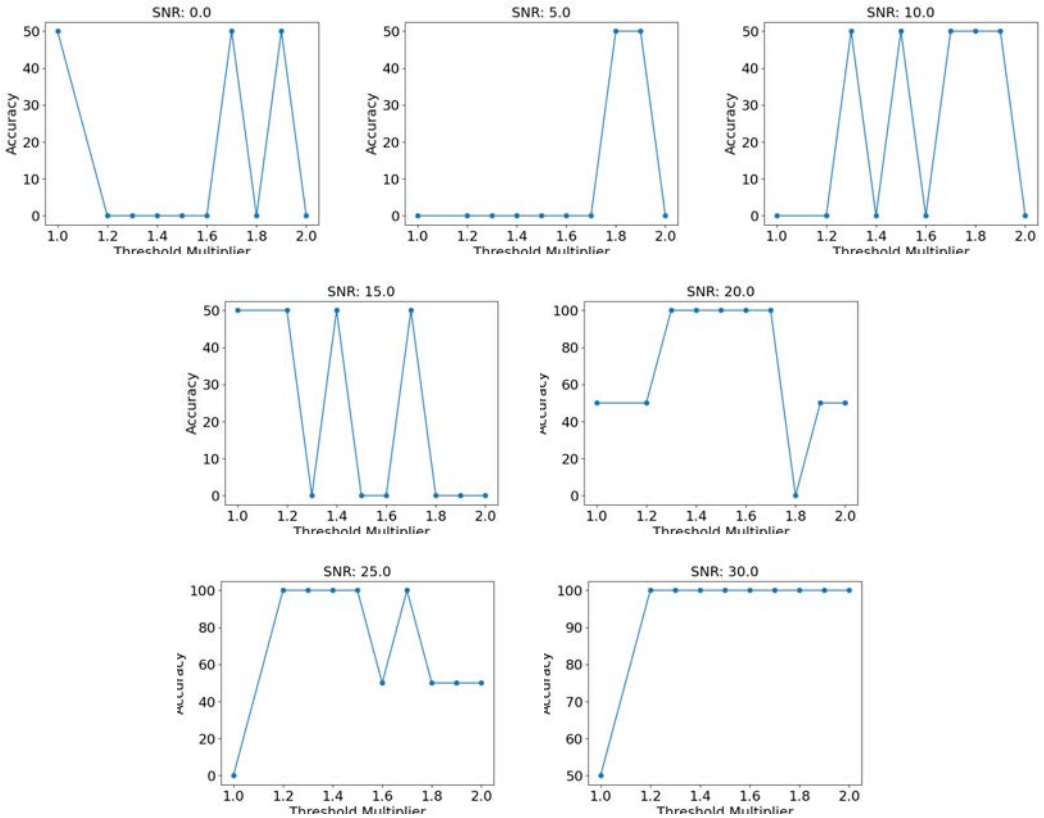



Figure 4.6: Accuracy vs Threshold multiplier for each SNR in range(0,30)

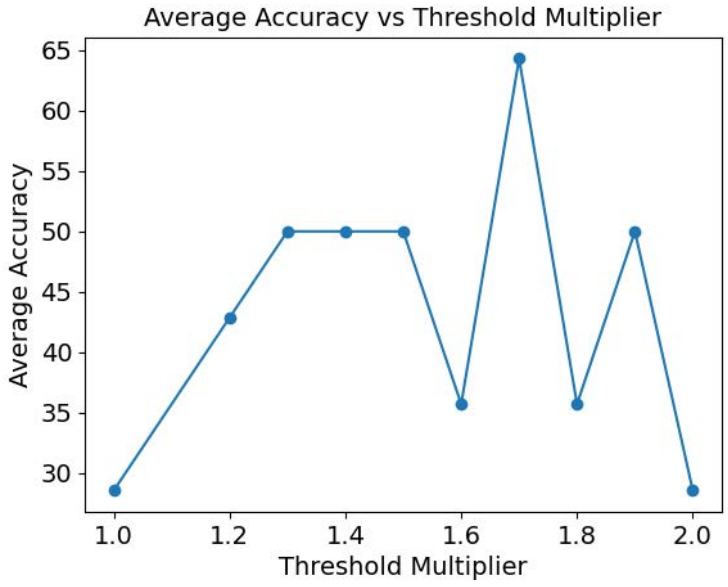



Figure 4.7: Average Accuracy vs Threshold multiplier

# Chapter 5

# Results

## 5.1 Jammer Hit Number

The outcomes of several experiments carried out during the course of this research investigation are presented in this chapter. Following the training of the model and the identification of the optimal evaluation methods, we now turn to the experimental results. In the first and most basic case, a jammer impacts the system once every sixty seconds. The scenarios then get more intricate, with the jammer having two or three system impacts.

Figure 5.1 shows the accuracy in relation to the signal-to-noise ratio (SNR) in situations where the jammer hits once, twice, and three times. The model was run about 100 times to guarantee an unbiased evaluation, and the average accuracy for each SNR was determined.

This figure 5.1 indicates high accuracy for the jammer hitting one and two times, with the two times having slightly higher score and average accuracy for three times. This could be happening because of the way accuracy is calculated (Equation 4.7).

Take the case where the jammer only hits once, for example. Two possible results for the model in this scenario are either it accurately classifies the single jammer or it achieves zero accuracy. As a result, the categorization problem here is binary. Every time the jammer strikes, the model needs to detect it accurately in order to reach 100% accuracy.

In the scenario where the jammer hits twice, the classification has three possible outcomes: 0%, 50%, and 100%. Thus, achieving 100% accuracy becomes comparatively easier. Single hits and spikes will have a major impact on the loss values plot since they are rare, as Figure 4.5 illustrates.

However, in the event that a jammer fires three times, the spike amplitude may not be able

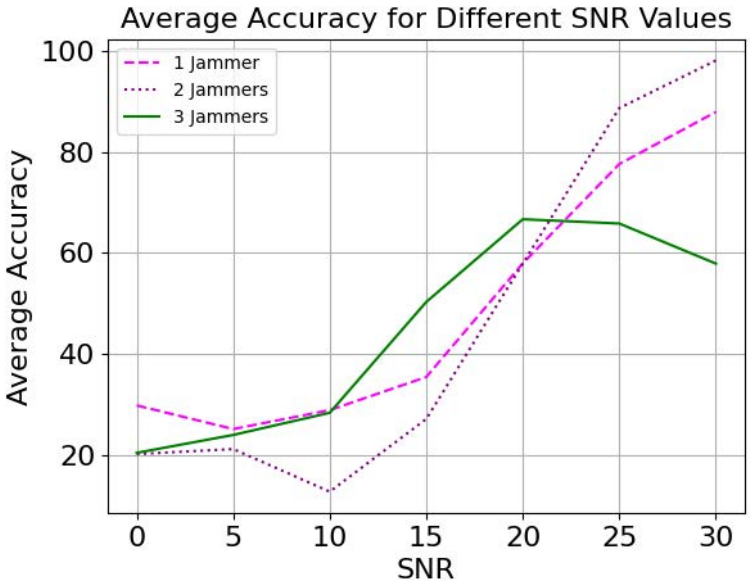



Figure 5.1: Loss vs SNR

to go above the threshold, causing the model to incorrectly identify whether there are two or one jammers. Figure 5.2 offers an example of this, showing that the model correctly classifies two of the three jammer strikes. Because noise obstructs the signal and makes proper recognition impossible, the third jammer hit, which happened near 20-second mark, is incorrectly identified.

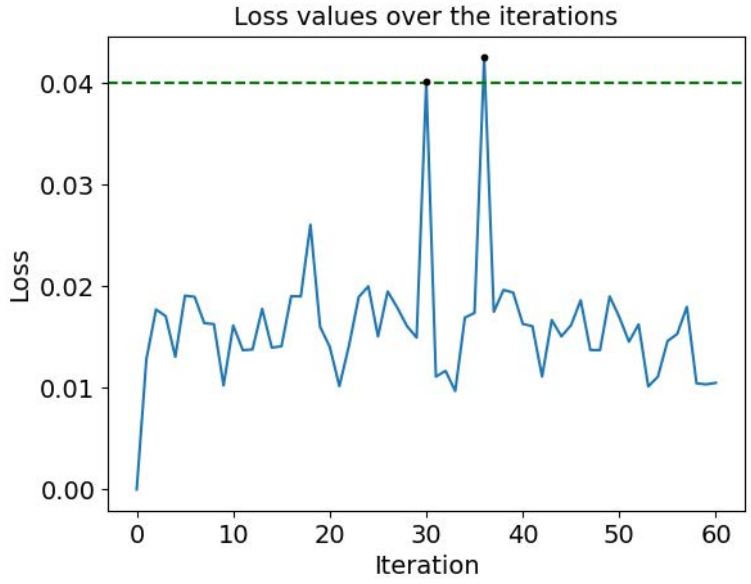



Figure 5.2: Model misclassification for jammer hitting 3 times

## 5.2 Jammer Position

This study also included an extra experimental inquiry that evaluated the model's accuracy across all signal-to-noise ratios (SNRs) taken into consideration, under various jammer strike positions throughout a 60-second period. This approach mirrors the threshold approximation discussed in Section 4.7. The position vector employed was (0, 60, 5). Accuracy metrics were computed for each position and SNR combination, and these results are visualized in Figure 5.3.

Upon observing Figure 5.3, the initial impression reveals a complex array of overlapping lines. Interestingly, the center region is more accurate than the edges, indicating that there may have been some spatial variance in the model's performance with respect to the locations of the jammers during the 60-second period. Surprisingly, in spite of these spatial subtleties, the signal-to-noise ratios (SNRs) behave in a consistent manner, albeit at varying degrees. This finding is similar to the distinctive plots of functions that have been moved vertically, where higher SNRs are associated with higher positions.

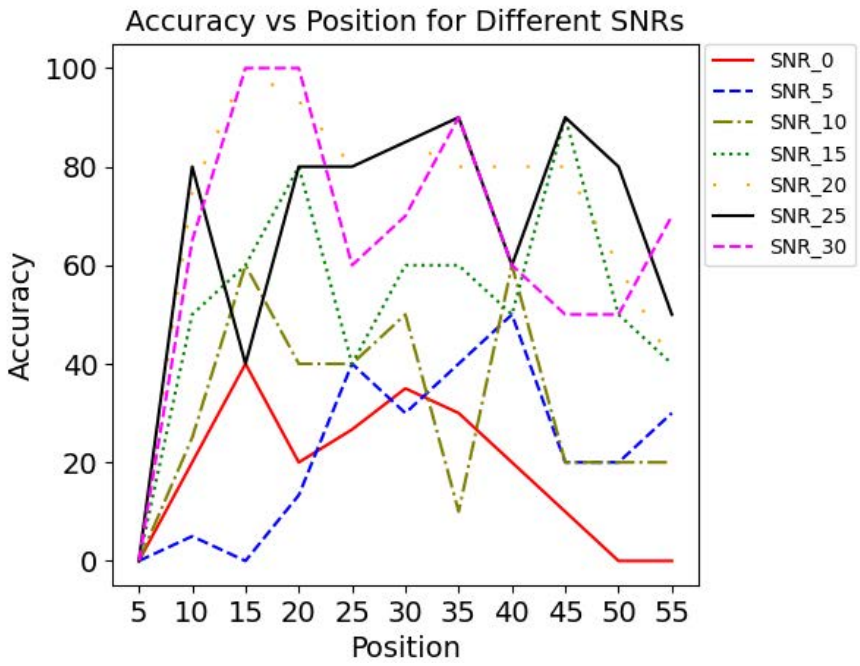



Figure 5.3: Average Accuracy vs Position for different SNRs

This phenomenon can be validated by calculating the average accuracy across all SNRs for each position and visualizing the results in Figure 5.4. The figure distinctly illustrates the model's accuracy is highest around the (10, 45) position, with poorer accuracy found below 10

and above 45. This pattern demonstrates how sensitive the model is to spatial location within the 60-second time frame, highlighting ideal performance in particular areas and showing changes throughout the range of tested positions.

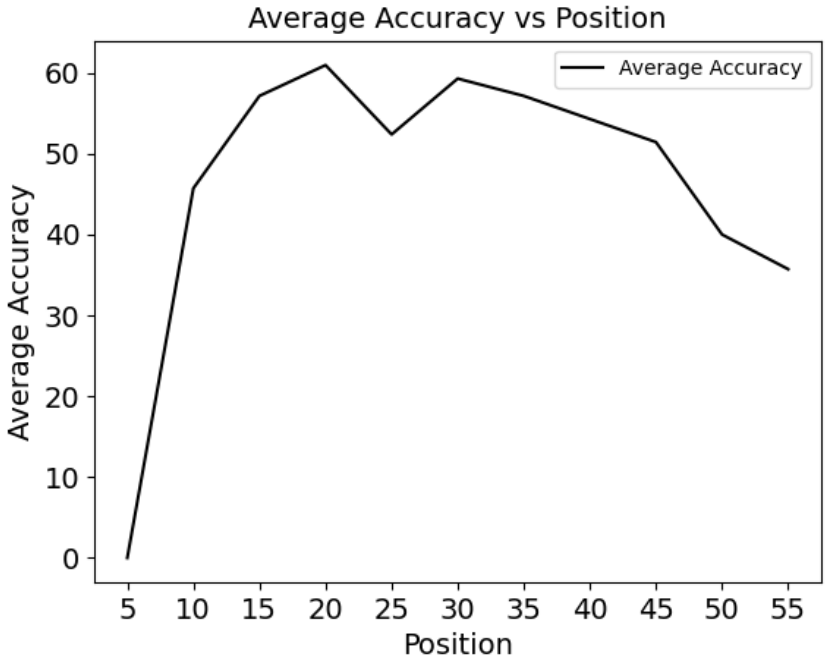



Figure 5.4: Average Accuracy vs Position

## 5.3 Discussion of Results

The findings of this investigation show how well the model can reliably identify jammer impacts that happen within a specified 60-second window. The results of multiple experiments, carried out with varying signal-to-noise ratios (SNRs) and location intervals, consistently shown that the model could accurately detect the presence of jamming signals. Additionally, the research emphasized how SNRs affect the model's functionality. Greater SNRs were typically associated with better accuracy levels, even if the overall accuracy pattern across positions reflected a steady trend. This discovery is consistent with expectations, as greater SNRs generally yield signals that are clearer and easier to differentiate, which helps the model classify data more accurately. Ultimately, the results highlight how the created model may be used practically in situations where accurate and timely jamming signal identification is essential. The model exhibits its ability to successfully support real-time decision-making processes in signal detection and mitigation strategies by reaching high accuracy in

a brief 60-second period.

# Chapter 6

# Conclusions

## 6.1 Summary

To summarize, a Deep Neural Network (DNN), namely an LSTM Autoencoder, has been developed and assessed in this thesis for the purpose of detecting jammers in RADAR systems. Initially, the paper gave a general review of RADAR systems and the fundamental ideas of Deep Learning. A thorough case study was then carried out, describing the application and assessment of the suggested DNN model's performance. The results show that even in challenging circumstances, the DNN was able to distinguish between regular RADAR signals and those impacted by jammers with a notable degree of accuracy. The results highlight how Deep Learning approaches can improve the security and dependability of RADAR in a range of applications. Subsequent investigations may concentrate on refining the model's functionality, investigating real-time uses, and tackling new risks associated with RADAR technology.

## 6.2 Future Work

Several promising directions for future research emerge from this study. Firstly, by investigating more sophisticated designs like Convolutional Neural Networks (CNNs) or attention processes, the Deep Neural Network (DNN) can become more resilient and perform better in identifying and categorizing jammers in a variety of radar situations. Furthermore, consider more advanced waveforms like OFDM is one immediate next step [19].

Furthermore, researching hostile assaults on the DNN itself (e.g. like in [18]) can be use-

ful in creating strong defenses against potential cyberattacks that aim to compromise radar systems. Further research into the integration of multimodal sensor data, such as the combination of radar and optical or LiDAR data, may result in anomaly detection systems that are more precise and trustworthy.

# Bibliography


[1] Mark A. Richards. *Fundamentals of Radar Signal Processing*. McGraw-Hill, 2005.

[2] Ming-Yang Cao, Xingpeng Mao, Xiaozhuan Long, and Lei Huang. Direction-of-arrival estimation for uniform rectangular array: A multilinear projection approach. In *2018 26th European Signal Processing Conference (EUSIPCO)*, 2018.

[3] Zhe Geng, He Yan, Jindong Zhang, and Daiyin Zhu. Deep-learning for radar: A survey. *IEEE Access*, 9:141800–141818, 2021.

[4] Ralf C. Staudemeyer and Eric Rothstein Morris. Understanding LSTM – a tutorial into long short-term memory recurrent neural networks. *arXiv*, 2019.

[5] Savvas Varsamopoulos, Koen Bertels, and Carmen Garcia Almudever. Designing neural network based decoders for surface codes. 2018.

[6] Mahmoud Said Elsayed, Nhien-An Le-Khac, Soumyabrata Dev, and Anca Delia Jurcut. Network anomaly detection using lstm based autoencoder. *Proceedings of the 16th ACM Symposium on QoS and Security for Wireless and Mobile Networks*, 2020.

[7] Yuanyuan Wei, Julian Jang-Jaccard, Wen Xu, Fariza Sabrina, Seyit Camtepe, and Mikael Boulic. Lstm-autoencoder-based anomaly detection for indoor air quality time-series data. *IEEE Sensors Journal*, 23(4):3787–3800, 2023.

[8] Merrill I. Skolnik. *Introduction to Radar Systems*. McGraw-Hill, second edition, 2001. Second Edition.

[9] Niraj Prasad Bhatta and Geetha Priya M. *RADAR and its Applications*. Publisher Name, Year. Optional additional information.

[10] Hai Deng and Zhe Geng. *Radar Networks*. Springer, 2020.

[11] Zhou Lu, Baobao Li, Xin Lai, and Haowei Zeng. A low-complexity 2d-doa estimation algorithm with massive ura. *Journal of Physics Conference Series*, 1575(1):012186, 2020.

[12] Seokhyang Cho. Direct signal-based doa estimation using ura. *International Journal of Scientific Engineering and Science*, 5(11):79–81, 2021.

[13] Antonios Argyriou. Passive angle-doppler profile estimation for narrowband digitally modulated wireless signals. In *2022 IEEE 12th Sensor Array and Multichannel Signal Processing Workshop (SAM)*, pages 246–250, 2022.

[14] R. Schmidt. Multiple emitter location and signal parameter estimation. *IEEE Transactions on Antennas and Propagation*, 34(3):276–280, 1986.

[15] Antonios Argyriou. Joint estimation of the number of antennas and aoa of a wireless communication transmitter. In *2022 IEEE International Symposium on Phased Array Systems  Technology (PAST)*, pages 1–4, 2022.

[16] Ranjan Mishra, G. Y. Sandesh Reddy, and Himanshu Pathak. The understanding of deep learning: A comprehensive review. *Mathematical Problems in Engineering*, pages 1–15, 2021.

[17] Yuanyuan Wei, Julian Jang-Jaccard, Wen Xu, Fariza Sabrina, Seyit Camtepe, and Mikael Boulic. LSTM-Autoencoder based Anomaly Detection for Indoor Air Quality Time Series Data. 2021.

[18] Dimitrios Varkatzas and Antonios Argyriou. Waveform manipulation against dnn-based modulation classification attacks. In *MILCOM 2023 - 2023 IEEE Military Communications Conference (MILCOM)*, pages 580–585, 2023.

[19] Antonios Argyriou. Range-doppler spoofing in ofdm signals for preventing wireless passive emitter tracking. In *2023 IEEE Radar Conference (RadarConf23)*, pages 1–6, 2023.