Aalto University
School of Science
Master's Programme in ICT Innovation, Data Science

Pragati Gupta

Presence detection for lighting control with ML models using RADAR data in an indoor environment

Master's Thesis
Espoo, September 25, 2021

Supervisors: Professor Arno Solin, Aalto University
Professor Manuel Roveri, Politecnico di Milano
Advisor: Omar Nasir M.Sc. (Tech.)

Aalto University
School of Science
Master's Programme in ICT Innovation, Data Science

ABSTRACT OF
MASTER'S THESIS

Author: Pragati Gupta

Title:
Presence detection for lighting control with ML models using RADAR data in an indoor environment

Date: September 25, 2021 **Pages:** 104

Major: Data Science **Code:** SCI3095

Supervisors: Professor Arno Solin
Professor Manuel Roveri

Advisor: Omar Nasir M.Sc. (Tech.)

There is an ever-rising demand for presence detection in intelligent buildings for reliable and real-time lighting control. RGB cameras have proven to be accurate for detecting human presence. However, RGB cameras pose privacy concerns and therefore cannot be used for indoor environments like offices. The solution which is commonly used in the lighting industry is thermal sensing using Passive Infrared sensors (PIR). PIRs are low-cost sensors and they have low power requirements and simple installation procedures. However, PIRs cannot detect stationary people and their output is bursty. Additionally, PIRs are prone to significant false positives which often leads to power wastage. Even though they are a sustainable solution for lighting control, there is still a high scope for improvement.

This thesis is concerned with lighting control using Frequency Modulated Continuous Wave (FMCW) RADAR, which tracks the human presence in the architectural premise. Unlike PIR, the range of the RADAR is long, and one RADAR acts as an alternative to numerous PIRs. Moreover, the RADAR data is large in volume, thus more reliable. The objective of this thesis is to measure true presence using RADARs. Deep Learning classification models are trained using RADAR data to make the machine understand the human presence. This thesis includes data collection from the RADAR, data labeling for the classification models, data visualization and exploratory analysis, and eventually data modeling using Deep Learning. The aim is to develop a generalized model for presence detection using RADAR data for lighting control. The proposed method achieves an accuracy of 98.6% when predicting the presence. These results improve over the state-of-the-art accuracy in presence detection.

Keywords: RADAR, FMCW, PIR, Rpi, FFT, MQTT, DL, ML, CNN, FNN, BCE, SGD, ASGD, ASHA

Language: English

Acknowledgements

I would like to express my sincere gratitude to Professor Arno Solin, Professor Munel Roveri for supervising me on this thesis project and guiding me in writing professionally.

Foremost, I would like to acknowledge Omar Nasir M.Sc. (Tech.) and Henri Juslen D.Sc. (Tech.) for seeing a potential in me and giving me an opportunity to work with Helvar Oy, Espoo. I am especially grateful to Omar for guiding me throughout the span of the thesis in all the domains i.e., technical, organizational and professional. Last but not the least, I would also like to express my appreciation the wonderful team of Helvar for being so supportive and extremely helpful to me.

I would like to dedicate this research thesis to my father Dr. Pawan Gupta for imbibing within me the crucial element of girl power since my toddler days. Coming from a small town in India, where men are encouraged more than women, I would like to thank him for giving me the wings to fly.

Espoo, September 25, 2021

Pragati Gupta

Abbreviations and Acronyms

AI	Artificial Intelligence
AIC	Akaike information criterion
ANIFS	Adaptive Neuro-Fuzzy Inference System
AoA	Angle of Arrival
BIC	Bayesian information criterion
CMOS	Complementary Metal Oxide Semiconductor
CNN	Convolutional Neural Networks
CW	Continuous Wave
DALI	Digital Addressable Lighting Interface
DFT	Discrete Fourier Transform
ES	Embedded Systems
FFT	Fast Fourier Transform
FMCW	Frequency Modulated Continuous Wave
FN	False Negatives
FNN	Feedforward Neural Networks
FOV	Field of View
FP	False Positives
FPA	Focal-plane arrays
FT	Fourier Transform
GDPR	General Data Protection Regulation
HVAC	Heating, Ventilation, and Air Conditioning
IF	Intermediate Frequency
IoT	Internet of Things
LADAR	Laser detection and ranging
LWIR	long-wave infra-red
LiDAR	Light detection and ranging
ML	Machine Learning
MLP	Multi-Layered Perceptron
MQTT	Message Queuing Telemetry Transport
MTI	Moving Target Indicator

PBT	Population Based Training
PDF	Probability density function
PIR	Passive infrared
RADAR	Radio detection and ranging
RELU	Rectified Linear Unit
RFID	Radio Frequency Identification
RGB	Red Blue Green
RX	Reflected signal
Rpi	Raspberry Pi
SNR	Signal to Noise ration
SONAR	sound navigation and ranging
SVM	Support Vector Machine
TDA	Topological data analysis
TN	True Negative
TP	True Positive
TX	Transmitted Signal
UHF	Ultra High Frequency
YOLO	You only look once

Contents

Chapter 1

Introduction

Lighting applications account for 19% of the world's energy use and 6% of all greenhouse emissions. In the United States itself, 65% of the energy consumption accounts for commercial and industrial sectors, and 22% of it is used for lighting. In European Union, the total domestic lighting usage is around 86 TWh and is predicted to increase much more in the coming years [47]. Commercial Buildings are classified as the highest energy consumers. The total annual energy consumption in office buildings varies from 100 to 1000 kWh/m^2, depending on parameters, such as geographic location, use, and type of office equipment, type of envelope, operational schedules, use of HVAC systems, and type of lighting [57]. In Northern Europe, office energy intensity lies in the range 269 to 350 kWh/m^2 yr, and for offices all over Europe, it is about 306 kWh/m^2 yr [19].

With the establishment of legislative frameworks like Energy Performance of Buildings Directive 2010/31/EU (EPBD) [2] and the Energy Efficiency Directive 2012/27/EU [4], European Union places a high demand for better and more energy efficient buildings. The goal is to achieve a highly energy-efficient and decarbonised building stock by 2050 [2]. According to previous research, modern office buildings have the potential for energy savings at a magnum level. Electric lighting is one area where energy savings are possible at a reasonable cost in new buildings as well as in retrofit projects [19]. Lighting is one of the most cost-effective ways to reduce CO_2 emissions. An effective way to address this energy problem is to deploy automatic lighting control systems. Automatic lighting controls can reduce energy consumption by up to 50% in existing buildings and by 35% in new constructions [58].

A lighting control system is an intelligent network-based solution that incorporates networking between various system inputs and outputs related to lighting control. The lighting control system is composed of a networked system of devices that together satisfied the wholesome functionality. These

devices may include occupancy sensors, photocells, imagers, cameras, etc. Lighting control systems provide the right amount of light where and when it is needed. Intelligent lighting control not only includes automatic turn off and turn on of the luminaries as per the occupancy, but it also includes automatic adjustment of the light intensity using dimmer. This dimming functionality along with daylight harvesting leads to huge energy cuttings. This basic functionality can be used to generate the benefits of flexibility to satisfy the visual needs of the user to improve energy efficiency and sustainability. Recently two new dimensions have been added to lighting control systems i.e., Well-being and Insights. According to Helvar, their lighting control system enhances the well-being of the users, boosts productivity, and makes the users feel good. The Lighting control system maintains the circadian rhythm and helps avoid the negative impacts of the dark season in cold countries like Finland. Many organisations that have studied this field in recent years reported productivity gains are between 1.5–12% with intelligent lighting control systems [28].

The main and the foremost functionality of the lighting control system still remains of presence/occupancy detection. Presence detection has always been one of the most discussed topics in the industry. Unfortunately, we still lack the technology able to accurately detect human presence, be it moving or stationary. The extensive use of data-generating devices has given rise to data-driven Machine Learning models for occupancy detection. This thesis is about effective presence detection using Machine Learning.

1.1 Problem Statement

Presence detection has been one of the hot topics in the industry [56]. RGB cameras have proven to be quite accurate for detecting human presence. However, there are some limitations of using RGB in indoor environments for industries like the lighting industry. RGB camera hinders people's privacy. People are usually not comfortable with the idea of a camera installed near them in order to track their activity. This is the main reason why RGB cameras cannot be used for indoor environments like offices.

The solution which is common is thermal sensing using Passive Infrared Sensors (PIR). PIRs are low-cost sensors and they have low power requirements and simple installation procedures. The problem with PIR is, it cannot detect stationary people and their output is bursty. Due to this reason usually, a timer is associated with the PIR data. The timer is reset to 0 whenever there is a movement in the room. This means if the PIR sensors do not track any activity for the past five minutes (as an example for chosen timer range),

only then do the lights turn off. This often leads to power wastage. Even though PIR sensors are a sustainable solution for lighting control, there is still a high scope for improvement.

There are many other solutions proposed by the researchers in the last two decades for an intelligent solution like presence detection with CO_2 monitoring, using pressure sensors, ultrasounds, range finders, Wi-Fi, energy traces, etc., the details of which are provided in Chapter 2. However, the problem still remains of efficiency and commercialising these solutions in the buildings or office spaces. Many of these solutions are not standalone i.e., they are not operational without workstations and cannot be embedded as edge devices, while some pose a threat to breach of personal privacy. Some of the solutions are extremely expensive to be deployed for lighting control, while some solutions are hard to scale and integrate with existing solutions. Moreover, other limitation of present occupancy detection includes reliance on a single occupancy sensor in a zone under control, lack of data analysis of the measured sensor signals, and unnecessary switching [18].

This thesis is about lighting control using Frequency Modulated Continuous Wave (FMCW) RADAR Data, which tracks the human presence in the room. Unlike PIR, the range of the RADAR is long, and one RADAR can offer as an alternative to numerous PIRs. Moreover, the RADAR data is fast and more reliable. The objective of this thesis is

- To measure true presence detection using FMCW RADARs with Machine Learning classification models. This detection should be real-time, faster, reliable, and accurate than the existing solutions.

- To analyse if FMCW RADARs can act as a substitute to traditional PIRs in a commercial scenario. FMCW is more expensive than a PIR, however, it is a replacement for multiple PIRs. The target motive is to reduce the number of false negatives and False Positives of the traditional PIRs, i.e., reducing the cases where the person is sitting in the space, and still, the PIRs indicate no presence and reducing the cases where the person is not there and still the PIRs detect presence. It is worth noticing that in case of PIRs False Negatives is the main problem as it does not detect stationary humans.

- The third and very important goal of the thesis is to build a generalized model for presence detection. This means that the model generated should work for all scenarios, i.e., the same model may be used for all the meeting rooms, for all the cafeterias, or for all the open areas. This is because for a simple scenario like lighting control, retraining

the model as per different environments makes less sense. One model for presence detection should work for all the scenarios.

1.2 Structure of the Thesis

Chapter 2 discusses the previous study in the domain of presence detection. It explains in brief about the sensors and techniques available in the industry for presence detection and also reflects on the advantages and disadvantages of using those techniques for lighting control in an indoor environment. Further, the basic concepts about the RADAR (primarily FMCW) are discussed. It is important to discuss the physics behind the FMCW RADAR to understand clearly why specific steps were undertaken in the upcoming chapters. Furthermore, this chapter also provides an overlook on Machine Learning for this thesis.

Chapter 3 reflects on the idea how the data pipeline was constructed for data collection. It explains in detail how the data from the RADAR was collected and how this collected data was labelled into classes. Finally, it gives insights on the analysis and feature engineering done on the RADAR data. Chapter 4 portray the idea behind Machine Learning in detail and illustrate the different ML concepts used in this thesis, such as ML networks, optimization techniques, loss functions, activation functions etc.

Chapter 5 demonstrates how the implementation of the Deep Learning models has been carried out, and what approaches have been used to optimize the results. It also discusses the baseline algorithm which gives good solution without using Deep Learning models. The objective of the Deep Learning models is to defeat the accuracy of the baseline algorithm by good margin. Chapter 6 discusses and visualize the different versions of the chosen networks. It compares different versions of the network with different parameters. The chapter also discusses the statistical metrics obtained by different models and the performance of these models on the test dataset.

Finally, in Chapter 7 and Chapter 8, concluding remarks have been made. It discusses the thesis in a holistic and brief manner and also talks about the future scope of the project.

Chapter 2

Background

This chapter reflects on the literature related to the applications of presence detection. In addition, background information of the fundamental concepts that underlie this thesis research will be addressed. The concepts include RADAR technology and Artificial Intelligence.

2.1 Previous Study

Human Detection can be conducted using a variety of methods, such as Thermal Imaging, Infrared Cameras, Video Cameras, Passive Infrared Sensors (PIRs), Carbon dioxide concentration, Pressure sensors, and Range Finders like RADAR, SONAR, and LiDAR.

2.1.1 Cameras and Imagers

In comparison to other sensors, Cameras are relatively affordable, easy to access, provide reliable, high resolution, and easy to interpret information. This is why computer vision has always been a hot topic of the industry.

Sarkar et al. [58] used a high dynamic range Complementary Metal Oxide Semiconductor (CMOS) video camera sensors for occupancy sensing and other functionalities like daylight estimation and lighting control. The sensors provide luminance along with the color information. It also has the capability of detecting small movements several feet away from the camera as long as it has an adequate resolution. However, the system has some feasibility issues like occupancy detection in a region of total darkness. Also, the system can not act as a standalone system, it needs to be set up with a workstation.

Shih, Huang-Chia [60] addressed the problem of occupancy detection and

people tracking using an image-based depth sensor and a programmable pan-tilt-zoom (PTZ) camera. This system detects the movement even under the dim lighting scenarios. The system uses Support Vector Machine (SVM) which is a non-parametric Machine Learning algorithm. The depth image sensor enables the contour information of the occupant for better activity recognition. The system combines vision based and sensor-based cameras, in order to fill the gap between the subjective sensing of people and system control parameters. However, this approach is not suitable for working spaces as people might not be comfortable with surveillance cameras mounted near them.

Mikkilineni et al. [42] proposed a novel occupancy sensing method that enabled temporal minimization of building energy consumption without privacy concerns, based on long-wave infra-red (LWIR) focal-plane arrays (FPAs), or thermal imagers, that detect thermal energy rather than visible light. The solution captures the energy and creates an image whose pixels represent the temperature. The paper used computer vision based techniques to overcome the accuracy issues suffered by traditional PIR based sensing, especially when the person is still. The paper provides the solution for multiple people tracking and counting at the same time while addressing the problem of breach of privacy with normal cameras.

With the advancement of Computer Vision and Deep Neural Networks, occupancy detection using RGB cameras has become quite mundane. However, for the purpose of the lighting industry, the use of RGB cameras is quite plagued with a lot of loopholes. The first and foremost problem is privacy infringement. It is against the privacy regulations to record people's activity in the working offices to regulate lighting. The second problem of using RGB cameras is, despite giving the solutions real-time, it needs a lot of computational power. Processing the video and then feeding it to a Neural Network is difficult to be solved using embedded devices. Signal Processing of cameras is more difficult than signals from other sensors. This is the reason why the cameras or the imagers are not used in the lighting industry which is highly performance-driven.

2.1.2 Passive Infrared Sensors (PIRs)

PIR sensors have been widely used for human detection systems as they are cheap and power consumption is relatively small. The main advantage of PIRs is their unobtrusive and privacy-preserving interaction.

Teixeira et al. [63] provided a survey of the multidisciplinary literature of human-sensing including presence detection. From the presence detection perspective, PIR sensors are often used for binary sensing as they are cheap

sensors and have very low power requirements. However, the main limitation of this is that PIRs tend to produce bursty positive detection and a large number of false-negative detection, for example, PIRs cannot sense people who are standing still.

Kim et al. [31] studied the use of passive infrared (PIR) sensors along with a door sensor to ensure the accuracy of occupancy detection. The algorithm could detect with an improved occupancy detection accuracy rates of 99.8% (when the number of occupants is 1 to 6) and 90.1% (when the number of occupants is: 1) were achieved. The results of the experiments show that the occupancy detection algorithm along with a door sensor could be used to improve the occupancy detection accuracy rates of the PIR sensors.

The use of only PIR sensors for occupancy monitoring does not offer enough savings and depends a lot on the type of the building and its occupancy. Aslam et al. [41] presented a paper using a data fusion approach of traditional occupancy monitoring PIR sensor with passive Radio Frequency Identification (RFID). The users of the experiment are provided with Ultra High Frequency (UHF) tags. The RFID readers which are mounted on the door, detects any tag which passes through them.

However, from the saving perspective, The time duration for which the maximum number of lights are switched on all at the same time, is much less than PIR. Additionally, the time duration for the minimum number of lights switched on for this method is longer than using PIRs standalone. Due to the less reliability in signal propagation that comes with passive RFID technology, an obvious malfunction of this approach comes when a tag is not read.

2.1.3 Sensing using Carbon Dioxide

Cali et al. [16] proposed an algorithm to predict the occupancy detection using the concentration of carbon dioxide with cheap CO_2 sensors. The algorithm uses the mass balance equation and assumes air to be homogeneous, i.e., the ideal mix of supply and room air. The results of the algorithm provide a correct presence profile up to 95.8% of the time, whereas for the people counting, it correctly predicted up to 80.6% of the time. The authors of the paper used presence patterns as inputs of stochastic user models to predict occupancy.

Ansanay-Alex, Guillaume [8] proposed an algorithm to use only indoor carbon dioxide concentrations to provide an estimated occupancy profile in an office setup. This research simply used the gradient of the monitored CO_2 in order to make a detection. The method is well functional for arrivals and departures in a closed space. Nonetheless, the results are only considerable

if the air change rate of the room remains inert for some duration of time, i.e., window opening or door opening may lead to wrong results. These are also rarely used for the control of lighting systems due to their slow response and sensitivity to environmental conditions [63].

2.1.4 Pressure Sensors

Another approach to predict the occupancy was proposed by Labeodan et al. [34] who evaluated the performance of chair sensors using sensing techniques based on strain, vibration, and a mechanical switch for occupancy detection in an office space. All three sensors are susceptible to the impact force created by users when they move the chair or when they seat in it. All three types of sensors were prone to false negatives, however only the vibration and strain sensors gave false positives. At the end of the experiment, the mechanical sensors worked best with an accuracy of 99%. The performance of the mechanical switch was compared to traditional PIRs for the purpose of lighting control. The results demonstrated that the chair sensors performed better than the PIR sensors. Still, a Pressure sensor can not be used exclusively for lighting control, they need to be used alongside other sensors as it is impossible to track the initial movement which is crucial for lighting control.

2.1.5 Multiple Sensors

To compensate for the drawback of individual single sensor, many researchers have tried assembling multiple sensors into one system in order to achieve better reliability and accuracy.

Dodier et al. [18] proposed an improved detection of building occupancy using multiple independent detectors by applying graphical probability models called belief networks. The multiple sources of information are aggregated together, creating more powerful sensors, such as carbon dioxide concentration and space relative humidity.

Ekwevugbe et al. [22] described a novel solution for building occupancy detection using a sensor fusion model based on an Artificial Intelligence using Adaptive Neuro-Fuzzy Inference System (ANFIS) algorithm. The system monitors climatic variables, energy data, and indoor events obtained from a non-domestic building to infer occupancy patterns.

Climate variables include data like indoor and outdoor variables including temperature, humidity, illumination, VOC, and CO_2 levels. Energy data includes using existing sub-metering (for electricity), appliance case temperature monitoring, and monitoring of personal computer (PC) temperature.

Indoor data includes variables such as sound data and PIR data. The resulting occupancy sensor from this research possessed improved reliability that may not be obtained from a single sensor.

2.1.6 Range Finders

The Range Finders are devices that work by sending the signals towards the objects, and then calculate the distance by observing the timing or the energy of the received signals reflected from those objects. These devices then extract the 2D or 3D snapshots of the environment. It is worth mentioning that in an indoor environment, multiple reflections, scattering, and cluttering add considerable noise to their range [63]. This is the reason why the range finders are usually confined to detection in outdoor spaces like Self Driving Cars. Based on the medium of propagation, range finders are classified into different classes, such as SONAR (sound or ultrasound), RADAR (radio waves), LiDAR (light), and LADAR (laser).

Among all of the range finders, laser-based ranging (LADAR) is least sensitive to multiple reflections, scattering and cluttering due to its relatively higher frequency. Arras et al. [10] presented a paper for the detection and tracking of people with LADAR range finders. The authors of the paper used the Supervised Learning technique to create a classifier for the detection of humans. Unlike vision sensors, LADARs provide a large field of view (FOV) and are almost unaffected by other conditions. The sensors are placed very close to the ground and extract the readings from the legs of the people. A 2D range image is created out of the received signals. The common approach is to extract legs by detecting moving blobs. The experiment obtained detection rates of over 90%. However, the problem still remains of the false negatives. LADARs are still not a reliable solution to be deployed as a standalone. Usually, they are paired up with cameras for the gaming industry. But they remain less efficient for working premises.

RADAR is one of the most prominent range finders used for occupancy detection. It can give accurate results in all types of luminance scenarios and all weather conditions. It gives accurate measurement due to the use of a shorter wavelength. Gürbüz et al. [27] targeted the problem of human target detection using synthetic aperture RADAR (SAR). The human spectrograms are used for the detection as the spectrograms for humans are unique in comparison to other objects. Simulations show that spectrograms can detect and identify human targets in low noise. However, this approach works only if the target is close to the antenna and signal to noise ratio (SNR) is high. In an indoor environment where the SNR is high, it is very difficult to provide reliable estimation using this method.

Apart from the most common motion sensors i.e., PIRs, other sensors called Doppler shift sensors also exist. Doppler shift means the apparent shift in the frequency of the radial component of the velocity of the object. Geisheimer et al. [25] presented a high-resolution Doppler model for the walking human using Continuous Wave (CW) RADAR. The outputs of the RADAR were compared with the infrared motion capture system to create the theoretical Doppler signal for a walking human. The experiment succeeded in unlocking information from the RADAR gait signature, which can be useful in occupancy detection. However, this experiment was confined to only standing humans. Further research is maybe required in order to create different signatures for different postures.

In [66], Zhou et al. used a system that analysed the Doppler RADAR signal of the heartbeat to detect people occupancy in the room. A complex signal processing is used to process these weak signals. Due to the Doppler effect, the periodic reflected signals from the chest gets a phase shift which is proportional to the velocity. The experiment concluded that less than two persons in the room could easily be detected using a single antenna of the receiver. To count more people, more antennas are required.

The ability of the RADARs to penetrate objects like a wall, make them less likely to be used for occupancy detection inside the room. For instance, Falconer et al. [23] demonstrated a robot-mounted motion detection RADAR which is suitable for operations even through the wall (gypsum board walls or hardwood walls). Time-domain and frequency domain signals are captured to detect not only the human occupancy but also human's particular activity like resting, walking, talking. In this thesis, a specific type of RADAR called FMCW RADAR has been used which not only detects moving objects easily but also less prone to noise. FMCW has been chosen as it performs the velocity and range measurement efficiently and therefore it is a good solution for the lighting industry. More advanced details about RADAR and FMCW RADAR will be covered in the following section.

2.2 RADAR technology

RAdio Detection And Ranging is an electromagnetic range finder device that observes the echoed signals by their time delays. The RADAR can operate in the radio or in the microwave domain. It is used for various applications on the ground, on the sea, and in the space. The applications of RADARs are controlling the air traffic, ship safety, sensing remote places, military applications etc [61].

As shown by Figure 2.1, the RADAR consists of a **Transmitter** which

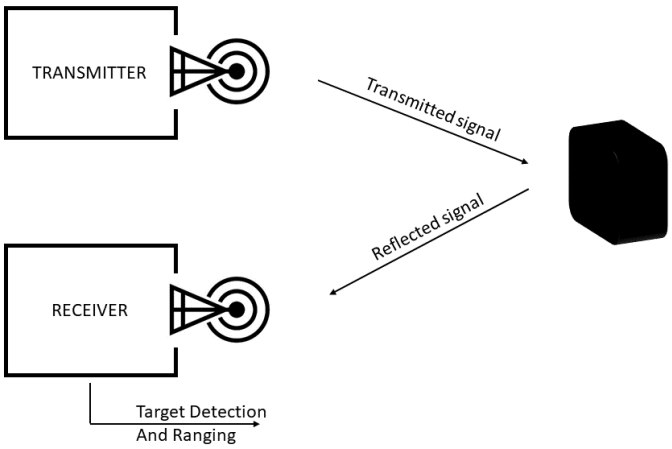


Figure 2.1: The basic principle of RADAR

is used to send the signals, a **Receiver** which is used to receive the echoed signals. The received signals are then analysed to process the range and the velocity of the targeted object.

2.2.1 Types of RADARs

On a broad level, the RADARs can be classified into Pulse RADAR and Moving Target Indicator RADAR.

1. **Pulse RADAR**

   In Pulse RADAR, the RADAR operates with pulse signals. The RADAR sends the signals for a pulse of duration and then waits to receive the signals. The Pulse RADAR is further classified into Basic pulse RADAR and Moving Target Indicator RADAR (MTI).

   (a) **Basic Pulse RADAR**

   It uses a single antenna for both sending and receiving signals. The antenna transmits signals after every clock pulse. This type of RADAR can only detect stationary objects.

   (b) **Moving Target Indicator RADAR (MTI)**

   This type of RADAR uses pulse signals to compute even the non-stationary objects. This uses the Doppler effect to distinguish the non-stationary object from stationary object.

2. **Continuous Wave RADARs**

As compared to Pulse RADAR, Continuous Wave RADARs sends continuous signals in order to detect objects. Both stationary and non-stationary objects can be detected using Continuous wave RADARs. Continuous wave RADARs are further divided into Unmodulated Continuous Wave RADAR and Frequency Modulated Continuous Wave RADAR (FMCW).

(a) **Unmodulated Continuous Wave RADAR**
This type of RADAR detects non-stationary objects using continuous wave signals. It is also called Continuous Wave Doppler RADAR. In this scenario the frequency of the continuous signal remains unchanged over time.

(b) **Frequency Modulated Continuous Wave RADAR (FMCW)**
This is the most advanced type of RADAR which can be used for detecting both stationary and non-stationary objects. An FMCW RADAR requires two antennas: one for transmitting and the other one for receiving. The distance and velocity of the objects can be easily computed using Fourier transformations (FFT) of the received signals. A more detailed explanation about FMCW is provided in the following section.

2.2.2 Frequency Modulated Continuous Wave RADAR

Continuous Wave RADARs are popular because of low power consumption and simple radio architecture. Also, these types of RADARs cancel out clutter noise by proper adjustment. The main advantage of the FMCW remains good range resolution, resistance to interception, simple solid-state transmitters. FMCW method performs the velocity and range measurement efficiently and therefore it is a good solution for the lighting industry [9] [36].

FMCW generates a linearly increasing frequency signal viz., sawtooth or sinusoidal wave using a synthesizer. The generated signal is then transmitted using one or more antennas. Each signal is called **chirp/sweep**. A chirp is a signal whose frequency increases linearly with time [53]. A chirp can be characterized by the four parameters broadly-

- start frequency which is generally called carrier frequency and denoted by f_c,
- a chirp bandwidth(B), which is the difference between the maximum reachable frequency $f_c + B$ and the carrier frequency f_c.

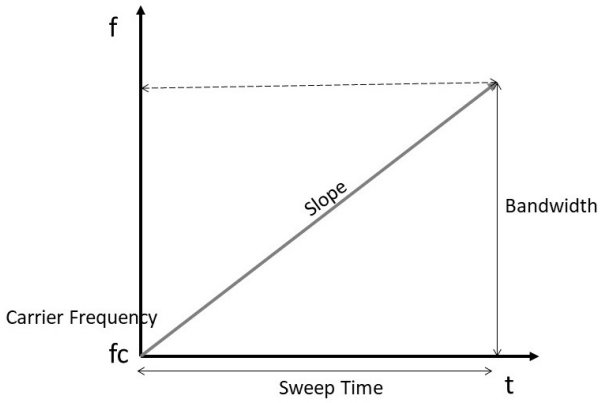
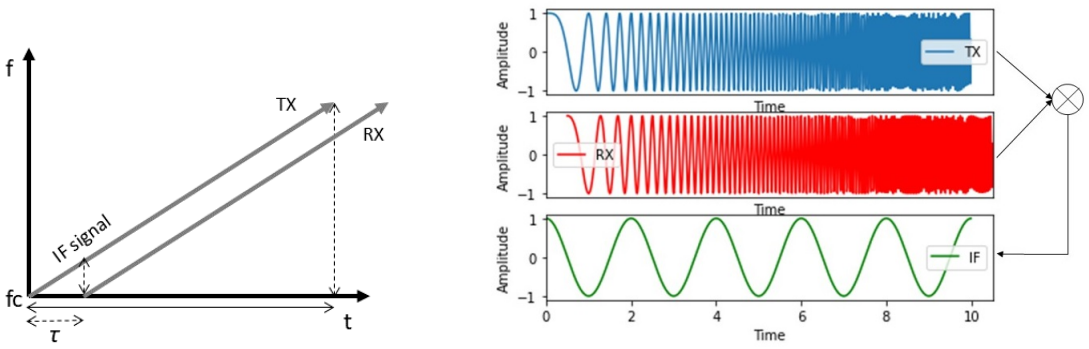


Figure 2.2: A basic chirp/sweep signal

- the sweep time(t_p), which is the duration for which an FMCW sends one chirp,
- The slope($k = B/t_p$), of the chirp which is defined as rate at which the chirp ramps up. It is the ratio between the chirp bandwidth and the sweep time.

Therefore, the frequency of the signal f_t at any time t, with carrier frequency f_c , amplitude A_t, slope k, is given by the expression

$$f_t = A_t \cos[2\pi(f_c t + \frac{k}{2}t^2)].$$



(a) IF signal (b) IF signal from TX and RX

Figure 2.3: Instantaneous Frequency (IF)

The transmitted signal (TX) is transmitted through the transmitter antenna. The TX signal strikes the surface of the object and gets reflected back. The receiver of the FMCW receives this reflected signal (RX) via its receiving antenna. RX signal is just the same TX signal but delayed.

To compute the range, velocity, or phase of the object, the most important factor to compute is the **Baseband signal** with frequency IF known as **Intermediate Frequency (IF)**. The frequency of the Baseband signal is the difference of the instantaneous frequency of the TX-chirp and RX-chirp. Figure 2.3a explains the IF signal in a mathematical way. It is to be noted that for a stationary object, irrespective of when the chirp was transmitted, this IF signal will remain the same, as this IF signal is proportional to the delay time which is proportional to the distance of the object from RADAR.

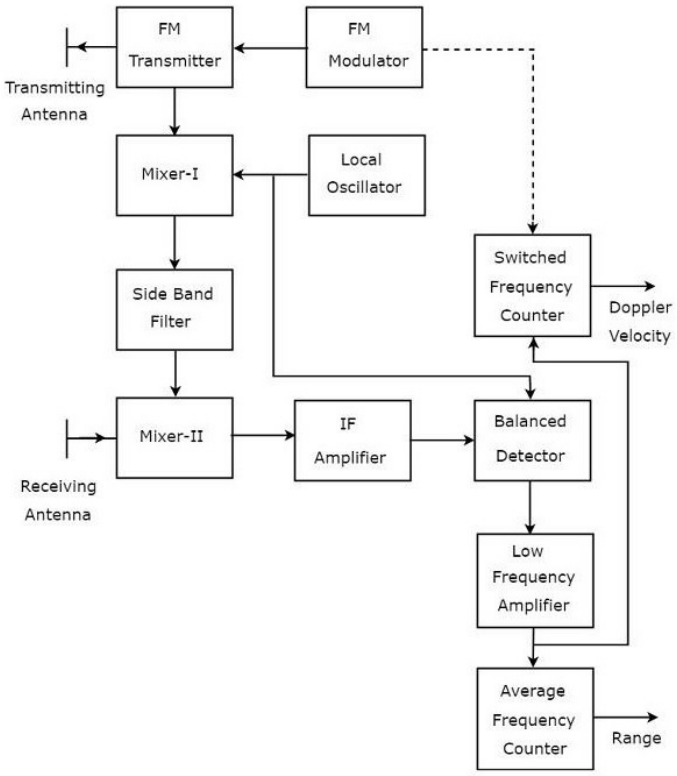


Figure 2.4: Block diagram for FMCW

Source: RADAR Systems - FMCW RADAR [5].

Figure 2.4 explains in detail how this Baseband signal is produced. The **Frequency Modulator** produces a modulated frequency signal TX which

is transmitted using **Transmitter**. The **Local Oscillator** produces a signal having Radio Frequency f_{rf}. The output of the Local Oscillator is then fed to a mixer along with the output of the Transmitter. The **Mixer** computes then sum and the difference of the Transmitted frequency and the RF signal from the Local Oscillator. A low pass filter then only allows the lower frequency to be passed i.e., only the difference of the Transmitter and the RF signal $f_o(t) - f_{rf}$ is allowed [5].

Another **Mixer**, is fed the received signal from the receiving antenna f_r and the output of the low pass filter. It is worth mentioning that the received signal RX is just a delayed version of the transmitted signal TX. It then computes the sum and the difference of these signals again, i.e., $f_r + f_o(t) - f_{rf}$, f_r - $(f_o(t) - f_{rf})$ [5]. An **Amplifier** amplifies the differenced output of the Mixer, i.e., only f_r - $f_o(t) + f_{rf}$ is amplified. The **Balance Detector** which is connected to both the Amplifier and the Local Oscillator computes the difference of the both input signals, hence generating a signal of frequency $f_r - f_o(t)$. This output is then inputted to **Low Frequency Amplifier** which amplifies the input frequency. This resultant signal is then called the Baseband signal [5].

In order to understand the functionality of FMCW in better detail, we will divide the topic into 3 parts.

- Part 1: Evaluation of the range of the target object.
- Part 2: Evaluation of velocity of the target object.
- Part 3: Evaluation of phase of the target object.

These parts are described in detail in the following subsection.

2.2.3 Signal Processing for FMCW

The Baseband signal is constructed by the RADAR when an object reflects the signals. However, in a real scenario, there are multiple objects in the scene resulting in multiple Baseband signals. Therefore RADAR receives a combination of these Baseband signals. To differentiate different objects, RADAR should be able to differentiate between different signals. To do so, RADAR uses Fourier transformations. A summary of Fourier transformations is provided in the Appendix section A.1.

Range Evaluation for Stationary targets

Consider an object as distance d from the FMCW RADAR. From Fig. 2.3a, we can clearly say that the frequency of the IF signal is

$$f_\tau = k\tau, \tag{2.1}$$

where k is the slope of the chirp, τ is the delay in the RX signal. But, $Time = Distance * Speed$, Therefore, substituting $Distance = 2d$ (To and fro distance covered by the signal from RADAR) and $Speed = c$ (speed of light) into Eq. (2.1), we get

$$f_\tau = k\frac{2d}{c}. \tag{2.2}$$

Eq. (2.2) indicates the Intermediate frequency of the Baseband signal obtained from an object placed at a distance d.

In a real scenario, an FMCW receives a signal which is a combination of different Intermediate frequencies. These IF signals are produced by objects placed at different ranges. From Eq. (2.2), it is evident that these IF signals are proportional to the distance d at which the object is placed. This is where Fourier Transformations discussed in section A.1 comes into the picture.

To calculate the range of different objects present in the view, the output received by the FMCW Receiver is transformed using FFT. This result is called **Range FFT** of the received signal RX.

Range Resolution is defined as the minimum distance that should be there in between two objects to differentiate them for FMCW. As previously explained, in order to differentiate two objects, the frequency difference should be at least 1/T i.e.,

$$\Delta f > \frac{1}{T}. \tag{2.3}$$

Consider two objects, placed at a distance of d_1 and d_2, from Eq. (2.2), their respective IF frequencies are given by

$$f_{\tau 1} = k\frac{2d_1}{c} \tag{2.4}$$

and

$$f_{\tau 2} = k\frac{2d_2}{c}. \tag{2.5}$$

Therefore,

$$\Delta f = k\frac{2\Delta d}{c}. \tag{2.6}$$

Substituting Eq. (2.6) in Eq. (2.3) and T with the observation period T_c, we get:

$$k\frac{2\Delta d}{c} > \frac{1}{T_c} \implies \Delta d > \frac{c}{2kT_c} \implies \Delta d > \frac{c}{2B}, \tag{2.7}$$

where B is the Bandwidth of the chirp and is equal to kT_c.

Hence, the Range Resolution (d_{res}), depends only on the speed of the light and Bandwidth of the chirp [53], therefore:

$$d_{res} = \frac{c}{2B}. \tag{2.8}$$

Maximum Range of the RADAR is defined as the distance beyond which FMCW fails to operate. From Eq. (2.2), substituting d with d_{max}, we get $f_{IF_max} = \frac{k2d_{max}}{c}$. Replacing f_{IF_max} with F_s, where F_s is the maximum sampling rate of RADAR, we have:

$$d_{max} = \frac{F_s c}{2k} \tag{2.9}$$

This means we can increase or decrease the maximum field of view range by changing either the maximum sampling rate or the bandwidth.

- Keeping Maximum Sampling rate constant F_s, the Maximum Range of the RADAR can be manipulated by changing the Bandwidth.
- Similarly, Keeping Bandwidth constant, the Maximum Range of the RADAR can be manipulated by changing the Maximum Sampling Rate F_s.

Velocity Evaluation for Moving targets

According to section 2.2.3, to separate different objects in the view, Range FFT is applied on the received RX signal. However the question to ask here is *"How can the two objects placed at the same radial distance from FMCW be separated?"* [53]. The FFT of the two objects placed at the same distance will give peak bins at the same frequency. Doing further signal, this problem can be resolved using the velocities of these objects which are at the same distance. It is to be noted that these objects can be differentiated if the velocities of these objects are different.

To solve this problem, Fourier Transformation comes into the picture. Nevertheless, instead of doing Range FFT, **Doppler FFT** is used.

Going a step further to what is explained in subsection A.1, Note that using Euler's formula, any sinusoidal signal in rectangular form can be expressed in the form of a complex exponential signal. More detailed explanation is provided in the section A.2.

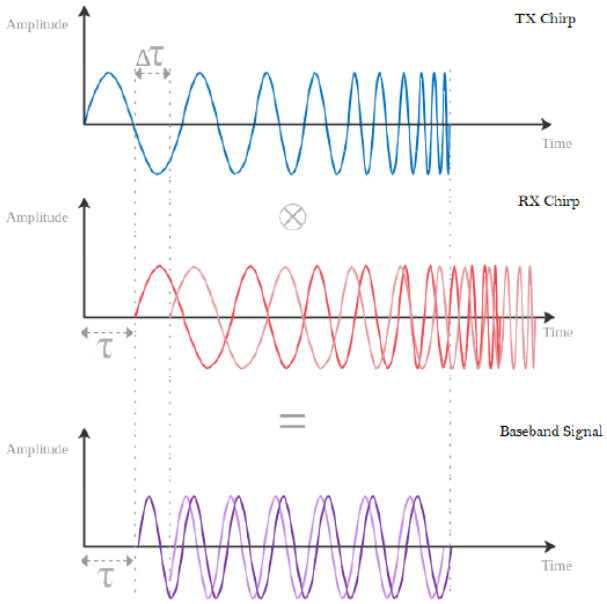


Figure 2.5: Two consecutive chirps giving rise to a phase difference

Source: Nilsson, J., Hassbring, L. (2020). Machine Learning for FMCW RADAR Interference Mitigation [48].

Now, consider the Fig. 2.5, what happens if the target object moves by a distance of Δd. It can be surely said that the frequency of the Baseband signal i.e, Intermediate frequency will be change by $\frac{2\Delta d}{c}$. This value is close to zero since Δd is very small. However, the initial phase of the IF signal will definitely change, even though the intermediate frequency is the same. Change in Δd makes $\Delta\tau$, round trip delay. Phase θ of a wave at any time t, can be represented as

$$\theta = 2\pi f t. \tag{2.10}$$

Therefore change in theta with change in time Δt is

$$\Delta\theta = 2\pi f \Delta t. \tag{2.11}$$

For the purpose of simplicity, we represent change in phase i.e., $\delta\theta$ with ω. It is known that, $\Delta t = \frac{2\Delta d}{c}$ and $f = \frac{c}{\lambda}$, therefore putting everything together

$$\omega = \frac{4\pi\Delta d}{\lambda}. \tag{2.12}$$

This is the change in phase of the IF signal, when an object moves by distance

$$A_t = A\sin[2\pi\frac{k2d}{c}t + \frac{4\pi\Delta d}{\lambda}]. \tag{2.13}$$

The phase of the IF signal is very sensitive to small change in object range [53].

To calculate the velocity of the object, two chirps are transmitted separated by time interval of T_c. It is known that $d = vT$ (v is velocity of the target object), From Eq. (2.12), substituting Δd, we get

$$\omega = \frac{4\pi v\Delta t}{\lambda}. \tag{2.14}$$

Substituting Δt with T_c, which is the time between two consecutive sweeps and reordering, we get:

$$v = \frac{\lambda\omega}{4\pi T_c}. \tag{2.15}$$

Therefore, in summary, it can be said that the phase difference obtained from the Fourier transformation of two consecutive chirps, can help in determining the velocity of that object.

The phase shift between consecutive chirps is the same for any two chirps. This phase shift can be easily calculated by performing FFT on the Range FFT from each chirp. This is called Doppler FFT. The whole process is the called Range-Doppler FFT, i.e., first performing Range FFT on the signals to evaluate the Range and then performing another Doppler FFT on consecutive chirps to obtain the velocities of these objects and then to differentiate them. The Range Doppler FFT is done with the help of a matrix, with the dimensions N x M where M is the number of chirps and N is the number of samples in each chirp.

A detailed visualization of the Range-Doppler FFT process can be seen in Figure 2.6. Every IF signal/Baseband signal has its own phase. Every baseband signal is inserted row-wise in a matrix. Every row undergoes a Range FFT operation for Range estimation. This is done in parallel as soon as the FMCW received the RX signals. The result is a peak in some columns of the matrix. Once the number of received signals is enough, column-wise FFTs are performed for radial velocity estimation.

This results in peaks in different cells of the matrix. These cells correspond to different objects at different locations from FMCW and moving at a different radial velocity to/from FMCW. The Doppler-FFT changes the $Range \times Chirp$ index dimension to $Range \times Velocity$. Also, frequency is changed from the range from [0, 2π] to [-π,π] in the normalized frequency domain. This makes sure that even the negative velocities are captured.

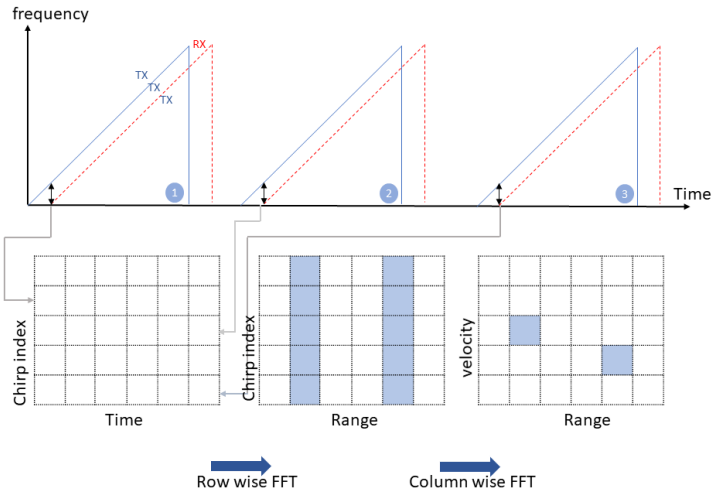


Figure 2.6: Range-doppler-FFT matrix created by taking Row wise FFT on Chirp Index× Time matrix followed by column matrix of the Chirp Index × Range matrix

Therefore, in summary, it can be said that the objects at different ranges or the objects at the same range but different velocities can be easily detected using Range-Doppler FFT.

Phase Evaluation for moving targets

In the previous subsection, the complete process of Range-Doppler FFT was explained. However, now one most case rises up, where the Range-Doppler FFT fails completely. Consider a case where two objects at the same radial distance are moving to/from the FMCW at the same radial velocity. The Range-Doppler FFT will give rise to one bin for two different objects. So this situation forces us to ask the question *"How can the two objects placed at the same radial distance and moving with the same radial velocity from FMCW can be separated?"*

A very simple answer is to give two eyes to the FMCW. The main advantage of humans having two eyes is to visualize the world in 3D, i.e., humans can perceive the depth of the objects in the field of view because eyes are located at different points. So using this analogy, if we use multiple receiving antennas for the FMCW, it will help us solving in the problem in question.

An FMCW measures Angle of Arrival (AoA) to resolve this problem. Angle estimation will require a minimum of 2 receiver antennas. Assuming

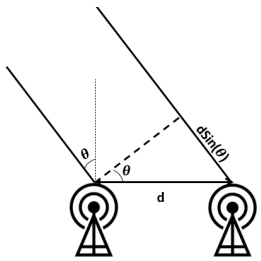


Figure 2.7: Angle of arrival in case of two receiver antennas

the receiver antennas are placed in close proximity, their ratio of the radial distance from one antenna with the other will be close to one. This means, even if the two antennas in consideration have some Range FFT (as they are almost at the same radial distance from the object), there will be a phase change involved. Eq. (2.12), shows that an object which made a small change in distance Δd in two consecutive sweeps gives rise to a phase difference $\Delta\theta$ or ω.

$$\omega = \frac{4\pi\Delta d}{\lambda}$$

Using the same mathematical notation, it can be said that the phase difference between the two antennas will be

$$\omega = \frac{2\pi\Delta d}{\lambda} \tag{2.16}$$

Here is the Eq. (2.16), the equation has a factor of 2 instead of 4 in Eq. (2.12). The difference can be understood by mere intuition. When an object moves by some distance Δd, the phase difference will be brought out by the two and fro signal. Here, in contrast, the phase difference is brought by the distance between the two receiving antennas. This is the reason the Eq. (2.16) has been divided by 2 [53]. Using the geometry of Fig. 2.7 and using Eq. (2.16), it can be derived that:

$$\omega = \frac{2\pi d\sin(\theta)}{\lambda}$$

Hence, θ which is the AoA can be derived as:

$$\theta = \sin^{-1}(\frac{\lambda\omega}{2\pi d})\ rads \tag{2.17}$$

It is worth mentioning that the maximum phase difference can not be more than π, For the phase difference more than π, it will be impossible to say if the phase difference was actually θ or $[\theta - \pi]$.

Now consider the Fig. 2.6, it is pretty much evident that the two objects at the same radial distance will result in same peaks in the Range-Doppler FFT, nonetheless, multiple antennas give rise to a phase difference.

If another FFT is applied to these phase differences, this gives rise to an FFT sequence which is called **angle-FFT**. Angle FFT helps in resolving such two objects.

2.2.4 RADAR Specifications

The RADAR used in this thesis has a carrier frequency of 24 GHz. This RADAR as seen in the Fig. 2.8 is developed by VTT. Table 2.1 provides the specifications of the RADAR.

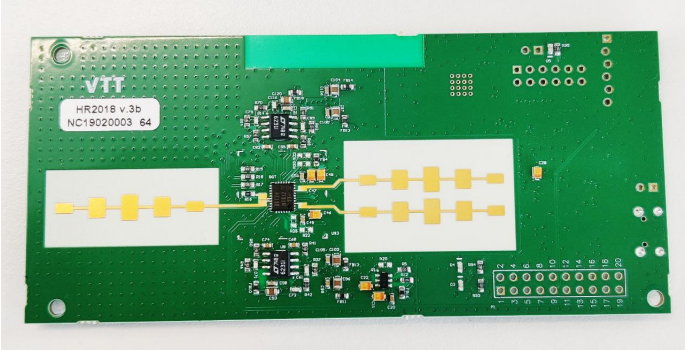


Figure 2.8: Internal details of the RADAR

S.No.	Specification	Value
1	Carrier Frequency	24 Ghz
2	Bandwidth of 1 sweep	250 MHz
3	Sweep Time	1 ms
4	Min sweep interval	16.7 ms
5	Centre Angle	210°
6	Max view distance	14 m

Table 2.1: Specifications of FMCW RADAR

2.3 Artificial Intelligence for Presence detection

Artificial Intelligence (AI) is intelligence demonstrated by machines, unlike the natural intelligence displayed by humans and animals, which involves consciousness and emotionality. There are numerous ways to define Artificial Intelligence (AI) as there are many controversial discussions about the definition of human intelligence. On a broader level, AI is an umbrella term for any hardware or software that enables a machine to mimic human intelligence. AI should be adaptive (learn from interaction) and, in many cases, autonomous as well [64].

2.3.1 Machine Learning

Machine Learning (ML) is a subset of AI. It is the study of computer algorithms that automatically improve its efficiency through experience and by the use of data [43]. From Tom M. Mitchell's definition of Machine Learning:

"**Machine Learning is a scientific field aiming to build a computer systems that automatically improve with experience. A machine learns with respect to a particular task T, performance metric P, and type of experience E, if the system reliably improves its performance P at task T, following experience E [44].**"

With the help of Mathematics, Statistics and Probability, the machines can learn from the data and then make a decision on the new unseen data. ML allows to automatically extract relevant information from the data. A Machine Learning model is then trained iteratively on a dataset with appropriately constructed features. The algorithm penalizes inaccuracies in the model, thereby forcing it to improve accuracy with every iteration.

There are several Machine Learning methods. Different methods can be tested to determine the best option for a certain problem. The ML algorithms are classified into Supervised, Unsupervised Learning, and Reinforcement Learning algorithms.

Supervised Learning

In Supervised Learning, the goal is to estimate the unknown model that maps known inputs to known outputs. The dataset, in that case, has defined targets or labels that are pre-determined. The problem can be Classification, Regression, or Probability estimation. Classification refers to the methodology of categorizing classes in the dataset by evaluating their relationships.

For example, a dataset that consists of images of different handwritten digits can be modeled to understand the differences. The discriminative leaned power of the machine learning model would be then evaluated on any unknown image [33]. Regression on the other hand is defined as a technique to quantify the relationship between dependent and independent variables [33]. A regression model is considered to be accurate if it can determine the true quantifiable value of the target variable given new inputs. An example of regression is housing price estimation based on measurements such as plot area, neighborhood, when was it constructed, etc. Numerous algorithms are used for Supervised Learning like Support Vector Machines [49], Decision Tree [46], K-Nearest Neighbors [20], Naive Bayes [45] etc.

Unsupervised Learning

On the contrary, In Unsupervised Learning, the goal is learning a more efficient representation of a set of unknown inputs. Here the expected targets are not provided to the algorithm. Therefore, the algorithm has to learn some patterns in the data without any supervision or prior knowledge of what the dataset might be about. An example is a model that differentiates between different customers of a supermarket based on their spending habits. It is not relevant to classify customers into categories as there are no pre existing differentiating criteria, rather the classification methodology is the objective of the training process. . The most common problems in Unsupervised Learning include Clustering and Compression. Numerous algorithms are used for Unsupervised Learning like K-Means [65], Self-organizing maps [32], Principal Component Analysis [30] etc.

Reinforcement Learning

Reinforcement Learning is an area of Machine Learning which is about deciding the best suitable action to maximize reward in a particular situation. It is used in many software and machines to find the best possible behavior or path it should take in a specific situation. Reinforcement Learning differs from Supervised Learning as in the latter the training data has the correct label with it so the model is trained with the correct answer itself whereas in the former, there is no answer but the reinforcement agent decides what to do to perform the given task. In the absence of a training dataset, it is bound to learn from its experience [62].

The main challenge with Machine Learning is that it is supposed to perform efficiently on new, unseen data. In order to verify of the Machine

Learning models performs well on the new data, usually, the dataset is divided into training and test set. The model is trained on training data. Nowhere in the training process, the test dataset is involved. A good model should give high performance on the test dataset along with the train dataset. This feature is called generalization. The generalization error i.e., test error must be minimized. The trained model's performance is tested with the test set to measure generalization error. If there is high performance on the train dataset and low performance on test dataset, the model is said to have overfitted the train set, or simply it has just learned the train set.

Based on the above discussion, Machine learning for Presence detection using RADAR in the context of lighting control is best described as a **Classification** problem. The target variable is the binary occupancy state of the RADAR, i.e, True and False which represents presence and absence respectively. The input in the case of RADAR is the discrete signals received by the antennas of the RADAR. The Machine Learning model needs to be trained with data from different scenarios and environments, so as to make one generalized model, i.e., Machine Learning algorithm needs to be trained for different scenarios to capture Experience E, given Performance Metric P which represents how well the model predicts occupancy for the given Classification Task T.

The motto of this study can now be summarized as the implementation and evaluation of a Presence detection modeling algorithm, which determines probabilities of human presence or absence, using FMCW RADAR in order to control lighting.

Chapter 3

Data Collection and Data Analysis

This chapter discusses the Data collection process, i.e., how the data is collected and how the data is labeled for the classification models. In addition, this chapter also discusses in detail about the feature engineering performed on the data along with exploratory analysis.

3.1 Experimental Design and Setup

The main and the foremost step for performing Machine Learning is data collection. The data must be collected and stored in a way that makes sense for the problem at hand. Therefore, a proper pipeline for the data gathering process was required. In this section, the complete data pipeline is discussed in detail.

3.1.1 Data Collection with RADAR

RADAR itself cannot be used for direct communication with a workstation. That is why Raspberry Pi 4 (Rpi) is used for communication. Rpi acts as a mediator between the RADAR and a workstation through which data is being processed or analyzed. The RADAR is connected to Rpi internally and packed inside a 3D printed case, thus forming one module. Please be aware in further discussions, we will be referring to this module i.e., combination of RADAR and Rpi as RADAR module or just RADAR for the purpose of simplicity. When supplied power, the Rpi inside the module opens a hotspot for a secure connection. The RADAR module is then connected to the workstation using SSH.

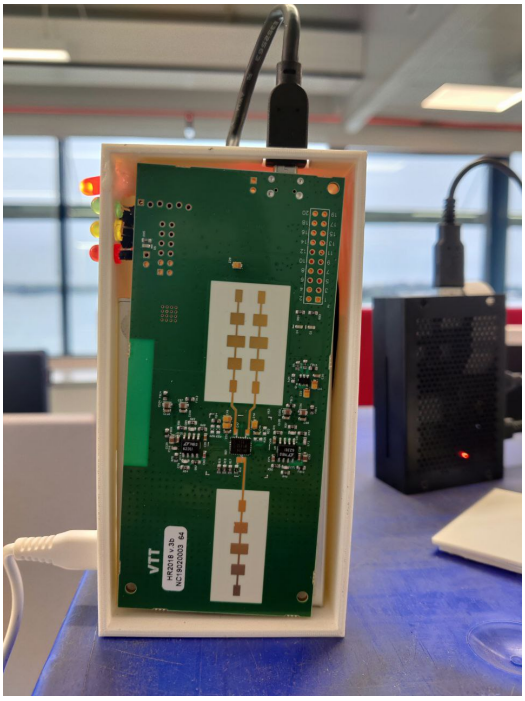


Figure 3.1: The RADAR module - RADAR + Rpi

The program inside the Rpi to read data from RADAR is executed via this SSH. In order to send data from Rpi to the workstation, MQTT is used. The Message Queuing Telemetry Transport (MQTT) is a lightweight, publish-subscribe network protocol that transports messages between devices. It is designed for connections with remote locations where a "small code footprint" is required or the network bandwidth is limited [40]. The hierarchy includes one broker and multiple clients. The broker receives all the messages from all the clients and then forwards them to appropriate destinations. Information is organized in the form of topics. When a client publishes information on Topic A to the broker, the broker sends this information to all the clients who have subscribed to this topic.

RADAR module publishes information to a broker, and then the broker sends the information to the workstation. The information received by the workstation is the actual received signals of the received antenna of the RADAR. Instead of using a local workstation for storing the data, AWS S3 buckets are used for storing the data. AWS S3 is an efficient method of storing high volumes of data that can be easily accessed by people having authorization.

To dump the data to AWS, an efficient packaging technique for data was

required. One Antenna of the RADAR generates 497 samples, i.e., 2x497 samples per chirp for two receiver antennas. In the basic raw mode, the time difference between two consecutive chirps is 31.25 ms. This means that the RADAR generates 2x497 samples in 31.25 ms or 32x2x497 samples in 1 sec. One way is to treat the signals of 1 chirp as one datum i.e., one data sample with 2x497 feature vectors, however, there is a problem with this approach. For instance, time associated with this data is very less and features per chirp are not sufficient enough for our deep learning model to make an efficient prediction. Therefore, multiple chirps are stacked together to form one datum. In this thesis, **32 chirps are stacked together to obtain one datum which represents 1 sec of signals**. This approach will even make the labeling much easier which is explained in a more elaborate manner in the next section.

So in summary, the data produced every second is stacked together in the file format and then dumped into AWS S3 buckets. It is worth mentioning that the name of the file is the timestamp at which the first chirp was transmitted. This helps in storing the data properly and it makes it really simple to compare the data with an RGB camera, which will be explained in the next section.

3.1.2 Ground Truth/labeling of the Data

The RADAR is set to receive signals for the time span of 31.25 ms. This means in one second, approximately 32 signals are generated. The RADAR data is collected for multiple hours in a meeting room of Helvar's headquarters in Espoo. This means on average, the RADAR generated 32x60x60 samples in 1 hour. In an ever-moving environment like IT offices, it is almost impossible to annotate such a vast amount of data manually. It's not possible to label if a person was present or not for every second for the RADAR data manually. One approach was to label the data using the meeting calendars i.e., label the data True if the meeting room is booked. But this approach is highly prone to false labeling. For example, a meeting room can be occupied by a person even if it's not booked. On the other hand, it may be possible that no one comes to the meeting room despite the booking. Therefore, a more reliable labeling mechanism was desired.

The other way to do this was to record a video of the same at the same time as the RADAR and then using this video to annotate the data manually. This approach is again prone to a lot of loopholes, for example, a manual annotation for days and days of data requires a lot of extra effort. Apart from effort, this manual annotation method is really not scalable when we are dealing with commercial usage. It works fine for research-based, but in

Figure 3.2: The RGB Camera

an industry when eventually clients are involved, there is a need for exploring a better automatic solution. [1]. The data which is used in this thesis is hours and hours of data. In order to achieve reliable inference from our data, precise labeling is required for the training set. Therefore, for creating the labels for the RADAR data for the training set, an RGB camera is used. Raspberry Pi 8.0 Mpix v2 camera with Sony IMX219 8-megapixel sensor is used for this thesis. The Camera Module can be used to take stills photographs as well as high-definition video. It supports 1080p30, 720p60, and VGA90 video modes, as well as still capture. The module is attached via a 15cm ribbon cable to the CSI port on the Rpi. However, instead of manually annotating these videos, the use cases of Machine Learning and Deep Learning are leveraged. With the advancement of Machine Learning and Deep Learning libraries, object detection and object recognition can be done with accuracy as high as 99%. This field of Artificial Intelligence that trains computers to interpret and understand the visual world is called Computer Vision. With the advancement of AI and innovations in Deep Learning and Neural Networks, Computer Vision had been able to take great leaps currently and has been able to surpass human intelligence in some tasks related to detecting and labeling objects for optical sensors.

3.1.3 Data Collection with RGB Camera

The RGB camera in the experiment and RADAR are mounted next to each other so that the field of view of RADAR and camera overlaps. The camera

Figure 3.3: Setup

used in this thesis captures 60 frames per second (fps). To label the RADAR data, the scene is captured with an RGB camera alongside RADAR. The camera is connected to a 32-bit Rpi. This Rpi leverages the camera to record videos. Each video file spans 1 second. The name of the file is the timestamp at which the file was created. As soon as the file is created, a parallel process dumps the data periodically to the AWS bucket. In order to prevent Rpi from memory overload, the files from the Rpi are removed as soon as they are dumped to AWS successfully. The important thing to notice here is the duration of the video. 1 sec is chosen so that it is easy later on to combine RADAR data as well RGB data. Special attention has been paid to clean the data before sending it to S3 buckets. For an instance, the videos of size 0 Kb are removed as these are the lost packets or the unsuccessful video captures.

3.2 Data Processing

In the previous section, we discussed about the data pipeline to collect the data. We collected two types of data, i.e., the data from the RADAR and the data from the RGB camera. In this section, the collected data will be

processed, i.e., the collected data will be transformed and feature engineered to make it suitable for building up the models.

3.2.1 Platform for Data Processing

For this thesis, the Data collected is huge. It is almost impossible for a local machine to run heavy computations on such a high volume of data. Therefore in order to run heavy computations without any memory leaks, the best possible technology available is the use of Cloud Resources. For the purpose of this thesis, we are using AWS EC2 instances. AWS EC2 instances are fast reliable and will be able to access the S3 buckets directly.

There are different parameters to be taken into account while deciding to use CPUs or GPUs to train the deep learning models. Those parameters include the memory bandwidth, cost effectiveness, and parallelism. GPU has on average better memory bandwidth in comparison to CPUs. Considering the dataset size, the larger the dataset the more advantage the GPU will have over CPU. As for the Parallelism of the model, some models have more parallelism than others. For example, Fast Forward Neural Networks (FNNs) and Convolutional Neural Networks (CNNs) can support high parallelism, therefore, it can be applied better on a Single Instruction Multiple Data (SIMD) processor such as the GPUs. Finally, coming to Cost-effectiveness, the power cost of GPU is higher than the CPU. For non-parallel processes or less parallel processes like Recurrent Neural Networks, it is better to use a CPU. In this work, the cloud instances based on CPU or GPU are used to train the models according to the needs mentioned above. For this thesis, the Deep Learning Base AMI Ubuntu instance of type p2.xlarge is used. The Deep learning instance is launched in order to use the GPU instances for running the Deep Learning models.

3.2.2 Processing of RGB Data for Label Generation

The data collected from RGB data which is stored in S3 buckets can be directly accessed from the EC2 instance as both the instance and the bucket are present in the same AWS region. For this project, a hybrid of two techniques is used for object detection on the RGB data collected.

1. **YOLOv3** - YOLO (You Only Look Once) is one of the most effective object detection algorithms. It is a computer vision algorithm that encompasses most of the innovative ideas present in the research. YOLO is popular because it attains high accuracy while also being able to run in real-time. The algorithm focuses on looking only once i.e., only one

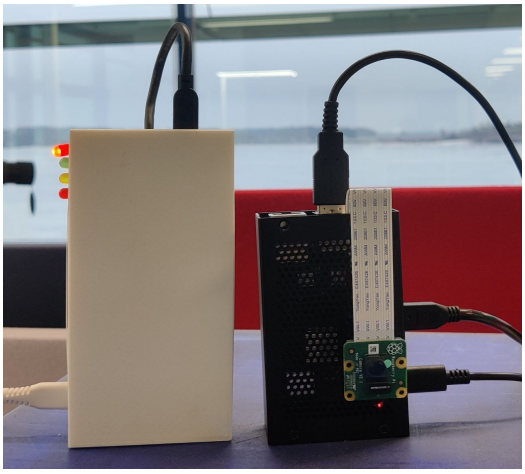


Figure 3.4: RADAR and Camera

forward pass through the neural network to make predictions. With YOLO, a single CNN predicts multiple bounding boxes and their corresponding class probabilities. YOLO trains on full images and directly optimizes detection performance [51]. Fig. 3.5 demonstrates the comparison of the performances of different object detection algorithms trained on COCO dataset [55].

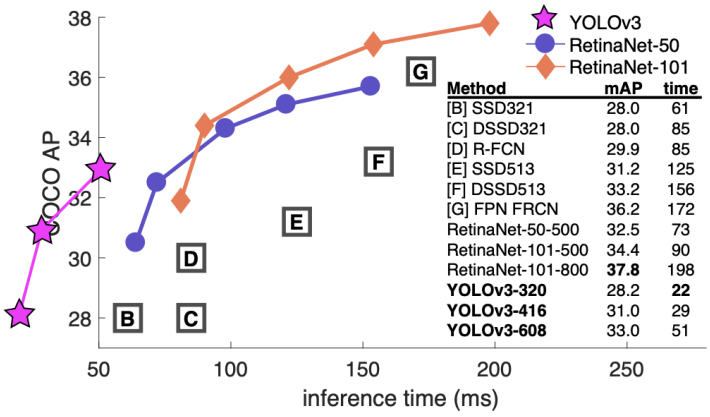


Figure 3.5: Comparison of YOLOv3 with other algorithms [55]

YOLOv3 [54]is chosen because it is extremely fast and accurate. The mAP of YOLOv3 is measured at .5 IOU which is on par with Focal Loss but about 4x faster. YOLOv3 uses a few tricks to improve training

and increase performance, including multi-scale predictions, a better backbone classifier, and more [54]. YOLOv3 returns output in the form of pre-defined 80 classes on which the model is trained. Here in this thesis, only person class is a matter of interest. In order to detect only humans, the detection class of the YOLOv3 model has been manipulated.

2. **Motion Detection** - This technique is used to detect motion in a scene. It is used because sometimes it is possible for a person to be occluded in the video. For an instance, consider a person half occluded behind the desktop monitor, or a scenario where there is only hand moving in front of the camera, or where someone is half visible behind the door, for such scenarios YOLOv3 will fail because there are a lot of occlusions present in the same. Therefore for such cases, a motion detection algorithm is used on the RGB camera data. When the movement is beyond a fixed threshold, only then the algorithm returns presence, otherwise, it ignores small movements, for example, the movement due to curtains or leaves of the plants is ignored. The approach is very simple and self intuitive. When the program starts, it will capture a called baseline image. The program will keep comparing the new frame with this baseline image. If there is a movement in the new frame, the contents of the image will be different and if this difference is beyond a certain threshold, the program will return Presence.

Once the data is collected into AWS S3, YOLOv3 and Motion Detector are run on the RGB data collected. Every second of collected data contains 60 frames, so for every second YOLOv3 and Motion Detector generate 60 labels. In the Lighting industry, the requirement for getting the true labels is within few seconds. The accuracy of milliseconds is not required in the lighting industry. Therefore, for the sake of simplicity and better results, the mode value of the 60 labels was taken for every second.

After using this pipeline, the labels corresponding to every second of RADAR data are ready. It is worth noticing that there might be some inaccuracies in the labeling because of the difference in the angle of view.

3.2.3 Processing of RADAR Data and Feature Engineering

In the previous section, we talked about how the pipelines are arranged for the proper flow of data and how the labels are generated using RGB data. In this

section, we will talk about how the data is arranged for the machine learning model to work efficiently. This section will also explain the operations and transformations done on the RADAR data. In addition to that, we will see exploratory data analysis done on the radar data.

The data from FMCW RADAR is generated in the form of chirps/sweeps. According to the hardware configuration of the RADAR used in this thesis, it generated 497 samples in one chirp. Since the RADAR have two receiver antennas, it generates 497 per antenna in 1 chirp. RADAR generates 1 chirp is approximately 31.25 ms, although the time may vary in fractions of milliseconds. This means the RADAR generates approximately 32 samples in 1 second. On the other hand, the RGB camera operates at 60 fps.

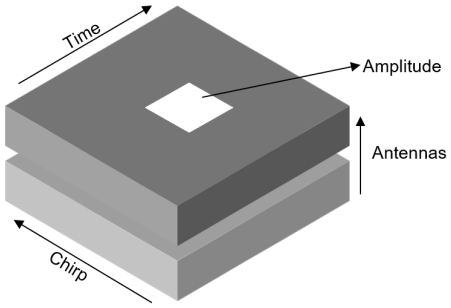


Figure 3.6: The data signature for RADAR

In order to overlap the data and make effective labeling, the data from 32 chirps of RADAR is combined together to make a combined signal of 1 second. Alternately, it can be imagined that one instance of data for our Machine Learning/Deep Learning model corresponds to 1 sec of data, i.e., 32 sweeps from the RADAR and 60 frames of the camera are mapped together. Therefore the data signature of the RADAR of 1 second can be imagined as a 3D matrix where the first dimension corresponds to antenna number, the second dimension corresponds to chirp number and the third dimension corresponds to Time. The value of the cell is the actual amplitude of the signal. Timestamping is used for the mapping of RADAR and RGB camera data. These 1-second videos of RGB camera when fed to YOLOv3 and Motion detector model, as explained in the previous subsection 3.2.2, generates output in the form of True/False.

In Fig. 3.7, random instances of RADAR data are sampled. Every plot corresponds to 1 second of data. The data from 32 chirps is flattened to make it a Time vs Amplitude plot. Since each chirp has 497 samples, therefore the

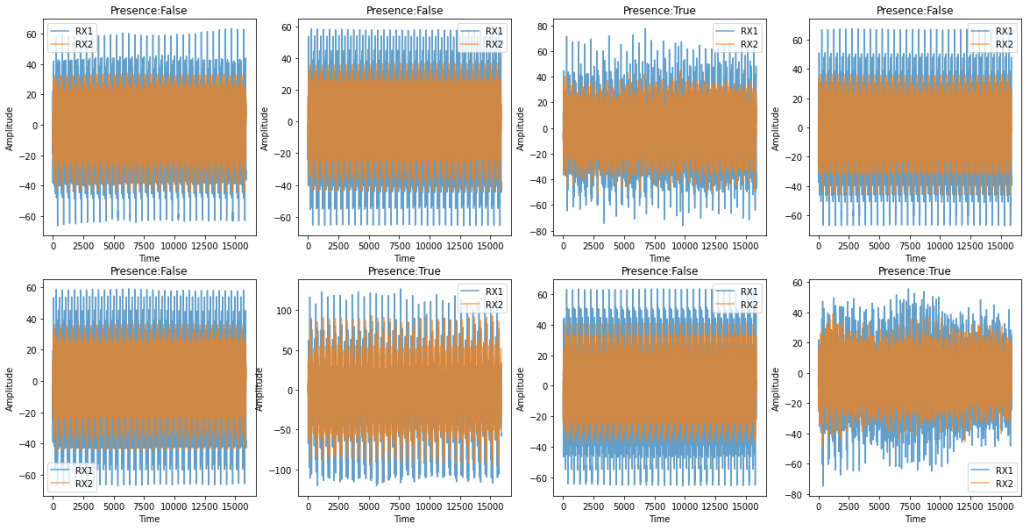


Figure 3.7: Random sampling of 1 second of RADAR data

Time axis has 32×497 samples on the x-axis. The y-axis corresponds to the Amplitude of the received signal. The title of the subplots presents if there was presence or absence in the RADAR view. False corresponds to absence while True corresponds to Presence. Clearly, there is a significant difference between presence and absence. The signals from the absence scenario are similar across time, or we can say that time does not change the signal. However it is completely different in the presence scenario, the signal tends to change a lot over time.

Deep learning models were tried directly using this raw data, but the model prediction was not highly accurate. One reason could be a significant difference in amplitude in different scenarios. The amplitude differs a lot based on the range, and this range of objects varies from scenario to scenario. Therefore the raw data can not be used directly for ML models. The second reason for not using this raw data directly for the ML model is the maximum range capacity. The RADAR data in its raw form can detect objects up to 250 meters. But in an office environment, we really do not need data beyond some threshold. The data beyond this threshold will be just noise. The third reason is, the data in raw form has a lot of noise due to cluttering. The SNR for raw data is heavily compromised in raw form. Therefore, some transformations were required on this data for the ML model to work properly.

As explained in subsection 2.2.3, the data was converted from $Time \times Amplitude$ to $Range \times Amplitude$ using Fourier Transformation. As ex-

plained earlier, Fourier Transformation converts the Time into Frequency. Since in FMCW RADAR frequency directly estimates the Range, that is why Time gets converted to Range. This transformation is called **Range FFT**. Here FFT means Fast Fourier Transformation, which is a faster method of computing Fourier Transformations of a discrete signal.

A very common problem associated with FFT is **Spectral Leakage**. An estimation of a signal computed using FFT or DFT does not contain spikes and zeros like shown in Fig. A.2, instead it will have a dominant peak which is then smudged over several consecutive bins. It happens because of a finite window of data as the data is never infinitely large. In order to cater to this problem, windowing functions are used to reduce spectral leakage. In this thesis **Blackman Window** is used. It is designed to have close to the minimum leakage possible. The Blackman window is defined as,

$$w(n) = 0.42 - 0.5\cos(2\pi n/M) + 0.08\cos(4\pi n/M)$$

where M is the number of samples and n is the n^{th} sample. It is also known as an apodization which means "removing the foot", i.e., smoothing discontinuities at the beginning and end of the sampled signal or tapering function [12] [52].

Once the Blackman windowing is applied to raw data, the FFT of the data is computed. The result of the FFT is then normalized to make the Euclidean norm of the signal equal to 1. Range FFT returns 512 samples (closest to 497), out of which half the samples are mirror images. But for the sake of putting an upper limit on the range i.e., to what range we want to detect objects, we have put an upper limit of 32 samples. This number 32 has been chosen after a considerable number of experiments. These 32 samples correspond roughly to 18.7 meters.

Fig. 3.8 represents a 3 dimensional view of Range FFT data. A comparison of the Range FFT signal for human presence and absence is done in the figure. It is worth mentioning that the plot indicates the Range FFT output of only one antenna. A similar signal is obtained from another antenna as well. It is very evident from the plots that in the scenario when there was no human, the signal remains constant over chirps. However, when there was human movement involved, the signal varies over chips. Every plot corresponds to 1 second of data. It is worth noticing that the y-axis which corresponds to the chirp index ranges from 0 to 32, and the Range also ranges from 0 to 32 on the x-axis.

To visualize this scenario even better, a 4-dimensional plot is drawn involving 2 antennas in Fig. 3.9. The color in the plots indicates the amplitude intensity. Yellow color corresponds to higher amplitude values and blue

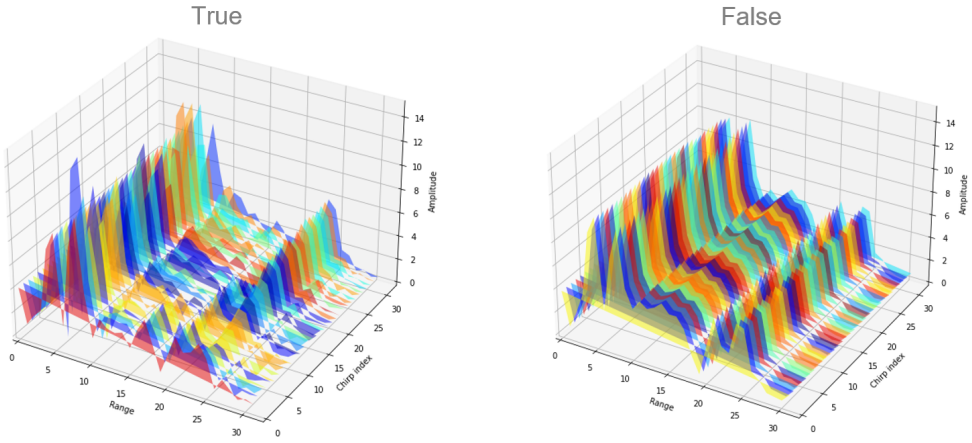


Figure 3.8: Range FFT comparison for human presence and absence with one receiver antenna

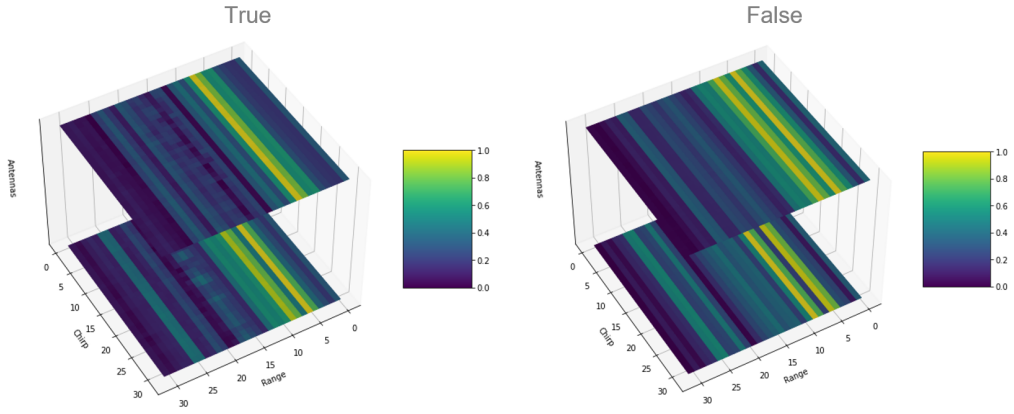


Figure 3.9: Range FFT comparison for human presence and absence with both receiver antenna

corresponds to lower amplitude values. It is evident in the plots that the amplitude values of the two antennas are always comparable in quantification. It can be seen in this plot as well that in the scenario of human absence, the amplitude of the signal remains constant across chirps. Comparatively in the human presence scenario, a distinctive smudge can be observed across chirps. This also comes by intuition that when the human movement is involved, a human body, which is a collection of points, is at a different distance from

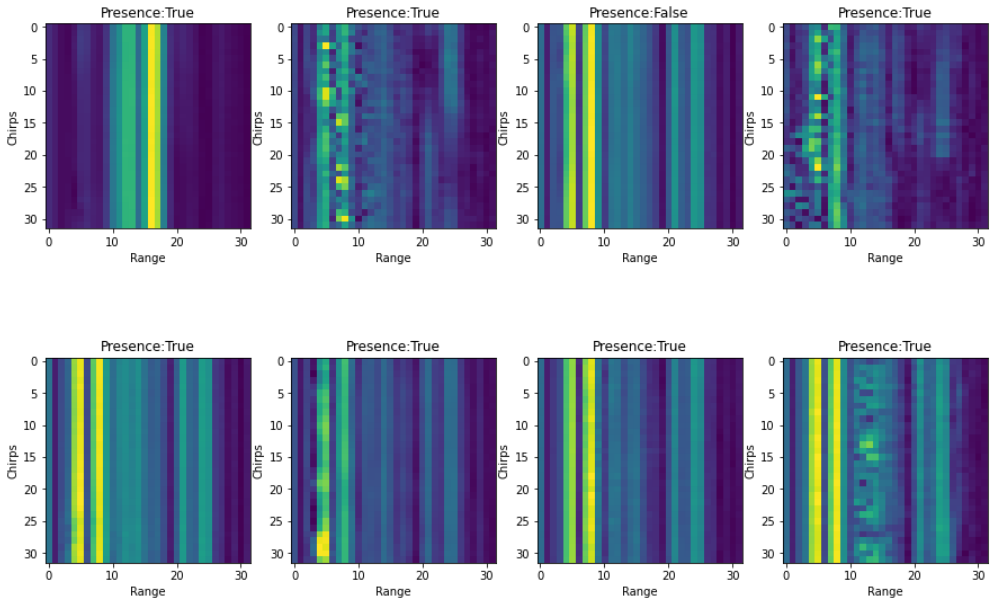


Figure 3.10: Range FFT on randomly sampled RADAR data

the RADAR. It is important to note that the distance from one antenna is of course different than the distance from another antenna, but the difference is diminutive as the antennas are placed in close proximity.

In Fig. 3.10, random instances of the Range FFT are sampled to display how data from Range FFT looks like. The subplots indicate the presence and absence scenario. The plot indicates results from only one antenna as the interpretation is much easier for one antenna. The other antenna gets the same results. The important thing to notice here is that these plots are obtained from different locations, i.e., from meeting rooms, from open offices, and from desk premises. This fact is evident from the plot itself. For example, the highest intensity does not remain in the same range. For some scenarios, the highest intensity bins are generated by objects closer to the RADAR, while in some scenarios, the highest intensity bins are generated by objects far from RADAR. Additionally, we can observe the smudge in the Range FFT which we discussed above can be seen clearly. The smudge can be seen in all the human presence scenarios irrespective of where the data was collected.

To analyze it even further, the bar plot is drawn by taking the average value per chirp which can be seen in Fig. 3.11. Every chirp will give one mean amplitude value, giving rise to 32 mean amplitude values corresponding to

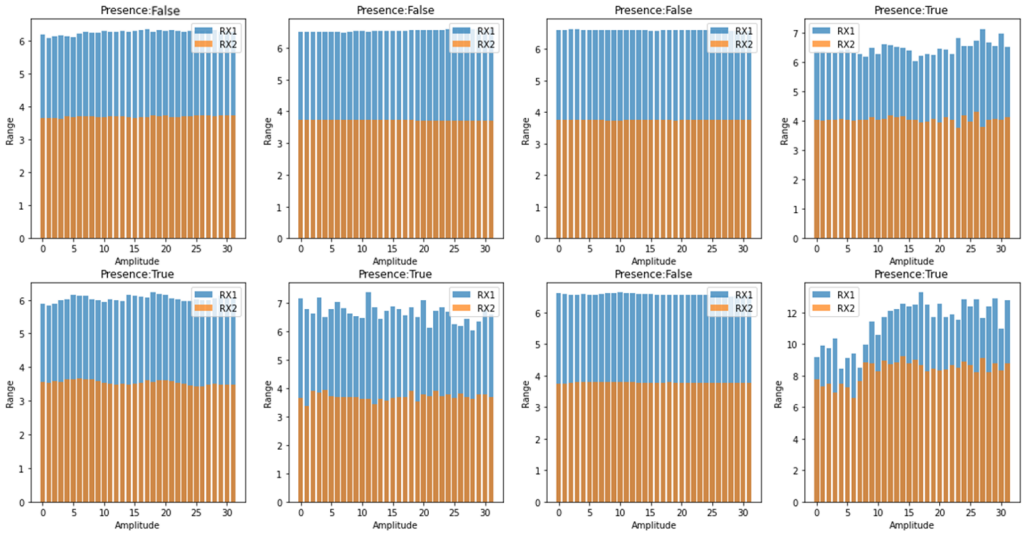


Figure 3.11: Amplitude vs Range Bar Chart for both the antennas where Amplitude is the mean value within 1 chirp

32 chirps. This is done for both the RX antennas. The objective behind this plot is to analyze if the mean value stays constant over chirps in the absence scenario and varies over chirps in the presence scenario. The result can be interpreted clearly in the absence scenario (indicated by False), where the mean amplitude value remains constant for both the antennas. This result is very intuitive as there is no movement, so all the chirps get the same signal.

Deep learning models were tried on Range FFT data. However, the model prediction was not highly accurate. One reason could be the significant difference in amplitude in different scenarios. The amplitude differs a lot based on the range, and the range of objects varies from scenario to scenario. Therefore the Range FFT can not be used directly for ML models. Another reason is the scenario when there is little movement across the chirp. The gradient is so less that the Machine Learning model assumes small movement as background noise. Therefore, transformation is required to capture these small movements and thus increasing SNR.

As explained in subsection 2.2.3, the Range FFT was further converted from $Range \times Amplitude$ to $Range \times Velocity$ using another Fourier Transformation. This transformation is called **Range-Doppler FFT** since Doppler FFT is applied on Range FFT. As explained earlier, Fourier Transformation converts the Chirp index into Velocity. It is worth noting that in the case of the absence scenario, all the values in the Range Velocity matrix will be zero or near zero as there is no movement in the scene. Fig.3.12 represents a 3 di-

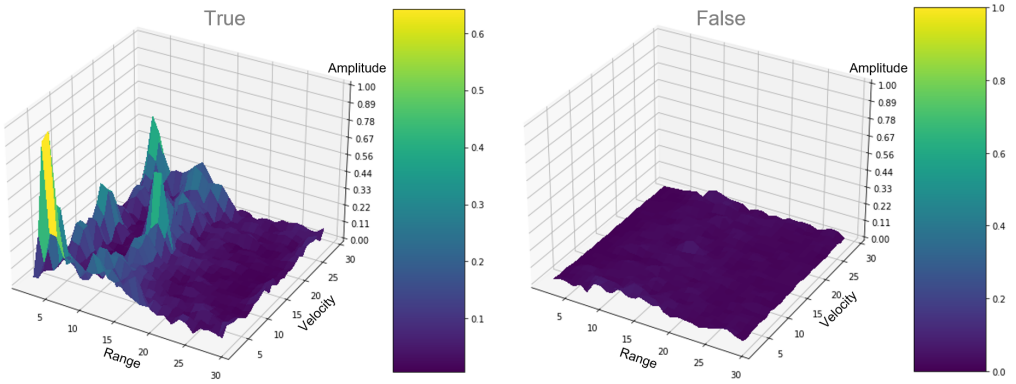


Figure 3.12: Range-Doppler comparison for human presence and absence with one receiver antenna

mensional view of Range FFT data. A comparison of the Range FFT signal for human presence and absence is done in the figure. It is worth mentioning that the plot indicates the Range-Doppler FFT output of only one antenna. A similar signal is obtained from another antenna as well. It is very evident from the plots that in the scenario when there was no human, the amplitude is near zero. However, when there was human movement involved, the signal varies over range velocity plot. Every plot corresponds to 1 second of data. The Range FFT plot returned 32×32 matrix, therefore by applying another Fourier transformation also returned 32×32 matrix. Nonetheless, the first bin of the Fourier transformation always returned a very high peak because of the digital component present in the transformation. It is of course not a concern here, therefore in this thesis, we have just ignored the first and the last bin of the Range-Doppler FFT. The peaks are highly prominent in the presence scenario and correspond directly to a moving object.

To visualize the Fig. 3.12 in more detail and from the perspective of two antennas, a 4 dimensional plot is drawn involving two antennas in Fig. 3.13. The color in the plots indicates the amplitude intensity. Yellow color corresponds to higher amplitude values and blue corresponds to lower amplitude values. The Range-Doppler matrices of both antennas are always comparable in quantification. It can be seen in this plot as well that in the scenario of human absence, the amplitude of the signal remains near to 0 throughout the time window. Comparatively in the human presence scenario, distinctive high intensity peaks corresponding to velocity can be observed in the signal. There are minute differences in the Range-Doppler of both antennas. This

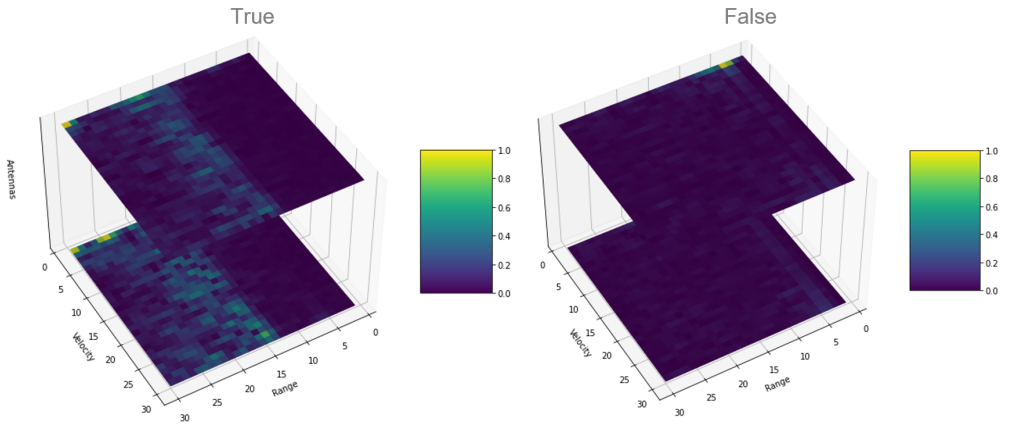


Figure 3.13: Range-Doppler comparison for human presence and absence with both the receiver antenna

is because of the placement of two antennas in two different positions.

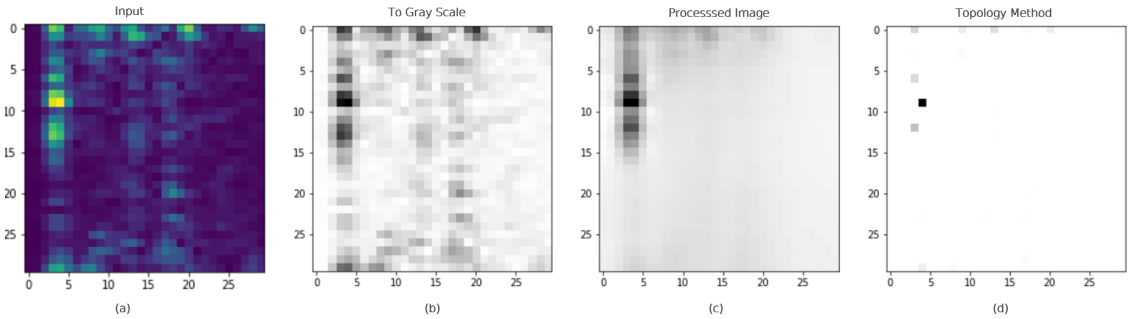


Figure 3.14: Range-Doppler signal with series of cleaning steps for noise reduction

One might ask "If the Range-Doppler FFT plot gives the Range vs Velocity values, then why are there so many high intensity regions?" The answer is Spectral Leakage again i.e., in a finite discrete signal, the Fourier transformation does not return a peak and zeros, instead, it returns a peak smudged to its sides. For an instance, consider the subplot (a) in Fig. 3.14. This is a Range-Doppler FFT of the scenario when there was only one person in the room. However, due to spectral leakage instead of showing only one high intensity region, the plot shows high intensity even in the neighbouring regions. To cross-check if the Range-Doppler FFT obtained was in fact

rightful, a peak detection mechanism is undertaken on the data. Peaks indicate significant events such as a sudden increase in the signal. A peak is detected when a threshold is exceeded [24]. To make sure that peaks are detected across global and local heights, and in noisy data, various denoising and pre-processing methods are utilized. The subplot (b) is the same plot as (a), but the color is set to gray for easier observation. In subplot (c), the two-dimensional signal is denoised. A special technique called 'fastnl' is used in this plot. NL here stands for Non-Local. In this method, the color of the cell is replaced by the average of the color of the similar cells. But the most similar cells to given cells have no reason to be close at all [15].

Once the data is denoised, it is then fed to a peak detection algorithm. In subplot (d), the Topological data analysis (TDA) method is applied to the RADAR signal. TDA is based upon the observation that data often possesses a certain intrinsic shape such as the shape of a point cloud, the shape of a geometric object, or in this case the shape of a signal. Persistent homology is probably the most prominent tool in TDA that gives us the means to describe the topological properties of these shapes [21]. After applying TDA to the de-noised sample, it is very much evident in the plot that there was only one person present in the room as there is only one high-intensity blob in the processed signal.

The whole process to denoise algorithm has been carried out only for the sake of visualization. However, the Deep Learning model is only fed with noised data. The reason is simple, we want our DL model to learn the denoising technique by itself. If we de-noise the data and then feed it to the NN model, it completely contradicts the foundation of Deep Learning i.e., there is no need for explicit feature extraction. Another way to think of it is, the de-noised data matrix is highly skewed, it has very few high intensity values as compared to zero. The ML/DL model would not work at all, rather it will simply learn the whole dataset. This is not what is desired in the thesis. We want our model to clean the data by itself.

3.3 Summary of Training, Validation and Test dataset

In this thesis for making the model generalized, the train, validation and test set have been collected independently. Usually, in ML modelling only test and train dataset are collected, and validation dataset is obtained by splitting the train set. In this thesis work, validation set has been collected completely independently because while collecting the RADAR data, every

data item (which is obtained by combining 1 sec of data) is not different. Take an example of a scenario, where there is no one in the room, the scene is completely empty. Now, for some duration t, till there is no change in the scenario, all the data items obtained in this duration t is extremely similar. If we use split function on the train set to obtain test and validation set, it is extremely likely that some of the data items generated in this duration become a part of the test set and some become a part of the validation set. Intuitively, it means if we split the train set into test and validation, we will be training on some train set and validation the model on an almost similar set. Therefore, in case of overfitting on the training set, the validation set will never be able to justify if there is overfitting. Instead, it will always support overfitting.

During the data collection part, special attention was paid to create balance in the data set, i.e., the data corresponding to the human absence should be quantitatively similar to the data corresponding to the human presence. Moreover, the dataset was collected from different locations to make the model generalized, for example, open office, cafeteria, meeting rooms, telephone booth, guest room, etc. For test, validation and train set, it was made sure that the location of the data collection does not overlap. Even if the location is the same, the angle and the exact position at which the RADAR is placed is always different.

Training dataset overall consists of approx. **8 hours** of RADAR and the RGB data. Table 3.1 demonstrates the distribution between human presence and human absence class in the train dataset. Training data is captured primarily from 3 different types of locations, i.e., the open office, the seating area premise, and the meeting rooms. Even though the location was same, the angle and the position of the RADAR was always varying after sometime.

Validation dataset consists of approx. **3 hours** of RADAR and the RGB data. Table 3.1 demonstrates the distribution between human presence and human absence class in the Validation dataset. Training data is captured primarily from 3 different types of locations, i.e., the telephone booth, parking lot, and the cafeteria. Even in this case, the angle and the position of the RADAR was always varying after sometime.

Test dataset consists of approx. **3 hours** of RADAR and the RGB data. Table 3.1 demonstrates the distribution between human presence and human absence class in the Test dataset. Training data is captured primarily from 3 different types of locations, i.e., different meeting room, Guest room, and research lab. Even in this case, the angle and the position of the RADAR was always varying after sometime.

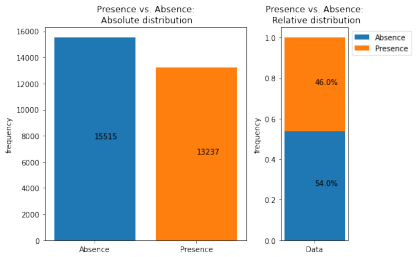
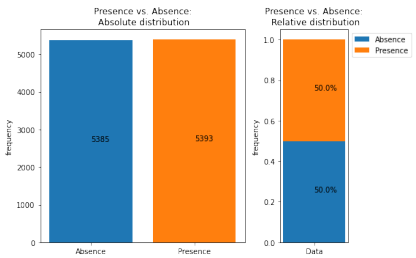
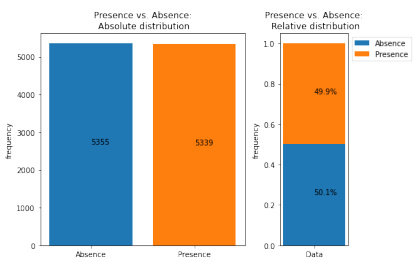
Train Set				
Class	Location	#samples	Time	Distribution
Presence	Open Office	4215	01:10:15	
	Desk Area	4567	01:16:07	
	Meeting Room	4455	01:14:15	
	Total Presence	13237	03:40:37	
Absence	Open Office	5026	01:23:46	
	Desk Area	5298	01:28:18	
	Meeting Room	5191	01:26:31	
	Total Absence	15515	04:18:35	
	Total	28752	08:00:00	
Validation Set				
Class	Location	#samples	Time	Distribution
Presence	Telephone Booth	1570	00:26:10	
	Parking	1836	00:30:36	
	Cafeteria	1987	00:33:07	
	Total Presence	5393	01:29:53	
Absence	Telephone Booth	1475	00:24:35	
	Parking	1932	00:32:12	
	Cafeteria	1978	00:32:58	
	Total Absence	5385	01:29:45	
	Total	10778	03:00:00	
Test Set				
Class	Location	#samples	Time	Distribution
Presence	Research Lab	1953	00:32:33	
	Meeting Room	1742	00:29:02	
	Guest room	1644	00:27:24	
	Total Presence	5339	01:28:59	
Absence	Research Lab	1667	00:27:47	
	Meeting Room	1826	00:30:26	
	Guest room	1862	00:31:02	
	Total Absence	5355	01:29:15	
	Total	10694	02:58:14	

Table 3.1: Dataset summary

Chapter 4

Methodology

This thesis will be focused mainly on the Classification problem of the "human presence" i.e., if the human is present or not. This will be a Supervised Learning problem where the features will be mapped to the classes/labels. For this thesis, we are using VTT built 25 GHz RADAR. The process includes data collection, data cleaning, data analysis, model building, and model deployment. Here is a flow of the process executed-

1. RADAR module provides an Analog to Digital converted signal. The data from these signals are collected over multiple hours at multiple locations.

2. The data obtained will be feature engineered using multiple techniques.

3. The idea is to engineer the received RADAR signal of every time frame (say 1 sec) to a 2-D map and then treating this 2-D map as an image or set of features.

4. Deep Neural Networks or other Machine Learning techniques will be applied on this already processed data.

5. For the purpose of training, we will leverage the cloud resources by AWS.

6. The last step is to deploy these models on the edge devices.

The previous sections gave insights on suitable representations of the RADAR output. Once the RADAR data has been converted to the final input feature matrix, the next step is to build a Machine Learning model that takes as input a feature matrix and target labels, and learns to predict an unknown RADAR signal.

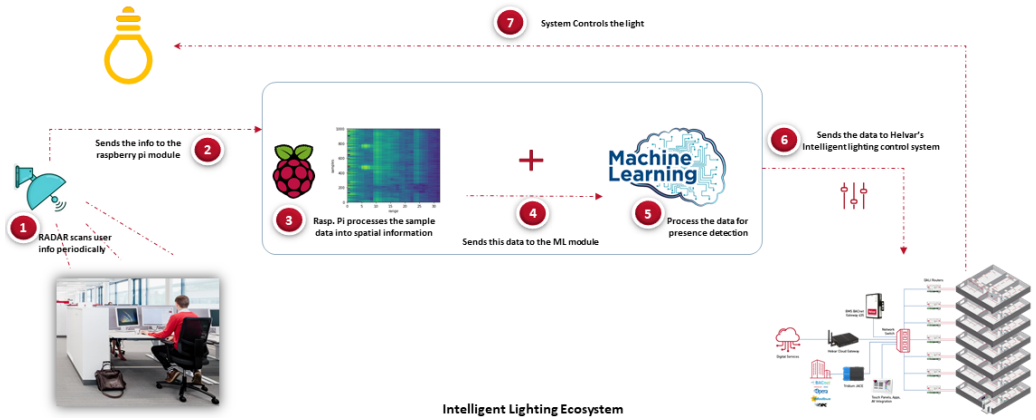


Figure 4.1: The complete flow of the process, i.e., from RADAR scanning to data modelling to model building and finally hardware control.

Deep Learning

Machine Learning tasks can be solved by designing the right set of features to extract. One can use the domain knowledge to design features that are robust. While the domain knowledge works perfectly fine for most the cases like the cases where features can be numerically or quantitatively explained, nevertheless, the problem arises for the situations like images, videos, or text. For such tasks, it is difficult to know what features should be extracted. It may be possible to extract the features in an image context with the algorithms that consider intensity and change in intensity with respect to pixels. Most of the time, these features are not perfect and eventually force the researcher to reiterate the feature extraction phase.

These problems can be solved with Representation Learning, i.e., we use Machine Learning not only to predict the mapping of the representation but also the representation itself. Deep Learning is a subset of Machine Learning based on Artificial Neural Networks which learns the representations. Deep Learning solves the central problem in Representation Learning by introducing representations that are expressed in terms of other, simpler representations. Deep Learning enables the computer to build complex concepts out of simpler concepts [26].

Deep Neural Networks is a Supervised Learning approach of a deep network of processors called neurons. A neuron originally is a biological term for a fundamental unit of the brain. The neuron is designed to pass electrochemical signals to the other neurons and to other parts of the brain [26]. The architecture of the Artificial Neural Network is inspired by the brain's

Neural Networks. The Neural Network processors receive and transmit signals between each other. The neurons store and process information in the connections between the neurons which are called weights. The weights are multipliers on the inputs of the neuron.

Deep Neural Networks are composed of layers. Each layer further is composed of several neurons. The input layer or the first layer of the network get their inputs directly from the data. For example, in the case of an image, the input layer consists of all the pixels values of the image. The rest of the layers of the network i.e., hidden layers use the previous layer's outputs as their input and feed their produced output to the proceeding layer. The final layer is called the output layer which is used for delivering the output of the whole Neural Network.

The depth of the Deep Learning system enables the computer to learn a hierarchical program. For example, when dealing with images, it learns the feature hierarchy all the way from pixels to the classifier. Each layer has some actions that are executed in parallel, however, actions in different layers are executed sequentially in the order of the layers [26].

4.1 Feedforward Neural Networks

Feedforward Neural Networks (FNNs) often called Multi-Layered Perceptrons (MLP) or Deep Feedforward networks are the typical form of Deep Learning models. The models are called Feedforward as the information flow is only in the forward direction, i.e., information flows through the function which is evaluated on X (input vector), then some intermediate computations which define f and finally to the output y. In other words, there is no feedback connection from the output that is fed in the model [26].

FNN's learns the characteristics of the input vector at each layer. This is a perfect analogy to the brain in which there are different regions for learning different characteristics of the input. The main advantage of FNN over traditional Machine Learning is its ability to learn the non-linear boundaries. Neural Networks can train a model by combining different distributions to one desirable complex distribution. When the data is not linearly separable, linear models fail to converge to good distributions [26].

The goal of the FNN is to approximate a function f^*. This is somewhat similar to linear regression where the goal is to find a function f where $y = f(x)$. In the similar fashion, an FNN defines a mapping between y and x, such that $y = f(x, \theta)$, where θ are the parameters of the FNN that result in the best possible approximation.

The layers between the input and the output layers are called hidden

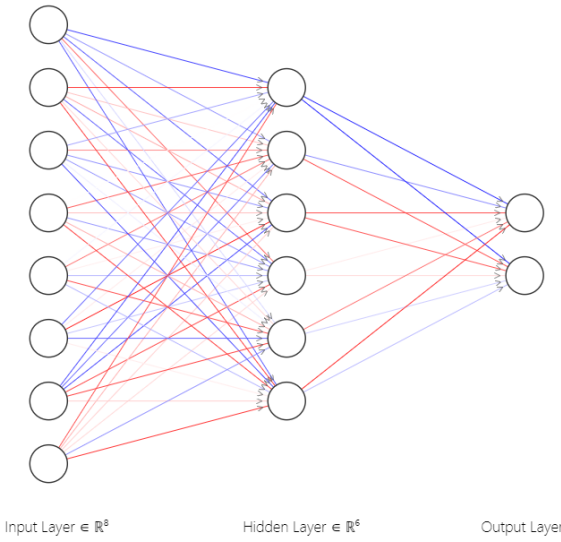


Figure 4.2: A Feedforward Neural Network with one hidden layer

layers. They are called hidden as the training does not show the output of these layers. There is no constraint on the number of hidden layers, a hidden layer can have as many hidden units as desired. The main task of the hidden layers is to increase the non -linearity, thus changing the representation of the data. The more the hidden layers are, the more complex model representation is [26].

In the experiments, FNN with different hidden layers and different complexity is used. Generally FNN can be represented as

$$f(x) = f^n(..f^2(f^1(x))..) \tag{4.1}$$

where
f^1 : *The input layer*
$f^{2....n-1}$: *The hidden layers*
$f(n)$: *The output layer*

For an instance, in Fig. 4.2, the NN is composed of an input layer, one hidden layer and one output layer. The output can be be represented in the form of $f(x) = f^2(f^1(x))$.

4.2 Training Neural Networks

In the training phase, the goal of the ML model is to find the optimal model parameter which produces the least error as per the loss/cost function. The

Loss/cost function of an ML model is defined as:

$$L(\theta) = L(\hat{y}, y_{true}) \tag{4.2}$$

An Optimizer is used to minimize the loss/cost function. It is used for updating the values of the weights and biases after every training cycle or epoch until the cost function reaches the global optimum. These algorithms minimize or maximize the value of a cost function using its gradient values with respect to the parameters. Neural Networks are trained by using iterative, gradient-based optimizers. There are many optimizers that are used for training the model like Stochastic gradient descent, Adagrad, Adam, RMSProp, etc. [26].

NN learns to predict the pre-defined targets from the provided training dataset. To get high accuracy, the NN needs to be trained on large training datasets. If the dataset is not large enough for the training, the network might lead to overfitting. This results in lower accuracy when it is tested on general data, which means the model is not generalized.

Two different sets are required for the network. The training set for training and the test-set for testing the network. The training set is further split into training and validation subsets that are used during the training process. The validation set is used to update the weights of the neurons which minimizes the loss function. After each epoch of the training process, the validation determines the performance of the training algorithm. The validation set evaluates when to stop the model which is used for preventing over-fitting [26]. The testset is then used to test the network after the whole training process is done. The test dataset should be completely new to the model i.e., it has never seen such samples before. Otherwise, the report loss measure is too optimistic [26].

4.3 Optimization

4.3.1 Activation function

An Activation function is the mimic of the stimulation of a biological neuron. In a Neural Network, the Activation function applies a transfer function to each neuron in the network. On a broader level, NN is just a combination of different linear functions. It is known that such a linear combination will always result in linear output. Therefore Activation functions are needed for the non linear transformations [50]. Another aspect to consider about an Activation function is its differentiability. The gradient of the loss func-

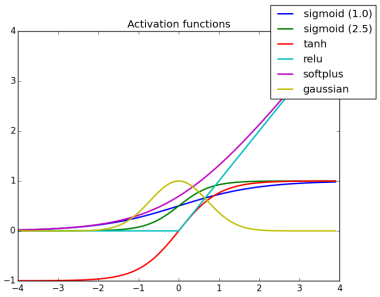


Figure 4.3: Activation Functions

tion is computed through the chain rule, using the output from each layer. Therefore, the Activation function must be differentiable [50].

Fig. 4.3 shows the common Activation function used in the study. Rectified Linear Unit (ReLU) is the most widely used activation function. It is zero if the input is negative and the same as input if it is positive. It resolves the problem of vanishing gradients. A Vanishing gradient occurs when the gradient is extremely small, thereby effectively preventing the weight from changing its value. There are other variants of ReLU like Leaky ReLU and Exponential Linear Unit which allows learning even the negative inputs. Other common Activation functions are Sigmoidal and Tanh. Sigmoidal restricts the output to [0,1] interval. It is often used when the output is in terms of probability. Tanh or hyperbolic tangent on the other hand transforms the output to [-1,1] interval.

$$Sigmoid(x) = \frac{1}{1 + \exp(-x)} \tag{4.3}$$

$$Tanh(x) = \frac{\exp(x) - \exp(-x)}{\exp(x) + \exp(-x)} \tag{4.4}$$

$$ReLU(x) = max(0, x) \tag{4.5}$$

4.3.2 Loss Function

Loss helps to analyze the difference between the predicted value and the actual value. The Function used to quantify this loss in the form of a numerical entity is called a Loss/Cost function.

The cost function can vary based on the type of problem. For an instance, same cost function can't be used for Regression and Classification. In FNN

for regression, the focus is on minimizing the quadratic cost [26]. Eq. (4.6) is a Mean Square Error type of cost function .

$$C(\theta) = \frac{1}{2N}\Sigma_x \|y(x) - f(x))\|^2 \tag{4.6}$$

There are many other forms of cost functions for regression problem apart from MSE like Mean Error, Mean Absolute Error (MAE), Mean Square Logarithmic Error (MSLE), Root Mean Square Error (RMSE).

For a binary Classification problem, the typical cost functions are Binary Cross-Entropy, Binary Cross-Entropy with Logit Loss, Hinge Loss, Squared Hinge Loss. For a multiclass Classification problem, the cost functions can be Multi-Class Cross-Entropy Loss, Sparse Multiclass Cross-Entropy Loss, Kullback Leibler Divergence Loss. For the purpose of this thesis, the chosen loss function is Binary Cross-Entropy. It measures the dissimilarity between the predicted probability and the actual value by determining the number of extra bits required to encode the predicted labels.

S.No.	Name	Type of Problem	Formula
1	MAE	Regression	$\frac{1}{N}\sum x\|y(x) - f(x))\|$
2	RMSE	Regression	$\sqrt{\frac{\sum_x (y(x)-f(x))t}{N}}$
3	Binary Cross Entropy	Classification	$\sum_{i=1}^{N} y(x)\log(f(x))$
4	Categorical Cross Entropy	Classification	$\sum_{j=1}^{M}\sum_{i=1}^{N}(y(x_{ij}) * \log(f(x_{ij})))$
5	Hinge Loss	Classification	$max(0, 1 - f.y)$

Table 4.1: Common cost functions with the type of problem they are used in.

4.4 Regularization

One of the quintessential aspects of training your ML model is avoiding overfitting and underfitting. The model trained might sometimes lead to underfitting. This happens when the complexity of the model is not enough to incorporate the complexity of the training dataset. Underfitting can be improved by increasing the depth or the width of the Neural Network [26].

On the other hand, The model will have a low accuracy if it is overfitting. This happens when the model is trying too hard to learn the training dataset,

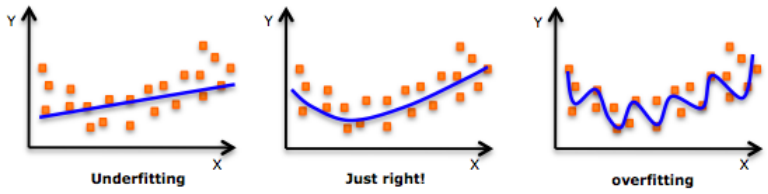


Figure 4.4: Representation of model underfitting and overfitting [26]

or, it learns too much that it is not able to generalize on the new unseen data. The model even learns the random noises in the dataset. Noise means the data points that do not really represent the true characteristics of your data, but random chance. This is when Regularization comes into the picture [26].

Regularization is a technique to shrink the coefficient estimates towards zero. Reiterating, this technique discourages the machine to learn complicated models to avoid the risk of overfitting. There are many regularization techniques used in Machine Learning, the most common ones being Ridge and Lasso Regression. In Deep Learning, apart from these common regularization techniques, some specific techniques are used, for example, Batch Normalization, Layer Normalization, Pooling, Early Stopping, Gradient Clipping, and Dropout. All these normalization techniques have been used in this thesis work while training the Deep Learning models.

4.5 Hyperparameter Tuning

Hyperparameter tuning is the process to choose the set of optimal parameters for a learning algorithm. A hyperparameter is a parameter that can not change during the training process, i.e., their values are set prior to the learning phase. However, there are some parameters that are changed for learning. For an instance, in an FNN, the width and the depth of the network are parameters since they need to remain the same during the training process. On the contrary, the weights of every unit or the neurons are optimized as the learning proceeds, i.e., they are adjusted whenever new data is provided in order to generalize to the new set as well.

In order to set the Hyperparameters, two different types of optimization processes are used in this thesis.

- **Grid Search** - It is a brute force technique of optimization which works by searching the provided subset exhaustively. The advantage of Grid Search is that it is guaranteed to find the optimal combination of the parameters specified. However, it is very computationally expensive.

- **Random Search** - It is less computationally expensive than grid search. In Random Search, it searches the specified subset randomly rather than searching them exhaustively. In Random Search, we are not guaranteed to find the optimal combination.

4.6 Convolutional Neural Networks

Convolutional Neural Networks (CNN) are trainable models with multiple processing layers and many parameters. In CNNs, the convolution mathematical operation is used instead of general matrix multiplication.

A sliding window called kernel is used to perform the convolution process. The input and the kernel are usually multidimensional arrays. The parameters of the Kernel are adjusted during the learning process. The input is convolved with the Kernel in order to create an output. In the case of a 2D image, the input would be a matrix of the number of pixels and the kernel would be a 2D convolution sliding window.

For example, in Fig. 4.5, the matrix in blue shade represents the input, the kernel is represented by the navy blue matrix. The Kernel is convoluted with a portion of the input, which generates one section of the output matrix.

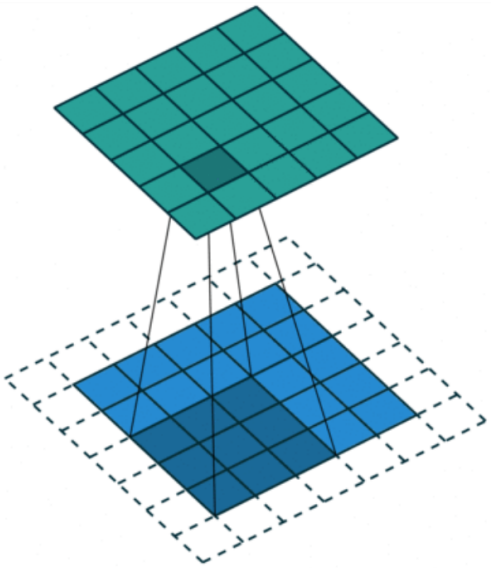


Figure 4.5: The input, the kernel and the output [59]

There are many advantages of using CNNs instead of FNNs in some scenarios like image Classification.

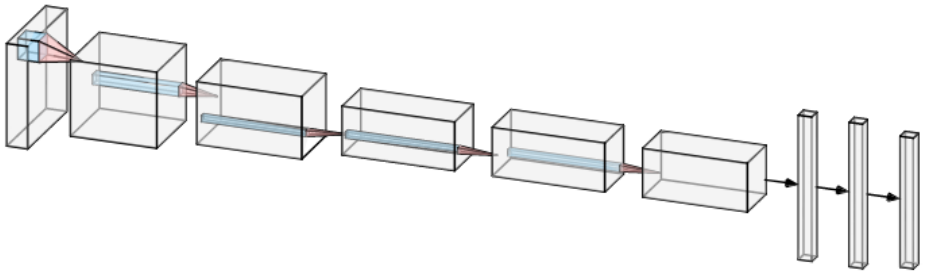


Figure 4.6: The structure of CNN.
Created by alexnail.me [35]

- **Fewer parameters**: CNN uses a small set of parameters i.e., the kernel to calculate outputs of the entire image, so the model has much fewer parameters compared to an FNN.
- **Sparsity of connections**: In each layer, each output element depends only on a small fraction of input elements, thus making forward and backward passes more efficient.
- **Parameter sharing and spatial in-variance**: The features learned by a kernel in one section of the image can be used to detect a similar pattern in a different section of another image.

4.6.1 Structure of Convolutional Neural Network

Every layer in Convolutional Neural Network is composed of three dimensions: width, height, and depth. The depth defines the number of channels (filters) for the layer. For example, In an RGB image of 28x28 pixels, the input will have the dimension of 28x28x3 where the width is 28, the height is 28 and the depth is 3. Figure 4.6 shows a representation of how CNN layer is structured. Each layer in a CNN receives a 3D input and transforms it into an output applying a differentiable function [38].

In this thesis, the PyTorch library has been used for building Convolutional Neural Networks. Different structures of the model were tried using training and testing set in order to minimize the loss function. The RADAR signal is converted to a 2D image map and then this image map was fed into the model.

CNNs have been used in this thesis because they are useful in the image/image map domain. CNNs are chosen in this thesis because pattern

recognition is the main concept in detecting RADAR signals. The signals subjected to motions have a distinguishable pattern as compared to the signals where there is no motion. Therefore the CNN network can detect useful features from the pattern in the input data.

4.7 Statistical Evaluation Metrics

A loss/cost function in a neural network is used to look for the optimal ML model that minimizes overall error. However, the loss/cost function does not encapsulate all aspects of the quality of the predictions. For classification tasks, simple statistical metrics come in handy. The simplest metric is Binary Accuracy which is defined as the number of bit changes required to convert a binary vector to another. This is illustrated by an example below.

Suppose,

$$Y = [01101]; Y = [01001],$$

where Y is the predicted output and Y is the actual ground-truth value.

There is a requirement of only one bit change to convert the predictions to the ground-truth, therefore the accuracy of the model is $1/5 => 80\%$. In binary classification problems with imbalanced class ratios, it is helpful to understand the number of times a model predicts each class. For example, consider a cancer detection dataset that is highly skewed with 98% class 0 samples. An ML model can achieve 98% accuracy on a dataset by predicting class 0 all the time. But in reality, this model is extremely bad. Therefore, to accurately describe model predictions, three other classification tests i.e., Precision, Recall, F1-Score will be used along with Accuracy.

4.7.1 Confusion Matrix

A confusion matrix categorizes the predictions into four categories, i.e., True Positive (TP), True Negative (TN), False Positive (FP) and False Negative (FN). In lighting control, TP represents correct occupancy prediction. This means the luminaire will be switched on when an occupant is present in the room. A TN represents correct in-occupancy prediction, i.e., the lights will be switched off when the space is empty.

An FP incorrectly predicts occupancy, which means the luminaire is turned on even when there is in-occupancy. This leads to energy wastage. Moreover, an FN will switch off the luminaire in an occupied room, resulting in a poor user experience. It is imperative that ML the model should strive to minimize FP and FN, and maximize TP and TN.

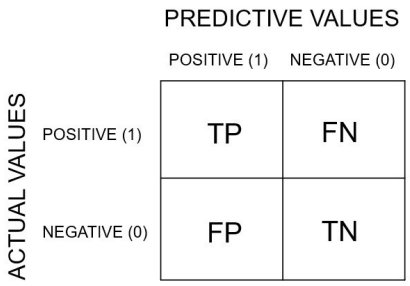


Figure 4.7: Confusion Matrix for Binary Classification.

4.7.2 Precision/Recall and F1-Score

Precision and Recall are statistical tests that include elements of the confusion matrix. Precision is defined as the **fraction of relevant instances among the retrieved instances**, and is given below:

$$Precision = \frac{TP}{TP + FP}. \tag{4.7}$$

Recall is defined as the **fraction of relevant instances that were retrieved**, and is given by:

$$Recall = \frac{TP}{TP + FN}. \tag{4.8}$$

The F1-Score is another statistical test that finds a balance between Precision and Recall. It is the harmonic mean of both and is defined by:

$$F1 - Score = \frac{2 * (Precision * Recall)}{Precision + Recall}. \tag{4.9}$$

Chapter 5

Implementation

This section discusses the implementation details of the Machine Learning model explained in the previous chapters. We will also discuss a baseline algorithm that provides fair results. The goal of the Machine Learning algorithms will be to beat this baseline algorithm with a sufficiently big margin.

There are various Deep Learning frameworks available in the market these days, the most common being Tensorflow and PyTorch. Both of these frameworks are open source. To build the deep learning models, **PyTorch** has been used. PyTorch has been chosen as building models is more intuitive and pythonic. Another reason for choosing PyTorch is its Object Oriented Programming (OOP) style.

5.1 Baseline Algorithm

A baseline is a method to create predictions for a dataset that uses heuristics, randomness, or simple summary statistics. This metric will then becomes the foundation of what you compare any other machine learning algorithm against. In this thesis, a simple distribution method is used to create a baseline model. The DL model in the upcoming chapters should always beat the predictions of this baseline algorithm.

Every cell in the input data corresponds to the intensity of an object at some range moving at some velocity. Intuitively, if there is no moving object in the room, the sum of all these intensity values should be much lower than the sum of these intensity values in case of movement. Fig. 5.1 shows the histogram of the summation of every value in each data input. A clear difference can be seen in the case of Presence and Absence.

The idea behind the baseline algorithm is to find the data distribution belonging to the Presence and Absence data set. The mean of the sum of the

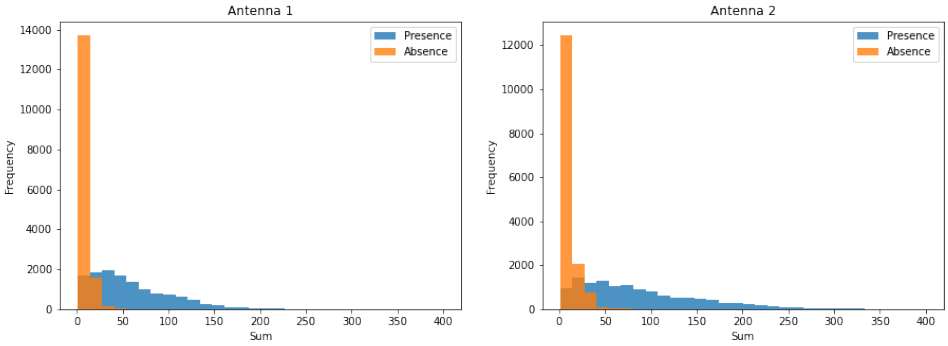


Figure 5.1: Presence and Absence distribution in Train set

values corresponding to Antenna 1 and Antenna 2 is used to find two different distributions. Training dataset is trained on these distributions and validated on the validation dataset. Once the optimal distribution is obtained, the test dataset is then evaluated on these distributions. The predicted value is the distribution for which the Probability density function(pdf) is more.

5.2 Data Preparation for PyTorch Models

For the Deep Learning models, the dataset is processed into pairs of input and labelled outputs for supervised learning. Let $X$and Y denote the complete set of data sample pairs, and x_t and y_t represents the input and output pair of data at index t. As explained in section 5.2, every x_t is a two dimensional matrix of size 30×30. Range-Doppler gave the signal of size $2 \times 32 \times 32$. However, as explained before, to avoid the digital component, the first and the last bins have been removed. $X$and Y can be represented in the following mathematical notation,

$$X = \{x_t | (x_{ijk}) \in \mathbb{R}^{2\times30\times30} \; and \; t \in T\},$$
$$Y = \{y_t \in \{0, 1\} \; and \; t \in T\},$$

where T is the index of the set: $T = \{0, 1, 2, ..N - 1\}$ and N is the total number of samples in the dataset.

Transformations on the Data

The dataset is then stacked in PyTorch supported data type to create a PyTorch tensor dataset. While converted the dataset into the PyTorch dataset,

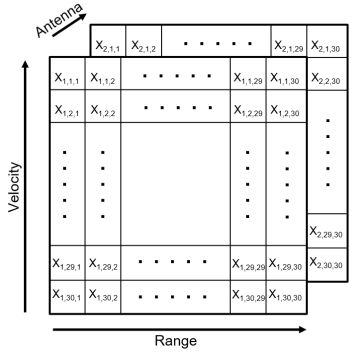


Figure 5.2: Data structure

PyTorch gives the provision to do the transformation on the dataset. These transformations are common 2D image transformations. Since the input dataset we are using resembles an image dataset, we can leverage these transformation functions. It is important to note that these transformations will not be the same across calls. Randomized transformations will apply the same transformation to all the images of a given batch, but they will produce different transformations across calls [7].

For this thesis, the chosen transformations are Gaussian blur and Random Cropping, and Random horizontal flip. Gaussian blur blurs the input with randomly chosen Gaussian parameters. It is chosen because it will help in noise reduction for the RADAR signal. Random cropping crops the input to the desired size and if the output size is more than the input, it pads the input with a specified padding mode. The specified padding mode here is 'reflect'. Reflect pads with a reflection of input without repeating the last value on the edge. Random cropping is very important for this task as it may be possible that some packets of the RADAR are lost over the communication channel. In case the input to the NN model is variable in size, it will run into an error. Random horizontal flip has been introduced to give more variety to the dataset. This is because x axis represents the range and y axis represents the velocity. It may be possible that while collecting the dataset for the presence scenario, there was high skewness between near moving objects and far moving objects. However, we want to give equal weightage to near and far objects in the presence detection scenario as long as our RADAR is set to a certain range threshold. Therefore horizontal flip will help in solving this problem.

Both the models specified in the following sections were evaluated **with** and **without** transformations. The transformation flag is a Hyperparame-

ter that will be tuned along with other Hyperparameters in the upcoming sections.

5.3 Network Design

In Chapter 4, we already discussed the underlying building blocks for developing a Neural Network. In this section, we shall discuss the Neural Networks from an implementation perspective. Two main types of neural networks are implemented in this thesis, i.e., Feed Forward Neural Network and Convolutional Neural Network.

5.3.1 Feed Forward Neural Network

Designing an FNN is fairly simple. As mentioned in the section 4.1, an FNN needs vector as in input. Therefore the first step is to flatten the input data as shown in Fig 5.3. This flattened 1D vector of shape $[1 \times 1800]$ is then fed to the FNN.

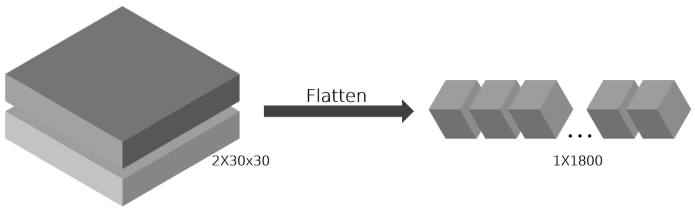


Figure 5.3: 2D matrix flattened into 1D vector

An FNN consists of an input layer, hidden layers, and then an output layer. Usually, the terms depth and width are used for defining the FNN structure. The depth of the network corresponds to the number of layers in the network and the width of the network is defined as the number of neurons in each layer. For the sake of regularization Batch Normalization and Dropout layers are also used for model building. Fig. 5.4 depicts the overall structure of the FNN.

FNN model structure used in this thesis introduced some hyperparameters, namely depth of the network, breadth of the network, the activation function to be used and the probability of the dropout layer. A custom class is built for the model building which helped in creating a customized FNN with variable hyperparameters. Different combinations of these hyperparameters were tried and tested for better accuracy. Hyperparameter tuning

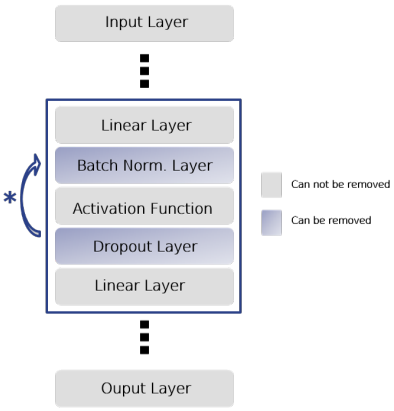


Figure 5.4: A flexible FNN structure with regularization layers - Batch Normalization and Dropout

is explained in more detail in the upcoming section. It is worth noticing, the hyperparameter discussed in this section only deals with how the FNN structure is created, it has nothing to do with the training phase, i.e., hyperparameters that decide how learning takes place, such as learning rate, momentum, early stopping, loss function, optimization function, etc.

Hyperparameter	Sampling set
Depth	{3,4,5,6}
Width	$\{2^5, 2^6, ..., 2^9\}$
Activation Function	$\{ReLU, Tanh, Sigmoid\}$
Dropout Probability	{0.1, 0.2, 0.3}

Table 5.1: Hyperparameter related to FNN model structure

5.3.2 Convolutional Neural Network

A CNN consists of single or multiple channeled input, followed by multiple Convolutional blocks, followed by single or multiple fully connected layers, and finally an output layer. The Convolutional block contains a Convolution layer and an Activation layer and may consist of other layers, such as the Batch Normalization layer, Dropout layer, or a Pooling layer. Usually, the term **Kernel size** is used for defining the Convolutional layer, which is a small matrix of weights and is used to slide over the 2D input data. It basically performs an element-wise multiplication with the part of the input

it is currently on, thereby resulting in summing up the results into a single output pixel. Apart from the Kernel size, other important attributes of a CNN are **Padding** and **Stride**. Padding helps in keeping the size of the input and output the same, and Stride is basically how many steps the Kernel will move in each step in convolution. For the sake of regularization Batch Normalization and Dropout layers are also used for model building. To reduce the spatial volume of input data after convolution, Pooling layers are used. Fig. 5.5 depicts the overall structure of the CNN used in the thesis.

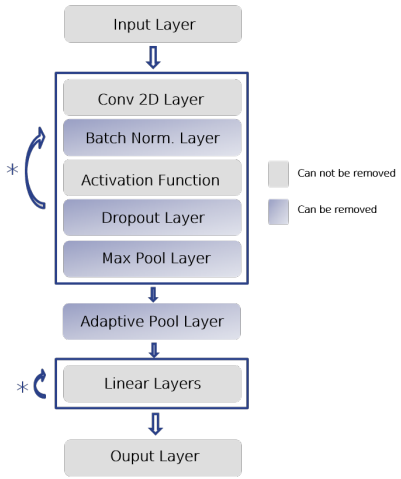


Figure 5.5: A flexible CNN structure with regularization layers - Batch Normalization, Dropout and Pooling Layers

Hyperparameter	Sampling set
Kernel Size	{3,5,7}
# CNN Blocks	{1,2,3,4}
Padding	{1,2}
Stride	{1,2}
# Output Channels	{64,128,256,512,1024}
# Linear Layers	{1,2}
Activation Function	$\{ReLU, Tanh, Sigmoid\}$
Dropout Probability	{0.1, 0.2, 0.3}

Table 5.2: Hyperparameter related to CNN model structure

Similar to FNN, CNN also introduces a lot of Hyperparameters to be decided before model training. Even in this case, a custom class has been

designed to incorporate the space for different Hyperparameters. Different hyperparameters were tested to make the model more stable, less prone to overfitting and more accurate. Table 5.2 reflects on the hyperparameter space related to the designing of CNN. Again, not to forget, these hyperparameters do not deal with how training was performed, they just design how the model layer looks.

5.4 Model Training

5.4.1 Minibatch Training

Neural Networks are trained with gradient descent where the error is used to update the weights. Since the error is statistical in nature, it means the more the training examples are used, the more accurate result will be. Optimization algorithms that use the whole dataset for training simultaneously in one go are called Batch Gradient Descent. On the other hand, using fewer examples in one run will give a less accurate and noisy estimate. Nevertheless, these noisy updates can result in more robust estimates if used over several passes. The technique where one sample is used at a time to estimate the update is called Stochastic Gradient Descent. A configuration where more than one sample but a subset of the whole dataset is used in one iteration is called Minibatch Gradient Descent and the training using Minibatch GD is called Minibatch Training [14]. Generally, small batch sizes are used in NN as Small batch size makes it much easier to fit into GPU memory. Additionally, the estimates are usually noisy, therefore offering a lower generalization error by implicit regularization.

In PyTorch, Minibatch training is often done by using DataLoaders. It corresponds to fetching Minibatch of data and collating them into batched samples using random sampling [3]. Random sampling does not preserve the order of the dataset. The order is important for models like RNNs where temporal conformity needs to be maintained. However, in this thesis only FNNs and CNNs are used, therefore random sampling can be used. The reason why CNNs and FNNs provide good results is that every data sample represents a signal of one second and this one second window is more than enough for classification problems.

Choosing the batch size is again a matter of attention. Yoshua Bengio [11] argued that Minibatch size is generally selected between 1 and a few hundred with B= 32 being a good default value. In this thesis a default value of 32 is chosen for initial runs, nonetheless, minibatch size is another hyperparameter that will be tuned along with other hyperparameters in the hyperparameter

tuning phase. The models have been training using B = [32,64,128].

5.4.2 Training

Every minibatch created is fed to the NN and a loss is calculated using the loss/cost function. The loss incurred from every minibatch is then backward propagated to update the weights of the NN using an Optimizer. Once all the minibatches of the dataset are used to update the weights of the network, it is referred to as the completion of one epoch. Usually, the dataset is trained over multiple epochs for robust results. The rate of change of weights is decided by the learning rate.

The training step also includes some parameters which are supposed to be decided before model creation, i.e., Hyperparameters. These hyperparameters have nothing to do with model structure. In this thesis for the training step, many hyperparameters are considered apart from batch size i.e., the learning rate, number of epochs, loss/cost function, the Optimizer, early stopping, and gradient clipping. Table 5.3 reflects on the hyperparameters used for the training step and their corresponding sampling set.

Hyperparameter	**Sampling set**
Learning Rate	loguniform distribution $\{1\exp{-4}, 1\exp{-1}\}$
Max Epochs	30
Loss/cost function	{BceLoss, BCEWithLogitsLoss, BCEWithLogitsLoss with positive weight}
Optimizer	{Stochastic gradient descent, Adam, ASGD, RMSProp,Adagrad}
Gradient clipping	{None, 0.1, 0.2}
Early Stopping	{True, False}
Bath size	{16,32,64}

Table 5.3: Hyperparameter related to model training

5.4.3 Hyperparameter Tuning

With advancement in Deep Learning, new algorithms for hyperparameter tuning are becoming better and better, such as algorithms like HyperBand [37], ASHA [38] or Population Based Training (PBT) [29]. PBT achieves superhuman performance while ASHA terminates the trials that are less desirable and allocates more time and resources to better trials. Thus, it helps in increasing the search space by 5x. Nevertheless, Most of the hyperparameter

search frameworks do not have these new optimization algorithms. To make the hyperparameter tuning process smooth and friction less, and to leverage on all the new state of the art algorithms, **Ray Tune** library [6] has been used.

Tune is a powerful library built on the top of Ray which is designed not only for hyperparameter tuning, but also for scaling across the larger distributed clusters. Tune has built-in fault tolerance, trial migration, and cluster auto-scaling, which allows to safely leverage spot (preemptible) instances, thereby reducing cloud costs by up to 90% [39].

For this thesis, the state of the art algorithm i.e., ASHA have been used. Hyperparameters defined in Table 5.3 and (Table 5.1 or) are fed into the selected algorithm

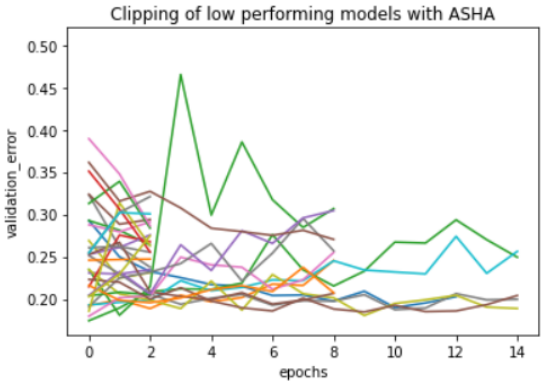


Figure 5.6: ASHA terminating the less desirable trials

S. No	Stopping Metric	Mode
1	Validation set accuracy	Max
2	Validation set loss	Min
3	Validation set recall	Max
4	Validation set AUC	Max

Table 5.4: ASHAScheduler attributes

However, it is important to notice here that the ASHA expects a few parameters to be explicitly provided. The first one being the stopping criteria metric and mode pair, i.e., which metric should be used for early stopping. Different metrics were tried along with ASHA Scheduler. A Mode defines if

the metric should be stopped as per maximum value achieved or minimum value achieved. Another argument which ASHA expects is how many different combinations of the search space it will try. Table 5.4 shows different combinations tried

5.5 Flow Chart

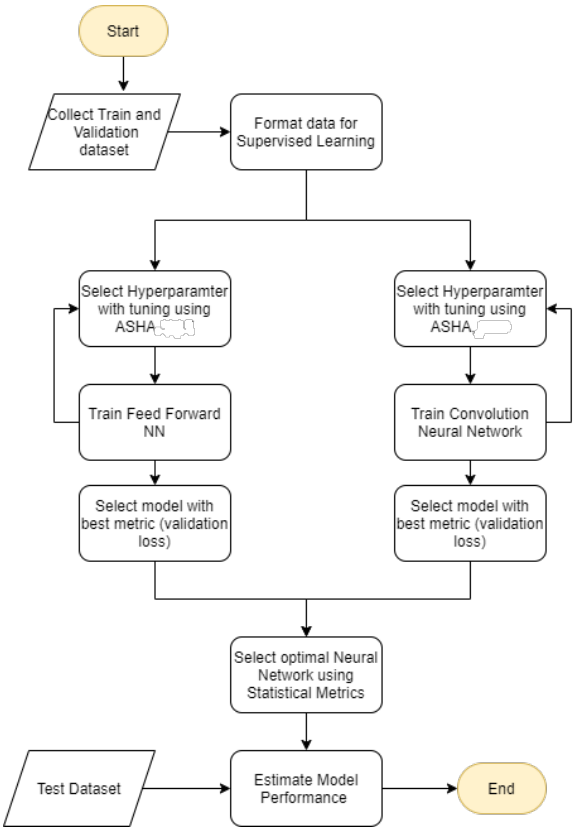


Figure 5.7: A complete flow chart for Implementation

Chapter 6

Evaluation

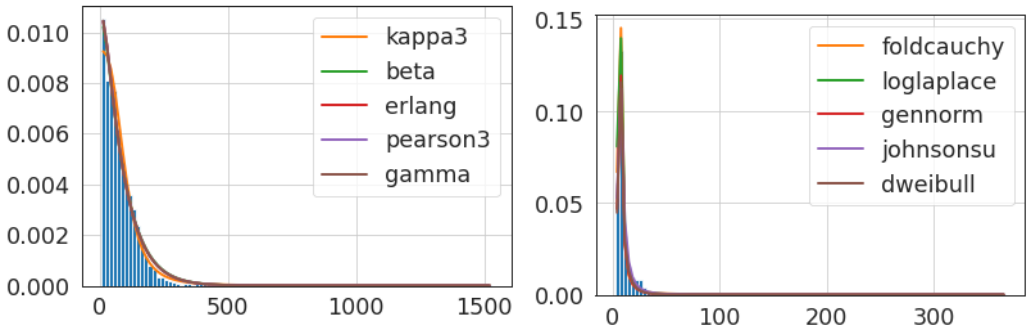
6.1 Evaluation of Baseline Algorithm

The train dataset is split into two datasets based on the labels i.e., Presence and Absence. Distribution of these Presence and Absence dataset is obtained by testing various different distributions. Table 6.1 shows the Top 5 distributions which correspond to these datasets along with Sum Square Error, Akaike information criterion(AIC), and Bayesian information criterion(BIC). Smaller Error, AIC, and BIC corresponds to better results. It can be seen from the table that for the Presence dataset, **kappa3** gave the best results and for the Absence dataset **foldcauchy** gave the best result. Fig. 6.1 shows the Top 5 distributions plotted on these datasets.

Presence				**Absence**			
Dist	Error	aic	bic	Dist	Error	aic	bic
kappa3	0.000004	2225	-290311	foldcauchy	0.000291	1928	-276033
beta	0.000004	2903	-288914	loglaplace	0.000495	2187	-267773
erlang	0.000004	2915	-288733	gennorm	0.000506	3006	-267437
pearson3	0.000004	2915	-288733	johnsonsu	0.000604	2414	-264673
gamma	0.000004	2913	-288659	dweibull	0.000642	3816	-263738

Table 6.1: Best Distributions on Presence and Absence Validation dataset

The validation dataset was evaluated using kappa3 and foldcauchy distribution. If the PDF of kappa3 is more, the predicted target is Presence, and if the PDF of foldcauchy is more, the predicted target is Absence. Using this logic, statistical metrics are obtained for the baseline algorithm. From the table, it can be seen that the baseline algorithm predicted the results with



(a) Distribution for Presence Data (b) Distribution for Absence Data

Figure 6.1: Data Distributions

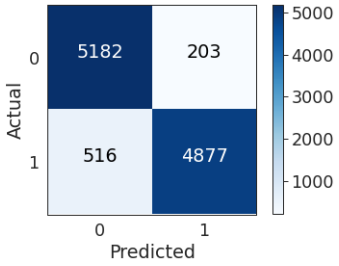


Figure 6.2: Confusion Matrix

Baseline	
Metric	Value
Accuracy	0.933
Precision	0.960
Recall	0.904
F1 Score	0.9313

Table 6.2: Statistical Metrics

the accuracy of 93.3%, the precision of 96%, Recall of 90.4%, F1 Score of 93.13%.

6.2 Evaluation of Feed Forward Neural Network

The Feed Forward Neural Network is evaluated using hyperparameter as shown in Table 5.1 and Table 5.3. Different combinations of the Hyperparameters were tried to choose the best combination of the space. In this section, we will select the best hyperparameter space using an exhaustive search. It is worth noticing that using an exhaustive search, the values corresponding to the Hyperparameters like Learning Rate are always discrete. It is impossible to check all the continuous values. Therefore, in the last subsection, we will compare the performance of the best Hyperparameter space obtained using an exhaustive search with that of the ASHA scheduler.

Different **Loss Functions** were tried to check which Loss Functions work

best on the dataset. Therefore various combinations were tried and different models were trained. Fig. 6.3 shows the comparison between different Loss Functions. A large Hyperparameter space was chosen to make a strong decision while selecting the Loss Function. It is clearly visible that Adagrad and RMSprop tend to fluctuate the validation loss a lot. For the same Hyperparameter space, **SGD** and **ASGD** works in a more stable fashion.

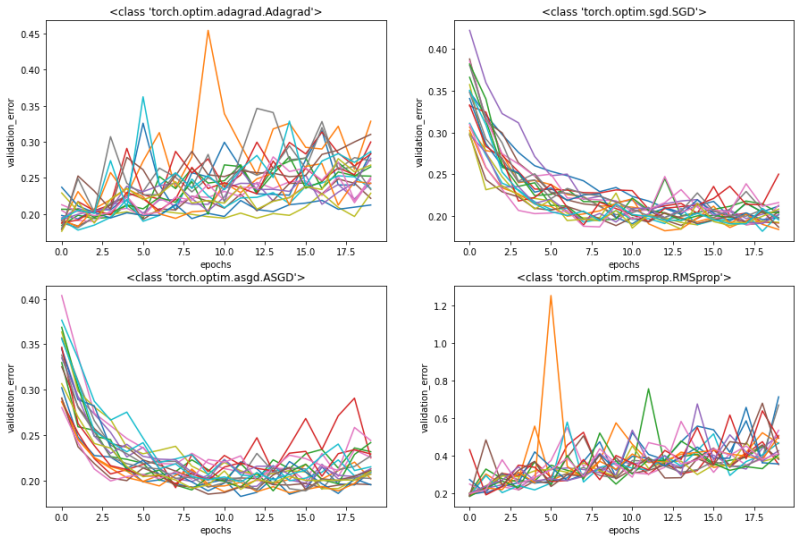


Figure 6.3: Comparison of different Loss Functions

Different **Learning Rates** (α) were tried to check which Learning Rates work best on the dataset. Learning Rate was selected from a random uniform distribution between [1e-4,1e-2]. Fig. 6.4 shows the comparison between Learning Rates when divided into 3 classes. It is clearly visible in the figure that for low Learning Rate i.e., when the Learning Rate $< 4e{-}3$, the convergence is very slow, and for Learning Rate $> 1e{-}2$, the learning starts to fluctuate again. For the range $[4e-3, 1e-2]$, the learning is quite stable and fast.

All the other hyperparameters were kept at some default values for the sake of visualization, since it was not possible to show all the comparisons. **BCELoss** is used as a Loss function since it is Binary Classification. The Activation function used is **ReLU** for the initial layers and **Sigmoidal** for the last layer. Tanh Activation functions performed inferior as compared to ReLU. The Dropout Probability of **0.2** worked out to be best. The Learning rate is chosen in **0.01**.

A very important Hyperparameters for FNN is its depth and Depth. Fig.

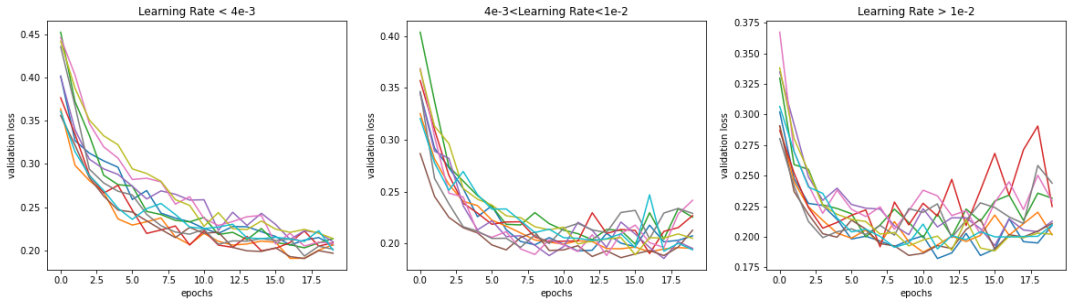


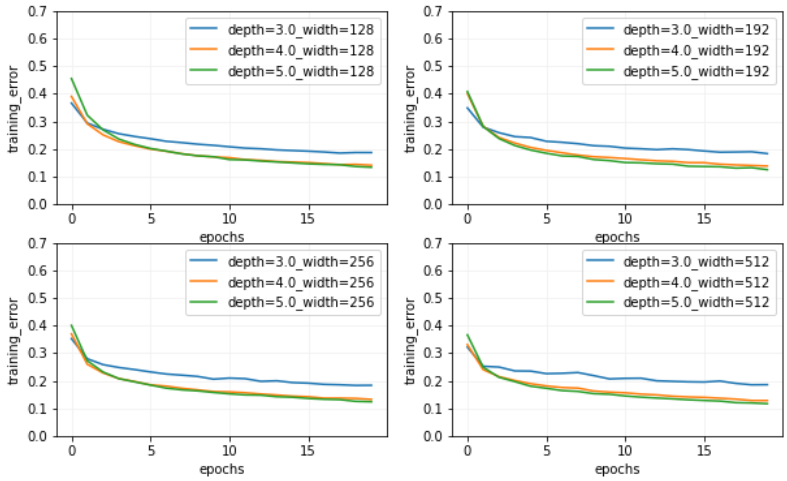
Figure 6.4: Comparison of different Learning Rates

6.5 depicts how the loss on training and test set varies by changing the depth and width of the network. It can be seen clearly for 6.5a that the model for depth 3 is underfitting. This is because even after 20 epochs, the training error reduces to an extent and then tends to remain the same. For depth 3, the model always underfits even with a large width. No matter how many more neurons are added in every layer, it always underfits. However, training loss is comparable for the models where depth is 4 or 5. Nonetheless, the model with depth 4 performs with much lower validation loss for all the different depths provided. It can be clearly visualized from the plot 6.5b that for all the depths i.e., 128,192,256, and 512, the model with depth 4 results in much lower validation loss as compared with the models where depth is 5. Also, the models with depth 4 are much more stable as compared to the models with depth 5. There is very little fluctuation in the models with depth 4, which also indicates the models are not prone to overfitting.

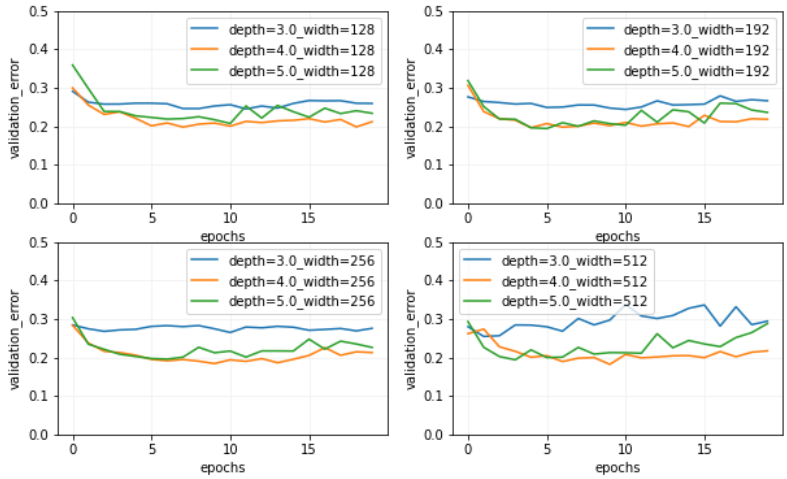
It is clear from the previous paragraph that the model with depth 4 gave the best result, though the analysis for the best width is still left. Fig. 6.6 compares different models with variable widths and the same depth 4. The model with width 512 performed best and achieved the best validation loss. Nonetheless, the model with with with 256 performed equally good. Even though the model with width 512 gave the best result, but we will still consider the model with width 256 for further analysis. The reason is-

1. Validation loss for width 256 is more stable over epochs, i.e., it is not fluctuating like 512. No matter how many times the model is retrained, there won't be a significant difference in the results obtained from the model with width 256.

2. In a lighting scenario, the emphasis is more on fast prediction. There is always a trade-off between faster prediction and more accurate pre-



(a) Training Error



(b) Validation Error

Figure 6.5: Train and Validation Error in FNNs with varying width and depth

diction. The Machine Learning model is deployed on the edge device whose computation power is little. Therefore it is always recommended to proceed with a less complex model in order to get a faster prediction.

Table 6.3 indicates the best validation error and their corresponding training errors for different versions of the FNN model. The top 3 versions of the FNN are shown in the bold text in the table. The model with depth = 4 and width = 512 performed best as per the validation loss. However, validation loss is not the only criteria to choose the model. The model with

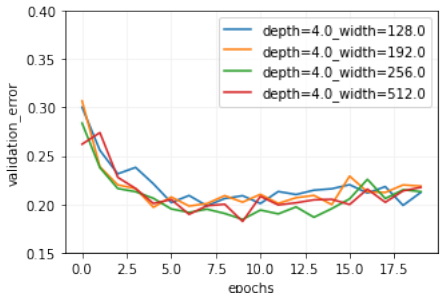


Figure 6.6: Validation Error for FNNs with depth 4

Width	128		192		256		512	
Depth	Train	Valid	Train	Valid	Train	Valid	Train	Valid
3	0.203	0.245	0.203	0.244	0.210	0.265	0.252	0.255
4	0.182	0.198	0.205	0.197	0.161	**0.183**	0.159	**0.182**
5	0.161	0.207	0.184	**0.194**	0.173	0.196	0.197	0.194

Table 6.3: Results with top 3 performing models

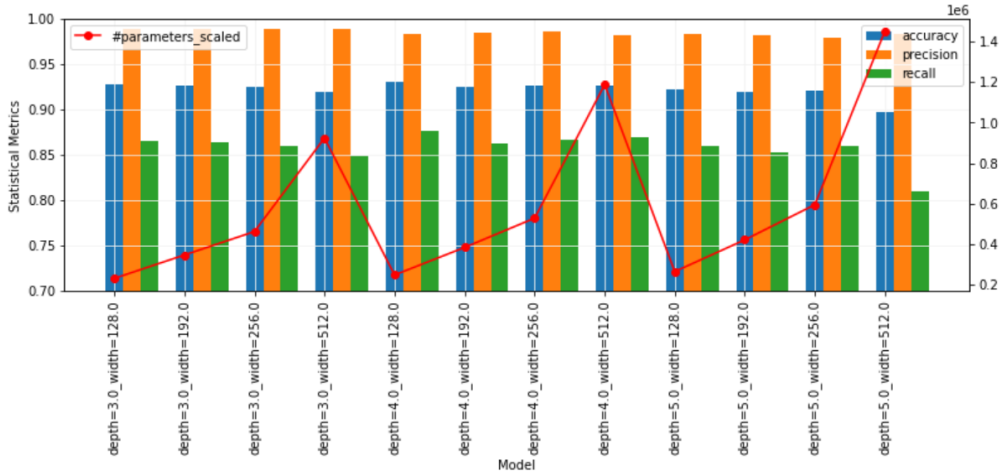


Figure 6.7: A statistical comparison of different FNN models

depth=4 and width=256 also performed with best the possible statistical metrics. Fig. 6.7 compares the different FNN models tried and compares the statistical metrics such as Accuracy, Precision, and Recall. It also shows the number of parameters that are required to train the model. It can be seen

that the statistical metrics for the two best models are comparable, although there is a huge gap between the number of parameters to be trained.

Hyperparameter selection for FNN with Ray Tune

To experiment more with the hyperparameter space, **50** different versions of the FNN model were evaluated using the **ASHA** algorithm. The objective is to explore if there exists a model with can beat the performance of the model discussed in the previous version. The metric used for early stopping is the loss function. The hyperparameter space experimented is discussed in the previous Chapter 5.

Table 6.4 shows the performance of the Top 10 performing models. It is evident from the table that model accuracy is high for a lower value of learning rate. The models discussed above were solely based on SGD as an optimizing function, however, after evaluation via Ray Tune, it is quite evident that **Adagrad** and **ASGD** performed equally good. Other optimization functions, such as Adam, RMSprop failed to give quality results. It is worth noticing that Adagrad as a Loss Functions gives quality results. But it fails miserably when the model is retrained with the same configuration. The reason is quite evident in the second plot in Fig. 6.3. The Table 6.4 highlights the best possible combinations achieved by ASHA. ASHA algorithm proposed the same combination of Hyperparameters as proposed by the exhaustive search. The only difference is the Learning Rate.

Name	Loss	α	B	Epochs	Depth	Width	Optim Fn.
1	0.174	0.0007	64	16	4	256	Adagrad
2	**0.178**	0.0006	64	14	4	256	SGD
3	**0.179**	0.001	64	8	4	256	SGD
4	**0.180**	0.00067	64	13	4	256	ASGD
5	0.181	0.0009	32	13	4	512	Adagrad
6	0.182	0.00094	64	5	4	128	ASGD
7	0.183	0.00025	64	7	4	128	Adagrad
8	0.188	0.00016	64	16	4	256	SGD
9	0.190	0.0007	32	3	4	128	Adagrad

Table 6.4: Hyperparameters corresponding to Top 10 performing models with ASHA for FNN

The top performing models are then compared based on their Training and Validation Loss. Even though the learning with Learning Rate 0.001

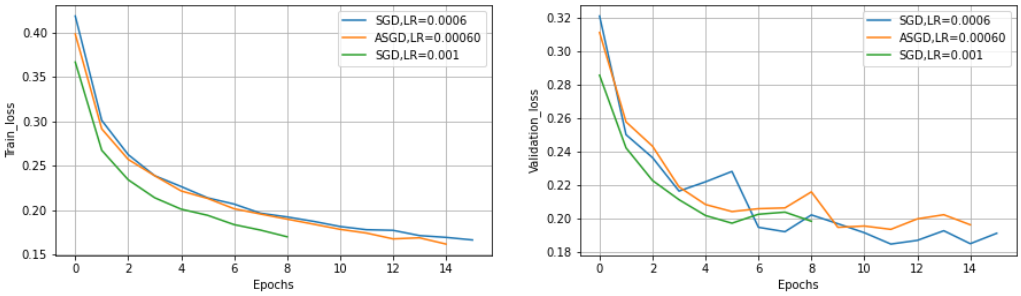


Figure 6.8: Training and Validation Losses for Top performing models of FNN

is faster, but the learning with Learning Rate 0.0006 is much more stable and validation losses are less. ASGD and SGD both gave the same results for Training Loss, but for Validation Loss, SGD works more smoothly. It is evident from Fig. 6.8.

Threshold Selection

One main consideration which was left in the discussion above is the threshold of the prediction itself. A NN gives the output in terms of probability and the threshold of 0.5 was used in the above discussion, i.e., if the probability is more than 0.5, the result is True, otherwise False. But the question is *Is choosing a threshold of 0.5 a good idea?*. It is important to tune the threshold of the model for the best performance metric.

Mathematically, the best threshold p is the one that satisfies:

$$TPR(p) = 1 - FPR(p), \tag{6.1}$$

where TPR is the True Positive Rate and FPR is the False Positive Rate.

It can be rewritten as:

$$p^* = arg\min_p |TPR(p) + FPR(p) - 1| \tag{6.2}$$

Fig. 6.9a shows the plot of TPR vs FPR for the best model selected. Nevertheless, this plot is not sufficient to evaluate the best threshold p. Using the Eq. 6.1, Fig. 6.9b depicts the Probability threshold vs $|TPR(p) + FPR(p) - 1|$. It is very much evident that the best probability threshold is definitely not even close to 0.5. It is close to 0.172. Fig. 6.9c demonstrates how the Statistical metrics vary by varying the threshold probability. It is

clear that beyond the threshold of 0.4, the Precision is unaffected, however, Recall drops of drastically. For a threshold less than 0.4, there is a trade-off between Precision and Recall, but a higher Recall is beneficial as it reduces the number of false-off events. Based on the plot, the peak F1-Score occurs at a threshold value of 0.172. Table 6.5 shows the comparison between the best models with three different thresholds.

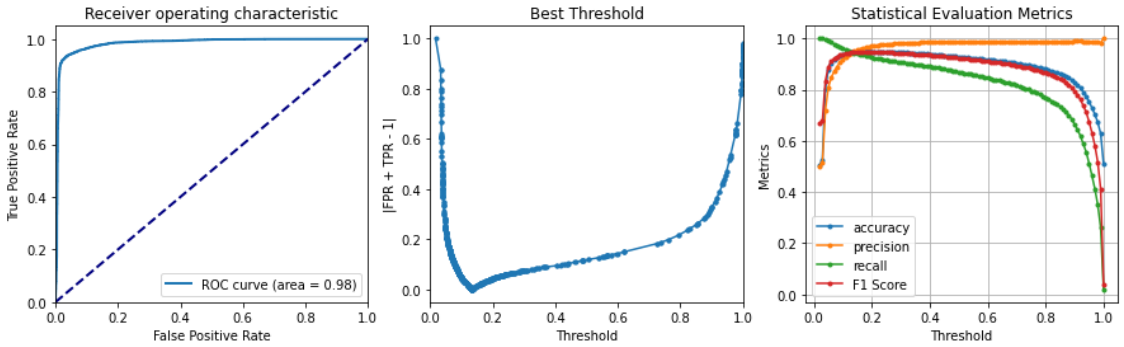


Figure 6.9: Optimal thresholds

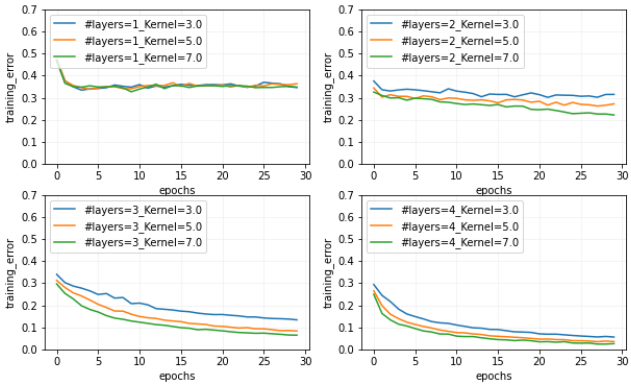
Network	Accuracy	Precision	Recall	F1-Score
FNN(p =*0.172*)	0.946	0.947	0.946	0.945
FNN(p =*0.25*)	**0.95**	**0.970**	**0.927**	**0.948**
FNN(p =*0.5*)	0.934	0.986	0.881	0.930

Table 6.5: Metric values at optimal thresholds for FNN

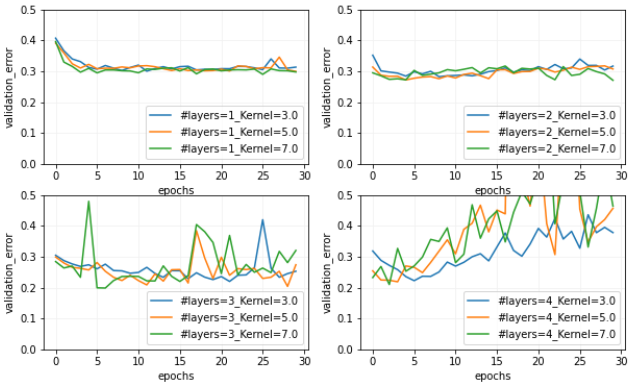
6.3 Evaluation of Convolutional Neural Network

The Convolutional Neural Network is evaluated using various Hyperparameters as shown in Table 5.2. It is not possible to show the training and validation error comparison between the whole hyperparameter space, therefore, for the sake of visualization, the only comparison between varying Kernel Size and the number of Convolutional layers is made. Fig. 6.10 depicts how the accuracy on training and test set is varying by varying Kernel Size and number of Convolutional layers. All the other hyperparameters were kept at

some default values. **BCELoss** is used as Loss function since it is Binary Classification, **SGD** has been used as an Optimizer with Learning Rate α of **0.001** with weight decay of **0.0001**. The number of output channels/feature maps of every Convolutional layer is set to **256**, and the Padding and Stride are set to **1** for every convolution operation. The Activation function used is **ReLU** for the initial layers and **Sigmoidal** for the last layer. **MaxPooling** is used after every Convolutional layer with Kernel size and stride of 2 which halves the number of rows and columns of the matrix. Finally, **AdaptiveAvgPool** is used that enables one output per output channel, which is then fed to the Linear layer.



(a) Training Error



(b) Validation Error

Figure 6.10: Train and Validation Error in CNNs with varying Kernel Size and number of Convolutional Layers

It can be seen clearly for 6.10a that for the first plot where the number of Convolutional layers is 1, no matter how much the Kernel size is increased, it tends to underfit. This is because even after 30 epochs, the training error

tends to stay almost the same. For the case when the number of Convolutional layers is 2, training error is going down at a much better pace but it still underfits. For the case with the number of convolutional layers being 3, Training error is surely going down at a good pace. The validation error with 3 Convolutional layers with Kernel Size 3 and 5 performs stably, however, the model with Kernel Size 7 tends to overfit after few epochs which can be seen from the fluctuations in the plot. For the case when the number of Convolutional layers is 4, the training error is going down very fast, however, it can be seen that the validation error started to fluctuate again. This is because of overfitting. This is very evident that the problem we are trying to solve here does not require a high complexity model. Even the simplest model with three Convolutional Layers is working fine.

Kernel Size	3		5		7	
#C.Layers	Train	Valid	Train	Valid	Train	Valid
1	0.343	0.300	0.363	0.300	0.346	0.290
2	0.322	0.282	0.305	0.272	0.221	0.271
3	0.155	**0.210**	0.086	**0.204**	0.153	0.198
4	0.149	0.222	0.139	**0.219**	0.134	0.210

Table 6.6: Results with top 3 performing models

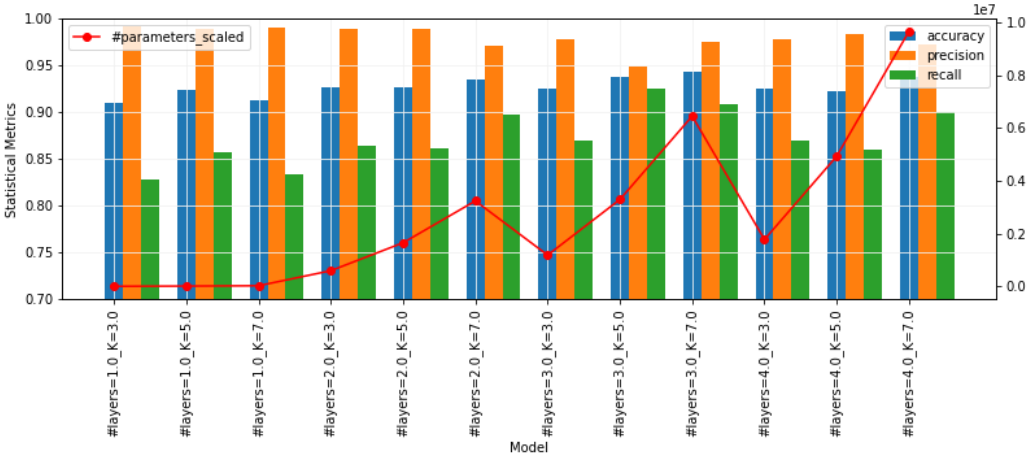


Figure 6.11: A statistical comparison of different CNN models

The models with three Convolutional layers with Kernel Size 3 and Kernel Size 5 performed almost equally well, with the model having Kernel Size

5 performing a bit better in terms of the validation error. However Statistical results like Accuracy, Precision, and Recall are better when Kernel Size is 5 which is evident in Fig. 6.11. According to literature, **Stacking smaller convolutional layers is always lighter than having bigger ones** [17]. Nonetheless, one important consideration which includes the number of output channels after every Convolutional layer is still missing. Therefore, we will increase the number of output channels for the model with Kernel Size 5.

Table 6.7 shows the comparison between different versions of the model with varying output channels in every convolutional layer. The models with the number of Convolutional layers equal 3, and Kernel Size of 5 was trained on models with a different number of output channels in every Convolutional layer.

In the model with Kernel Size 5, the model with the number of output channels being [128,128,128],[128,256,512], and [256,512,1024] in the three consecutive layers tends to work best. Nevertheless, the model with channels [128,128,128] was chosen because of its simplicity and high recall. High Recall is very important for lighting scenarios. **Therefore, the best choice for CNN was the model with the number of Convolutional Layers equal 3 and Kernel size equals 5 with the number of output channels equals 128 in all the layers.**

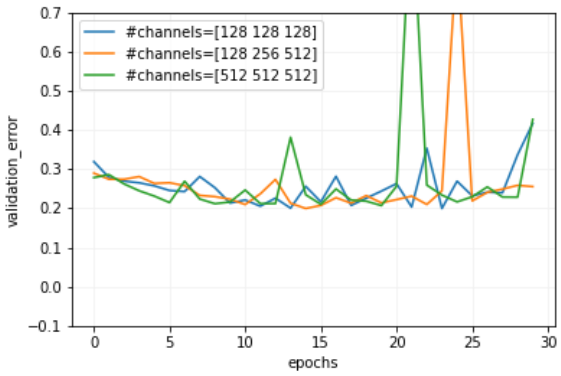


Figure 6.12: Validation Loss for best configuration for Output channels

Hyperparameter selection for CNN with Ray Tune

Similar to section 6.2,to experiment more with the hyperparameter space for CNN, **50** different versions of the CNN model were evaluated using the

Channels	Validation Loss	Accuracy	Precision	Recall	F1 Score
{64,64,64}	0.220	0.923	0.98	0.864	0.920
{64,128,128}	0.214	0.935	0.975	0.893	0.932
{64,128,256}	0.214	0.928	0.980	0.875	0.924
{128,128,128}	**0.199**	0.937	0.970	0.903	0.935
{128,256,512}	**0.199**	0.938	0.981	0.894	0.935
{256,256,256}	0.225	0.934	0.955	0.912	0.933
{256,512,1024}	**0.199**	0.926	0.982	0.869	0.922
{512,512,512}	0.207	0.923	0.982	0.861	0.918

Table 6.7: Different number of output channels in the consecutive laeyrs

ASHA algorithm. The metric used for early stopping is the loss function. The hyperparameter space experimented is discussed in the previous Chapter 5.

Table 6.8 shows the performance of the Top 10 performing models. Fig. 6.13 demonstrates the Validation loss of these models with epochs. The common thing among most of the Top performing models is the Kernel Size of 5. Looking more closely the Model#1 represented by blue color, despite giving the best validation loss ends up being early stopped by ASHA. This means this model was less promising. Model#2 represented by orange color does not early stops and gives good validation loss. Model#3 also does not early stops, but the performing over epochs is much inferior to the Model#2.

Name	Loss	α	B	Epochs	Kernel Size	#Layers	Optim Fn.
Model#1	**0.192**	0.032	64	9	5	3	Adagrad
Model#2	**0.202**	0.0014	128	29	5	3	ASGD
Model#3	**0.207**	0.002	128	22	5	4	SGD
Model#4	0.209	0.019	128	9	5	3	Adam
Model#5	0.216	0.0025	128	25	5	4	ASGD
Model#6	0.229	0.002	64	3	3	3	Adam
Model#7	0.242	0.002	128	26	3	3	SGD
Model#8	0.261	0.07	64	3	5	3	Adagrad
Model#9	0.276	0.0015	128	25	3	2	SGD
Model#10	0.283	0.034	128	9	5	2	SGD

Table 6.8: Hyperparameters corresponding to Top 10 performing models with ASHA for CNN

Even though the models obtained using ASHA Scheduler gave decent

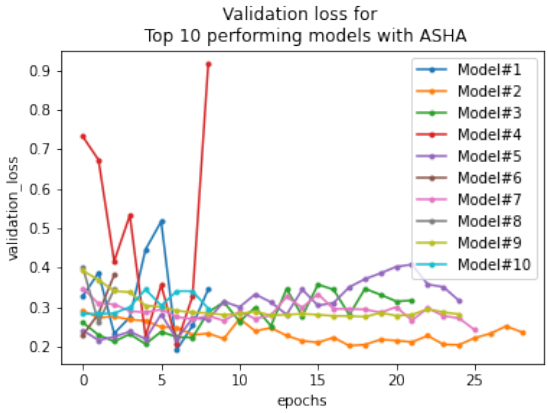


Figure 6.13: Validation loss for Top 10 performing models and early termination with ASHA

results, but the Validation losses obtained by the exhaustive search were still better. It can be seen by comparing the Table 6.7 and Table 6.8. *Therefore, the model which performed best has 3 convolutional layers with Kernel Size of 5 and feature map of [128,128,128] in the three consecutive layers and 256 neurons in the last linear layer. The model is trained with ASGD as the optimization function, BCELoss as the loss function with a learning rate 0f 0.001.*

Threshold Selection

One main consideration which was left in the discussion above is the threshold of the prediction itself. Fig. 6.14a shows the plot of TPR vs FPR for the best model selected. Fig. 6.14b depicts the Probability threshold vs $|TPR(p) + FPR(p) - 1|$. It is very much evident that the best probability threshold is close to **0.43**. Fig. 6.14c demonstrates how the Statistical metrics vary by varying the threshold probability. It is clear that beyond the threshold of 0.43, the Precision is unaffected, however, Recall drops off drastically. For a threshold less than 0.4, there is a trade-off between Precision and Recall, but a higher Recall is beneficial as it reduces the number of false-off events. Table 6.9 shows the comparison between the best models with three different thresholds. Fig. 6.15 demonstrates the confusion matrix of the validation data set on the selected model with a threshold of 0.48.

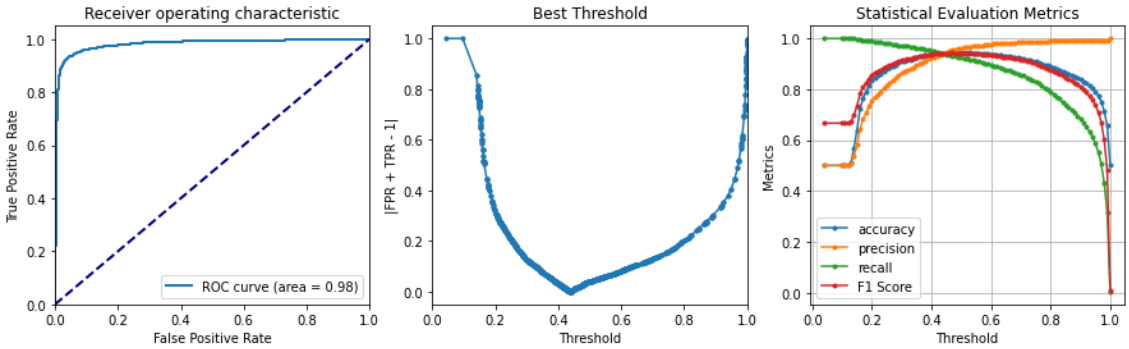


Figure 6.14: Optimal thresholds for CNN

Network	Accuracy	Precision	Recall	F1-Score
FNN(p =*0.43*)	0.941	0.937	0.943	0.941
FNN(**p =0.48**)	**0.943**	**0.956**	**0.928**	**0.942**
FNN(p =*0.5*)	0.942	0.962	0.923	0.941

Table 6.9: Metric values at optimal thresholds

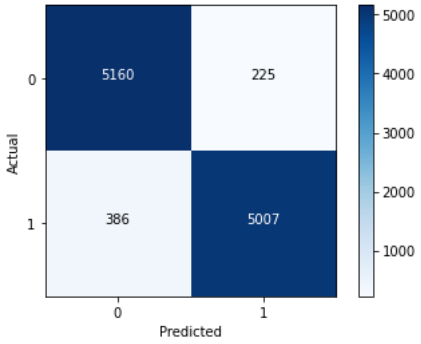


Figure 6.15: Confusion matrix with optimal threshold

6.4 Evaluation on Test dataset for selected models

The last step is to evaluate the performance of the best trained models on the new and unseen train dataset. The thesis targeted on creating a generalized model for human detection. To verify if the models are generalized in nature, a completely different dataset was collected from different locations. To verify

the hypothesis which states that the models are general in nature, optimized models are finally evaluated using the test data. The Baseline model was evaluated using just statistical metrics, FNN and CNN were evaluated using both statistical and empirical metrics.

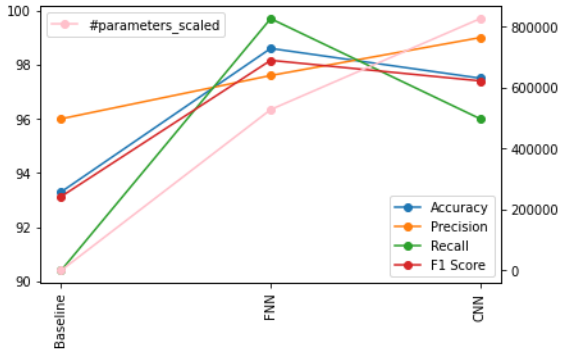
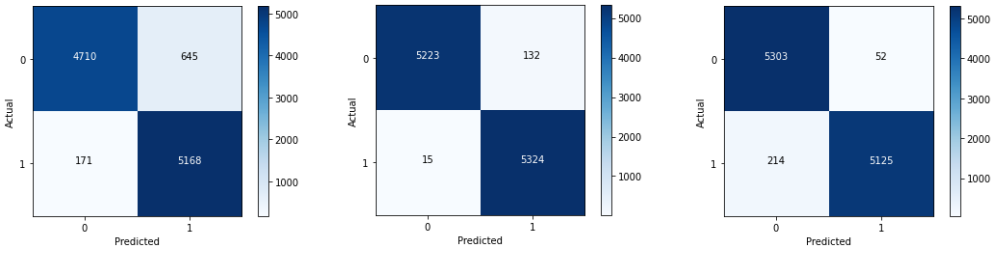


Figure 6.16: Comparison of the models

Model	Accuracy	Precision	Recall	F1 Score
Baseline	93.3%	96.0%	90.4%	93.13%
FNN	98.6%	97.6%	99.7%	98.6%
CNN	97.5%	99%	96%	97.4%

Table 6.10: Model performance on Test data



(a) Baseline (b) FNN (c) CNN

Figure 6.17: Confusion Matrices obtained by different networks on the Test data

Table 6.10 and Fig. 6.16 shows the performance of three different types of models. Clearly, FNN models worked out to be best for the test dataset

for both statistical and empirical metrics. It is very clear from the results that FNN models gave the best results. When tested on the test dataset, the models predicted the results with an accuracy of 98.6%, the precision of 97.6%, recall of 99.7% and F1 Score of 98.6%. Even the number of parameters that are needed to be trained over the NN is less for FNN as compared to CNN.

Chapter 7

Discussion and Future Work

In this thesis, the concepts of FMCW RADAR and Deep Learning are introduced. The background and the literature related to the topic are presented. The research has been focused on collecting data from FMCW RADAR and then applying several ML algorithms. To create the labels for the data RGB camera has been used for object detection by exploiting computer vision algorithms. The research was focused on developing a generalized Machine Learning model, i.e., the model should predict quality results irrespective of the space from where the data is obtained. As a result of this thesis, a Deep Learning model is proposed which tries to predict human presence with the best possible statistical metrics.

7.1 Discussion

Having the occupancy information of an indoor environment can help optimize resources utilization in smart buildings and eventually reduce the total energy consumption. In this thesis, a novel solution for presence detection has been proposed using FMCW RADAR. It is observed that presence detection with FMCW RADAR is a promising technique for indoor presence detection and gave much more effective results than already existing solutions.

Traditionally PIR sensors have been used for human presence detection. These sensors are thermal-based sensors that result in a change in voltage when a human is present. However these sensors are prone to a lot of false negatives, i.e., for the scenarios when the human is stationary, PIRs are subjected to inaccuracies. On the other hand, because of the timer associated with these sensors, the luminaries stay on for this duration when there is no track of movement. This eventually leads to electricity wastage. This is why special attention was paid to RADAR data collection. A large portion of the

data has gathered from a scenario where human was stationary. Nonetheless, ML models build on this data performed with high accuracy primarily because RADAR is quick in detecting micro-movements. Intuitively, a human body always generates a micro-movement, and henceforth, a RADAR can quickly detect them.

Among the tested algorithms, the FNN model with depth 4 ad width 256 resulted in the highest accuracy of **98.6%** on the test set. The algorithm used 528129 parameters to be trained. In our experiment, CNN models resulted in comparable prediction accuracy. However, the number of parameters here is larger than FNN. It is worth noticing that the RADAR data is prone to a lot of cluttering, scattering, reflections in an indoor environment. This is the reason for inaccurate detection sometimes. Also for labeling, we are using an RGB camera. It might be possible that the combination of YOLOV3 and motion detection can lead to false labeling. The labeling method can be prone to fallacies as well.

We can not deny the fact that this method comes with a lot of additional cost overheads. RADAR is of course more expensive than the traditional sensors. Also, the process discussed in this thesis is subjected to much higher computational power and thus more costs for computational devices. However, if this RADAR data is coupled with other use cases in an indoor environment like HVAC control, human well being, human activity tracking(based on the type of the building), it is very much possible to balance out the costs of this method with the prolificacy of the results which can be obtained by just one device. RADAR can definitely be an alternative for not only multiple PIR sensors but also for other numerous indoor sensors. The thesis was based on a comparatively easier problem at hand. However, it is worth mentioning that this FMCW RADAR deserves more exploitation to solve other challenging problems, some of which are mentioned in the next section.

7.2 Future Work

The main contribution of this thesis is the novel solution for binary human presence detection with FMCW RADAR. To extend the results, datasets with different use cases need to be collected and experimented with. One major use case associated with the research is people counting. People counting is already subjected to a multitude of research activities. The idea is to convert the RADAR data into point clouds data and then into the image. The labels can be produced using the same method defined in this thesis. The second important use case is zone detection, i.e., in which zone the hu-

man is present. This is an important research area to consider in the future as for indoor lighting control, the position of the human is of utmost value. RADAR is a substitute for multiple PIRs and if so, there should be an efficient model to predict the location of the person in order to turn on/off lights in a particular region. For example, in a parking lot, if one RADAR is installed, there needs to be an effective algorithm to turn on/off lights with the movement of the vehicle. Another use case that is associated with the RADAR is distance limiting based on the architectural premise. It is well known that RADAR is prone to a lot of cluttering and the RADAR signals can easily surpass the non-concrete walls. One efficient use case is to build an algorithm to limit the boundaries of the signals. For an instance, if the RADAR is installed in one meeting room, it needs to be unaffected by the activities going on in the adjacent room.

FMCW RADAR has the capability to give higher quality data which can give higher prediction accuracy results. The purpose of this thesis work was to demonstrate this capability. Different approaches and algorithms can be further explored to exploit this high informational FMCW RADAR data.

Chapter 8

Conclusion

Deep Learning is a powerful tool that has the potential to solve real-world problems and complex functions. One such problem to optimizing the human detection problem, particularly for lighting control in an indoor environment is focused upon in this thesis. Efficient systems are desirable and have been the subject of a multitude of research for more than two decades now.

Energy efficiency is an important aspect of a sustainable lifestyle by reducing carbon footprints by minimizing the electricity production. The objective of the study was to develop a model with the ability to predict human presence. Traditional lighting control systems react to occupancy changes and are extremely prone to inefficient energy consumption patterns. By developing machine learning models that predict human presence and absence with much higher confidence than traditional PIRs can lead to significant energy savings. Intelligent systems that use powerful RADAR signals can make real-time decisions, which can be translated into efficient building automation systems. It can be stated with confidence that the stated methodology in this thesis and its outcomes reinforce the notion that decision-making based on FMCW RADAR has demonstrable potential. Furthermore, continued research is encouraged to improve upon the accuracy and explore other domains correlated with human presence.

Bibliography

[1] Camera module v2. https://www.raspberrypi.org/products/camera-module-v2/.

[2] Energy performance of buildings directive. https://ec.europa.eu/energy/topics/energy-efficiency/energy-efficient-buildings/energy-performance-buildings-directive_en.

[3] Torch.utils.data - pytorch 1.9.1 documentation. https://pytorch.org/docs/stable/data.html#torch.utils.data.DataLoader.

[4] EUR-Lex - 32012L0027 - EN - EUR-Lex. https://eur-lex.europa.eu/eli/dir/2012/27/oj, Jul 2021.

[5] Radar Systems Tutorial - Tutorialspoint. https://www.tutorialspoint.com/radar_systems/index.htm, Jul 2021.

[6] Ray Tune: Hyperparameter Optimization Framework — Ray 0.4.0 documentation. https://docs.ray.io/en/ray-0.4.0/tune.html, Jan 2021.

[7] torchvision.transforms — Torchvision 0.10.0 documentation. https://pytorch.org/vision/stable/transforms.html, Jul 2021.

[8] Ansanay-Alex, G. Estimating occupancy using indoor carbon dioxide concentrations only in an office building: a method and qualitative assessment. In *REHVA World Congress on Energy Efficient, Smart and Healthy Buildings (CLIMA)* (Jun 2013), pp. 1–8.

[9] Arai, I. Life-detection radar for rescue purpose. *IEICE Technical Report*, SANE99-100 (Apr 2000).

[10] Arras, K. O., Mozos, O. M., and Burgard, W. Using boosted features for the detection of people in 2d range data. In *Proceedings 2007 IEEE International Conference on Robotics and Automation* (Apr 2007), IEEE, pp. 3402–3407.

[11] Bengio, Y. Practical recommendations for gradient-based training of deep architectures. In *Neural Networks: Tricks of the Trade*. Springer, 2012, pp. 437–478.

[12] Blackman, R. B., and Tukey, J. W. The measurement of power spectra from the point of view of communications engineering- part 1. *Bell System Technical Journal 37*, 1 (1958), 185–282.

[13] Bracewell, R. N., and Bracewell, R. N. *The Fourier transform and its applications*, vol. 31999. McGraw-Hill New York, 1986.

[14] Brownlee, J. How to control the stability of training neural networks with the batch size. `shorturl.at/dxyRY`.

[15] Buades, A., Coll, B., and Morel, J.-M. Non-Local Means Denoising. *Image Processing On Line 1* (2011), 208–212. `https://doi.org/10.5201/ipol.2011.bcm_nlm`.

[16] Calì, D., Matthes, P., Huchtemann, K., Streblow, R., and Müller, D. Co2 based occupancy detection algorithm: Experimental analysis and validation for office and residential buildings. *Building and Environment 86* (2015), 39–49.

[17] Chazareix, A. About Convolutional Layer and Convolution Kernel. `https://www.sicara.ai/blog/2019-10-31-convolutional-layer-convolution-kernel`, Dec 2020.

[18] Dodier, R. H., Henze, G. P., Tiller, D. K., and Guo, X. Building occupancy detection through sensor belief networks. *Energy and Buildings 38*, 9 (2006), 1033–1043.

[19] Dubois, M.-C., and Blomsterberg, Å. Energy saving potential and strategies for electric lighting in future north european, low energy office buildings: A literature review. *Energy and buildings 43*, 10 (2011), 2572–2582.

[20] Dudani, S. A. The distance-weighted k-nearest-neighbor rule. *IEEE Transactions on Systems, Man, and Cybernetics*, 4 (1976), 325–327.

[21] Edelsbrunner, H., and Harer, J. *Computational Topology: An Introduction*. American Mathematical Soc., 2010.

[22] Ekwevugbe, T., Brown, N., and Fan, D. A design model for building occupancy detection using sensor fusion. In *2012 6th IEEE International Conference on Digital Ecosystems and Technologies (DEST)* (2012), IEEE, pp. 1–6.

[23] Falconer, D. G., Ficklin, R. W., and Konolige, K. G. Robot-mounted through-wall radar for detecting, locating, and identifying building occupants. In *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065)* (2000), vol. 2, IEEE, pp. 1868–1875.

[24] Felinger, A. *Data Analysis and Signal Processing in Chromatography.* Elsevier, 1998.

[25] Geisheimer, J. L., Greneker III, E. F., and Marshall, W. S. High-resolution doppler model of the human gait. In *Radar Sensor Technology and Data Visualization* (2002), vol. 4744, International Society for Optics and Photonics, pp. 8–18.

[26] Goodfellow, I., Bengio, Y., and Courville, A. *Deep Learning.* MIT Press, 2016. `http://www.deeplearningbook.org`.

[27] Gürbüz, S. Z., Melvin, W. L., and Williams, D. B. Detection and identification of human targets in radar data. In *Signal Processing, Sensor Fusion, and Target Recognition XVI* (2007), vol. 6567, International Society for Optics and Photonics, p. 65670I.

[28] Helvar. Circadian lighting supporting wellbeing. `https://helvar.com/wellbeing`, Jul 2021.

[29] Jaderberg, M., Dalibard, V., Osindero, S., Czarnecki, W. M., Donahue, J., Razavi, A., Vinyals, O., Green, T., Dunning, I., Simonyan, K., et al. Population based training of neural networks. *arXiv preprint arXiv:1711.09846* (2017).

[30] Jolliffe, I. Principal component analysis. *Encyclopedia of Statistics in Behavioral Science* (2005).

[31] Kim, S. H., Moon, H. J., and Yoon, Y. R. Improved occupancy detection accuracy using pir and door sensors for a smart thermostat. In *Proceedings of the 15th IBPSA Conference, San Francisco, CA, USA* (2017), pp. 2753–2758.

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

[33] Kotsiantis, S. B., Zaharakis, I., Pintelas, P., et al. Supervised machine learning: A review of classification techniques. *Emerging Artificial Intelligence Applications in Computer Engineering 160*, 1 (2007), 3–24.

[34] Labeodan, T., Aduda, K., Zeiler, W., and Hoving, F. Experimental evaluation of the performance of chair sensors in an office space for occupancy detection and occupancy-driven control. *Energy and Buildings 111* (2016), 195–206.

[35] Lenail, A. Nn svg. `https://alexlenail.me/NN-SVG/`.

[36] Li, C., Xiao, Y., and Lin, J. Experiment and spectral analysis of a low-power *ka*-band heartbeat detector measuring from four sides of a human body. *IEEE Transactions on Microwave Theory and Techniques 54*, 12 (2006), 4464–4471.

[37] Li, L., Jamieson, K. G., DeSalvo, G., Rostamizadeh, A., and Talwalkar, A. Efficient hyperparameter optimization and infinitely many armed bandits. *CoRR, abs/1603.06560* (2016).

[38] Li, L., Jamieson, K. G., Rostamizadeh, A., Gonina, E., Hardt, M., Recht, B., and Talwalkar, A. Massively parallel hyperparameter tuning. *CoRR, abs/1810.05934* (2018).

[39] Liaw, R. Cutting edge hyperparameter tuning with Ray Tune - riselab - Medium. *Medium* (Mar 2020).

[40] Light, R. A. Mosquitto: server and client implementation of the mqtt protocol. *Journal of Open Source Software 2*, 13 (2017), 265.

[41] Manzoor, F., Linton, D., and Loughlin, M. Occupancy monitoring using passive rfid technology for efficient building lighting control. In *2012 Fourth International EURASIP Workshop on RFID Technology* (2012), IEEE, pp. 83–88.

[42] Mikkilineni, A. K., Dong, J., Kuruganti, T., and Fugate, D. A novel occupancy detection solution using low-power ir-fpa based wireless occupancy sensor. *Energy and Buildings 192* (2019), 63–74.

[43] Mitchell, T. *Machine Learning*. McGraw-Hill International Editions. McGraw-Hill, 1997.

[44] Mitchell, T. M. Machine learning and data mining. *Communications of the ACM 42*, 11 (1999), 30–36.

[45] Murphy, K. P. Naive bayes classifiers. `https://www.ic.unicamp.br/~rocha/teaching/2011s2/mc906/aulas/naive-bayes.pdf`, 2006.

[46] Myles, A. J., Feudale, R. N., Liu, Y., Woody, N. A., and Brown, S. D. An introduction to decision tree modeling. *Journal of Chemometrics: A Journal of the Chemometrics Society 18*, 6 (2004), 275–285.

[47] NIEUWELING, C. Residential lighting. `https://ec.europa.eu/jrc/en/energy-efficiency/products/residential-lighting#:~:text=IntheEuropeanUnion,the,numberoflampsperhome.`, Nov 2016.

[48] Nilsson, J., and Hassbring, L. Machine learning for fmcw radar interference mitigation. Student Paper.

[49] Noble, W. S. What is a support vector machine? *Nature Biotechnology 24*, 12 (2006), 1565–1567.

[50] Nwankpa, C., Ijomah, W., Gachagan, A., and Marshall, S. Activation functions: Comparison of trends in practice and research for deep learning. *arXiv preprint arXiv:1811.03378* (2018).

[51] Open Data Science, O. Overview of the yolo object detection algorithm. `https://medium.com/@ODSC/overview-of-the-yolo-object-detection-algorithm-7b52a745d3e0`, Sep 2018.

[52] Oppenheim, A., and Schafer, R. //discrete-time signal processing. upper saddle river (n. *J.): Pearson Higher Education, Inc., 2010. 1108 p* (2010), 1108.

[53] Rao, S. Introduction to mmwave radar sensing: Fmcw radars. `https://training.ti.com/node/1139153`, journal=TI Training, Apr 2017.

[54] Redmon, J. Yolo: Real-time object detection. `https://pjreddie.com/darknet/yolo/`.

[55] Redmon, J., and Farhadi, A. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767* (Apr 2018).

[56] Rothe, H. Ask the community: presence detection at home. `https://heikorothe.com/blog/2019_07_23_presence_detection_survey/`, Jul 2019.

[57] Santamouris, M., and Dascalaki, E. Passive retrofitting of office buildings to improve their energy performance and indoor environment: the office project. *Building and Environment 37*, 6 (2002), 575–578.

[58] Sarkar, A., Fairchild, M., and Salvaggio, C. Integrated daylight harvesting and occupancy detection using digital imaging. In *Sensors, Cameras, and Systems for Industrial/Scientific Applications IX* (2008), vol. 6816, International Society for Optics and Photonics, p. 68160F.

[59] Shafkat, I. Intuitively understanding convolutions for deep learning-towards data science. `https://towardsdatascience.com/intuitively-understanding-convolutions-for-deep-learning-1f6f42faee1`, Jun 2018.

[60] Shih, H.-C. A robust occupancy detection and tracking algorithm for the automatic monitoring and commissioning of a building. *Energy and Buildings 77* (2014), 270–280.

[61] Skolnik, M. I. Introduction to radar. *Radar handbook 2* (1962), 21.

[62] Sutton, R. S., and Barto, A. G. *Reinforcement learning: An introduction*. MIT press, 2018.

[63] Teixeira, T., Dublon, G., and Savvides, A. A survey of human-sensing: Methods for detecting presence, count, location, track, and identity. *ACM Computing Surveys 5*, 1 (2010), 59–69.

[64] Touger, E. What's the difference between artificial intelligence, machine learning, and deep learning? `shorturl.at/euTU4`, Jun 2020.

[65] Yadav, J., and Sharma, M. A review of k-mean algorithm. *International Journal of Engineering Trends and Technology (IJETT) 4*, 7 (2013), 2972–2976.

[66] Zhou, Q., Liu, J., Host-Madsen, A., Boric-Lubecke, O., and Lubecke, V. Detection of multiple heartbeats using doppler radar. In *2006 IEEE International Conference on Acoustics Speech and Signal Processing Proceedings* (2006), vol. 2, IEEE, pp. II–II.

Appendix A

Appendix

A.1 Fourier Transformations

According to [13] "Fourier transform (FT) is a mathematical transform that decomposes functions depending on space or time into functions depending on spatial or temporal frequency."

Specific to FMCW, Fourier transform is used to convert a time domain signal into the frequency domain. Consider a frequency signal in the form

$$A_t = A \sin [2\pi f t + \phi]. \tag{A.1}$$

Suppose $A = 2, t = 1sec, \phi = 0$ and $f = 4$.
Therefore, the Eq. (A.1) becomes

$$A_1 = 2 \sin [8\pi]. \tag{A.2}$$

The Fourier transformation of this signal, returns a frequency-amplitude domain signal, with a peak at **4** as the frequency of the signal was 4. It can be seen in Fig. A.1.

However, in a real-time scenario, signals are not continuous, rather discrete. Therefore for discrete signals, Discrete Fourier transform (DFT) is commonly used. The resolution limit is determined by the number of input samples. The output of the DFT has the same size as the input. For a device like RADAR where the frequency of the signal is usually in GHz, the sampling index is huge, therefore the number of outputs of DFT will be huge. Increasing the number of inputs, increased the computation power by $O[N^2]$.

Therefore, to make the computations less complex, the number of samples is chosen as a power of 2^N which enables to use Fast Fourier transform (FFT) instead of DFT. The complexity of FFT is $O[N \log N]$ instead of $O[N^2]$. FFT can be used either by increasing the sampling rate or by appending zeros at

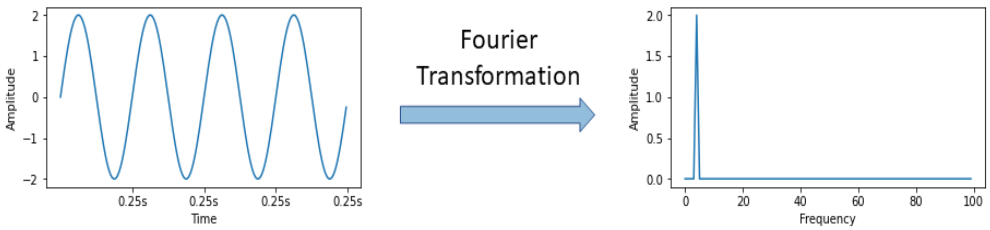


Figure A.1: Fourier transformation of a constant frequency signal with $A = 2$ and $f = 4$.

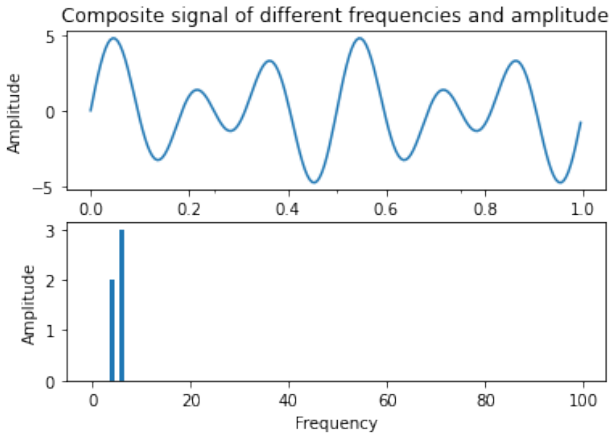


Figure A.2: Fourier transformation of a composite signal

the end before performing FFT. Zero padding makes the resolution between two consecutive bins easier.

It is worth mentioning that the resolution of the FFT increases with the length of the observation period. In general, an observation window if T can separate frequency components that are separated by more than $\frac{1}{T}$ Hz [53].

In a real scenario, a signal which is a combination of different frequencies. For instance, let us imagine a composite signal as shown in the Fig. A.2. The signal is formed by the addition of two different signals with frequencies of 4 Hz and 6 Hz, and the amplitude of 2 m and 3 m. The frequencies and the amplitude of these composite signals can be easily obtained by using FFT on this signal. The plot beneath the composite signals indicates the FFT output on the composite signal.

A.2 Euler Representation of the Wave

A Euler's formula is expressed as $Ae^{j\theta}$, where A is the amplitude of the signal and θ is the initial phase of the signal. The Fourier transformation of the signals produces a complex signal, i.e., an amplitude value as well as the phaser. In Fig. A.3, waves and their corresponding Euler's representations are shown.

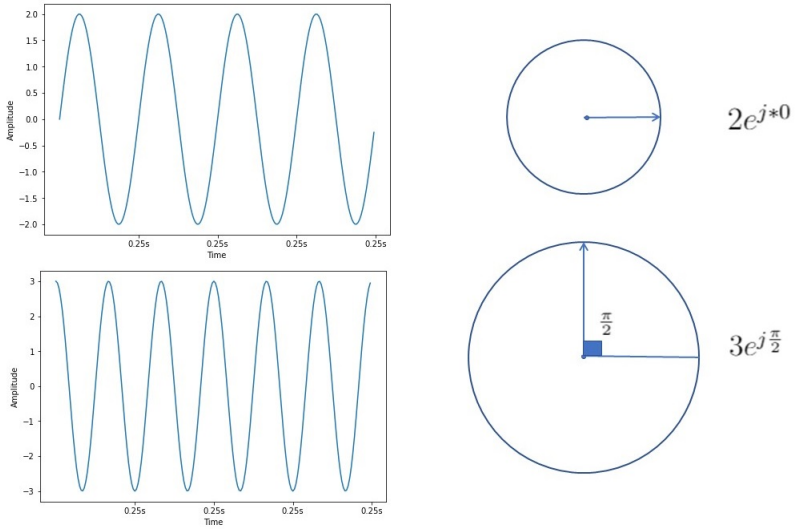


Figure A.3: Wave and its Euler representation