DRONE PROPELLER RECOGNITION THROUGH MACHINE LEARNING WITH MILLIMETER WAVE RADAR

Yüksek Lisans Tezi

Fatma ÖZÜDOĞRU

Eskişehir 2024

DRONE PROPELLER RECOGNITION THROUGH MACHINE LEARNING WITH MILLIMETER WAVE RADAR

Fatma ÖZÜDOĞRU

MASTER OF SCIENCE THESIS

Department of Electrical and Electronics Engineering

Programme in Circuits and Systems Theory

Supervisor: Prof. Dr. Tansu FİLİK

Eskişehir

Eskişehir Technical University

Institute of Graduate Programs

January 2024

FINAL APPROVAL FOR THESIS

This thesis titled DRONE PROPELLER RECOGNITION THROUGH MACHINE LEARNING WITH MILLIMETER WAVE RADAR has been prepared and submitted by Fatma Özüdoğru in partial fulfillment of the requirements in "Eskisehir Technical University Directive on Graduate Education and Examination" for the Degree of Master's in electrical and Electronics Engineering Department has been examined and approved on 23/01/2024.

Committee Members	**Title, Name and Surname**	**Signature**
Member	: Prof. Dr. Tansu FİLİK	
Member	: Asst. Prof. Dr. Efnan ŞORA GÜNAL	
Member	: Asst. Prof. Dr. Can UYSAL	

Prof. Dr. Semra KURAMA
Director of the Institute of Graduate Program

23/01/2024

SUPERVISOR APPROVAL

Master’s student Fatma ÖZÜDOĞRU, whom I supervise, has completed this thesis titled DRONE PROPELLER RECOGNITION THROUGH MACHINE LEARNING WITH MILLIMETER WAVE RADAR. According to my inspections, the work is scientifically and ethically appropriate for the student to the thesis defense exam.

Supervisor

Prof. Dr. Tansu FİLİK

ABSTRACT

DRONE PROPELLER RECOGNITION
THROUGH MACHINE LEARNING WITH MILLIMETER WAVE RADAR

Fatma ÖZÜDOĞRU

Department of Electrical and Electronics Engineering
Programme in Circuits and Systems Theory
Eskişehir Technical University, Institute of Graduate Programs, January 2024

Supervisor: Prof. Dr. Tansu FİLİK

Mini-sized Unmanned Aerial Vehicles (UAVs) are commonly called “drones”, have become ubiquitous in civilian and military applications, readily available off-the-shelf and widely utilized. Detecting and classifying these devices are essential requirements that involve the application of various technologies. The distinctive feature of these drones is their propellers. This thesis concentrates on employing millimeter-wave radar to detect the presence, rotation speed, and dimensions of these propellers.

The research explores the integration of a consumer-grade millimeter-wave Frequency Modulated Continuous Wave (FMCW) radar module, specifically the Texas Instruments IWR1843, with machine learning techniques for UAV propeller identification. Utilizing the unique capabilities of millimeter-wave radar technology, this study aims to capture and analyze intricate frequency patterns produced by UAV propellers through machine learning models.

Experimental validation in controlled environments demonstrates the potential of the millimeter-wave FMCW radar module, combined with machine learning algorithms, to accurately detect and identify UAVs based on propeller characteristics, achieving up to 92% accuracy in presence, propeller size, and throttle level. Additionally, this thesis highlights the superiority of the proposed ensemble neural network method over a multi-output single Multi Layer Perceptron (MLP) model, as demonstrated through experiments.

Keywords: UAV detection, Propeller identification, mmWave doppler radar, Ensemble neural network, KNN classifier, MLP regressor, IWR1843.

ÖZET

MİLİMETRE DALGA RADARI KULLANILARAK MAKİNE ÖĞRENMESİ İLE DRONE PERVANESİ TANIMA VE KİMLİKLENDİRME

Fatma ÖZÜDOĞRU

Elektrik Elektronik Mühendisliği Anabilim Dalı
Devreler ve Sistemler Teori Bilim Dalı
Eskişehir Teknik Üniversitesi, Lisansüstü Eğitim Enstitüsü, Ocak 2024

Danışman: Prof. Dr. Tansu FİLİK

Drone olarak da adlandırılan mini boyutlu insansız hava araçları (IHA), uçuşa hazır olarak kolaylıkla satın alınabilmektedir. Çeşitli teknolojik uygulama alanlarında, bu cihazların tespiti, takibi ve sınıflandırılması yüksek önem arz etmektedir. Sinyal işleme bağlamında düşünüldüğünde, Drone'ların en ayırt edici özelliği pervaneleridir. Bu tez, IHA'ların en önemli parçalarından biri olan pervanelerinin milimetre dalga radarını kullanarak çeşitli makine öğrenmesi metodları ile varlığını, dönüş hızını ve boyutunu tespit etme konusundaki çalışmaları içermektedir.

Bu araştırma, Texas Instruments'ın tüketici sınıfı milimetre dalga Frekans Modülasyonlu Sürekli Dalga (FMCW) radar modülünün (IWR1843), İHA pervanesinin tanınması ve kimliklendirilmesi için makine öğrenme teknikleriyle birlikte kullanımını araştırmaktır. Milimetre dalgalı radar teknolojisinin benzersiz yeteneklerinden yararlanan bu çalışma, makine öğrenimi modellerini kullanarak İHA pervaneleri tarafından üretilen karmaşık frekans modellerini yakalamayı ve analiz etmeyi amaçlamaktadır.

Kontrollü ortamlarda yapılan deneysel doğrulama, makine öğrenimi algoritmalarıyla birleştirilmiş milimetre dalgalı FMCW radar modülünün, İHA pervanesinin varlığı, boyutu ve gaz seviyesinin yakınsanması gibi pervane özelliklerine göre %92'ye kadar doğru bir şekilde tespit etme ve tanımlama potansiyeline sahiptir. Ayrıca bu tez sırasında önerilen çoklu sinir ağı yönteminin birden fazla çıkışlı tek MLP modeline üstünlüğü, yapılan deneylerle gösterilmiştir.

Anahtar Sözcükler: İHA tespiti, Pervane tanıma ve kimliklendirme, Milimetre dalga doppler radarı, Birleştirilmiş çoklu sinir ağı, KNN sınıflandırma, MLP algılayıcı, IWR1843.

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to my advisor, Prof. Dr. Tansu FİLİK, whose guidance, support, and insightful feedback have been invaluable throughout the completion of this thesis.

I would like to thank Asst. Prof. Dr. Can UYSAL and Asst. Prof. Dr. Efnan ŞORA GÜNAL for serving in my committee.

I am also deeply thankful to Mehmet AKTAŞ, Metin ÇAM and Asst. Prof. Dr. Semiha TÜRKAY for their support, encouragement and constructive criticism and love.

I am grateful to my family for their continuous encouragement, love, understanding, and belief in my abilities.

Last but not least, I would like to express my deepest gratitude to Mustafa GÖKÇE for his help in overcoming obstacles and persevering in my academic studies, for his encouragement, love and understanding.

To all those mentioned above and to everyone else who played a part, whether big or small, in this academic journey, I extend my heartfelt thanks.

Fatma ÖZÜDOĞRU

Eskişehir, January 2024

23/01/2024

STATEMENT OF COMPLIANCE WITH ETHICAL PRINCIPLES AND RULES

I hereby truthfully declare that this thesis is an original work prepared by me; that I have behaved in accordance with the scientific ethical principles and rules throughout the stages of preparation, data collection, analysis and presentation of my work; that I have cited the sources of all the data and information that could be obtained within the scope of this study, and included these sources in the references section; and that this study has been scanned for plagiarism with “scientific plagiarism detection program” used by Eskişehir Technical University, and that “it does not have any plagiarism” whatsoever. I also declare that, if a case contrary to my declaration is detected in my work at any time, I hereby express my consent to all the ethical and legal consequences that are involved.

Fatma ÖZÜDOĞRU

CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF IMAGES

GLOSSARY OF SYMBOLS AND ABBREVIATIONS

FMCW	: Frequency Modulated Continuous Wave
UAV	: Unmanned Aerial Vehicle
mmWave	: Millimeter Wave
DVB-T2	: Digital Video Broadcasting-Second Generation Terrestrial
CFAR	: Constant False Alarm Rate
GSM	: Global System for Mobile Communication
MIMO	: Multiple Input, Multiple Output
YOLO	: You Only Look Once
CNN	: Convolutional Neural Network
SNR	: Signal-to-Noise Ratio
CA-CFAR	: Cell Averaging Constant False Alarm Rate
OS-CFAR	: Ordered Statistics Constant False Alarm Rate
PDF	: Probability Density Function
CDF	: Cumulative Distribution Function
MLE	: Maximum Likelihood Estimation
SSR	: Secondary surveillance radar
CW	: Continuous Wave
MTI	: Moving Target Indicator
SAR	: Synthetic Aperture Radar
OTH	: Over-the-Horizon Radar
GPR	: Ground penetrating radar
THz	: Terahertz
IF	: Intermediate Frequency
FFT	: Fast Fourier Transform
AoA	: Angle of Arrival
AI	: Artificial intelligence
FNN	: Feedforward Neural Networks
RNN	: Recurrent Neural Networks
GAN	: Generative Adversarial Networks
ISAR	: Inverse Synthetic Aperture Radar
SVM	: Support Vector Machine

KNN : K-Nearest Neighbors
MLP : Multilayer Perceptron
PCA : Principal Component Analysis
GBM : Gradient Boosting Machines
2D : Two Dimensional
RC : Radio Control
ESC : Electronic Speed Controller
ADAS : Advanced Driver Assistance Systems
RF : Radio Frequency
PLL : Phase Lock Loop
ADC : Analog-to-Digital Converters
TX : Transmit
RX : Receive
Li-Po : Lithium Polymer
SDK : Software Development Kit
USB : Universal Serial Bus
OS : Operating System
CSV : Comma Separated Values
SVC : Support Vector Classification
TP : True Positive
TN : True Negative
FP : False Positive
FN : False Negative
MSE : Mean Squared Error
MAE : Mean Absolute Error
RCS : Radar Cross Section

1. INTRODUCTION

1.1. Motivation of The Millimeter Wave Radar

Millimeter Wave (mmWave) is one of many types of radar technology that is equipped with short wavelength electromagnetic waves. Radar systems simply send electromagnetic signals along a path and receive that is reflected. These reflections are due to objects alongside that path. The received reflected signals may carry information about the distance, range, velocity, shape, and bearing of objects with respect to the transmitting path.

As the name implies, these electromagnetic waves are in millimeter range. This is relatively a short wavelength in the spectrum and is considered to introduce some significant advantages and improvements in radar technology.

Since the wavelength is small, the transmitting and receiving antennas are really small. And also because of this short wavelength, it can differentiate small movements (low speeds and acceleration, small objects). In this thesis, the IWR1843 module which is a single-chip 76-GHz to 81-GHz industrial mmWave radar sensor evaluation module is used. The wavelength of this module is between 3.701 millimeters and 3.945 millimeters which provides high resolution range Doppler data than can be used to detect centimeter level small movements.

A typical mmWave radar system contains transmitting and receiving antennas, analog to digital converters, microcontrollers and digital signal processors. The mmWave radar module used to implement the application in this thesis uses frequency modulated continuous wave (FMCW). Traditional radar systems send short time pulse signals along a path periodically and capture the reflected signals. The FMCW method sends modulated signals continuously and with this way, it can measure velocity, range and angle simultaneously.

1.2. Thesis Problem Statement

The drone industry has seen remarkable growth and diversification in recent years, expanding into various sectors and applications. Drones are extensively used in industries such as agriculture, construction, real estate, cinematography, logistics, and infrastructure inspection. Ongoing technological innovations have led to the development of more sophisticated drones. This includes improvements in battery life,

flight time, range, payload capacity, and autonomous capabilities. Drones have gained importance in security and defense too, both for surveillance purposes and in military operations. This has led to the development of specialized drones for defense applications.

Detecting and monitoring drones has become increasingly important. Drones flying in unauthorized areas or close to manned aircraft pose serious safety risks. Drones can be used for malicious purposes such as carrying explosives, conducting surveillance for illicit activities, or breaching secure areas. Unauthorized drones equipped with cameras can invade individuals' privacy by capturing images or videos without consent. Many regions have strict regulations governing drone flight, including no-fly zones around airports or government buildings. Drones flying near critical infrastructure, such as power plants or military installations, can pose a threat. Large gatherings or events are susceptible to security threats from drones flying over.

Since the drone industry evolves really quickly, it becomes harder and harder to detect and track (monitor) drones in the airspace. Drones come in various sizes, including small ones that can be challenging to detect visually or using traditional radar systems. Their ability to fly at low altitudes and maneuver swiftly makes them harder to track. Some drones are designed with stealth capabilities, such as low radar cross-section or noise reduction techniques, making them harder to detect using conventional methods. In urban settings with various background noises, such as traffic or other machinery, distinguishing the sound of a drone can be challenging. Visual detection might also be hindered due to buildings, foliage, or other obstructions. Existing detection systems may have limitations in terms of their range or accuracy, especially when dealing with small drones. False positives or negatives can occur, impacting the effectiveness of detection.

The proposed method of this thesis includes detecting a drone with the Doppler data (Doppler radar recording in presence of a rotating propeller) of its propeller movements (rotation). Each drone propeller and its rotation have unique characteristics in the Doppler data. This can be seen as a noise to the naked eye, but there are many statistical and machine learning methods able to provide valuable information from this data [1]. This thesis aimed to bring a new perspective for detecting and identifying unique signatures in Doppler data of rotating propellers of drones using a cost-effective Doppler radar module.

1.3. State of Arts

There are many works and publications in the literature related with drone detection, identification and monitoring, using statistical analysis or machine learning with various techniques such as radar, image processing, acoustics and signal processing.

A study in [2] focuses on the detection and tracking of drones using phase-interferometric Doppler radar. It uses joint range-Doppler-azimuth processing to for feature extracting to identify drones. There are other works [3, 4] done about drone movement detection using FMCW radar. Radar Cross Section (RCS) is also an important parameter for detecting, identifying and monitoring drone. In another work [5], this is done with range-Doppler matrices of an FMCW radar. Another method is proposed in a work [6] to detect drones with passive coherent Digital Video Broadcasting-Second Generation Terrestrial (DVB-T2) radar using Constant False Alarm Rate (CFAR) algorithm. It has shown that the peaks of propellers are visible in range-Doppler matrix of the radar. In another work in [7], dual polarized FMCW radar was used to detect Unmanned Aerial Vehicle (UAV) with modeling it is using Weibull distribution and CFAR detector for adaptive thresholding.

In another paper [8] detecting of small drones using W-Band radar demonstrated. Classification of a drone is also discussed using micro-Doppler analysis. Using bi-static radar system for drone detection is also investigated in the literature. In a work [9], it has shown that it can be practically used for this kind of an application.

GSM passive radar is another technique to detect and track objects using signals from cell phone networks. It is successfully shown that, in a work [10], this system can be used for detecting and tracking small drones on air. Multiple Input, Multiple Output (MIMO) radar systems also capable of small drone detection and tracking, as shown in a work [11].

There are various machine learning related works in the literature about drone detection and tracking using radar and/or sensor-based system. In a work [12], drone detection and classification are done with radar and camera sensor fusion. Another work in [13] focuses on fusing mobile surveillance radar system data with computer vision. It specifically uses YOLO (You Only Look Once) network and processes the range-Doppler data by converting them to images. Convolutional Neural Network (CNN)

models can be used to detect and classify UAVs. In a work [14], CNN models trained with images, range-Doppler maps, and audio recordings.

In another paper [15], focused on utilizing radar micro-Doppler properties to distinguish between drones and birds based on their unique signatures generated by propeller blade rotation and wing beats. Experimental measurements show that phase-coherent radar systems can reliably differentiate and characterize micro-Doppler signatures of various drones and bird species, with W-band radar demonstrating higher signal-to-noise ratio (SNR) while K-band radar effectively capturing micro-motion characteristics.

The FMCW Doppler Radar Module, which will be introduced in section 4.1.1.1, used during this thesis is highly adopted in various use cases. In a work [16] it is used for human activity classification. In another work [17], it is used for fitness tracking.

1.4. Overview of The Thesis Chapters

In the second chapter, radar concepts and terahertz radars are discussed. Also, the importance of detection of mini-micro UAV is emphasized. And finally, machine learning is introduced and how it can be applied to analyze and process Doppler data to estimate propeller related data.

In the third chapter, the proposed method is introduced. It is discussed about the system overview of the proposed method, how the data acquisition and preprocessing are done, how to extract the unique features of propellers in Doppler data, and how the performance and quality of the proposed system are analyzed.

Fourth chapter includes the implementation of the proposed method. It explains how the data gathering and processing are done, which tools are used the phases, and how the environments are set.

Experimental results are presented and discussed at the fifth chapter. Also, which learning models are selected and why is explained in this chapter with reasons.

Sixth chapter concludes the proposed method, overall system architecture, experimental results, and achievements. This chapter delves into potential directions for future studies, while also examining existing research gaps.

2. THEORETICAL FOUNDATIONS

2.1. Radar Concepts and Terahertz Radars

2.1.1. Introduction to radars

Radar stands for "Radio Detection and Ranging." It's a technology that uses radio waves to detect the presence, direction, distance, and speed of objects [18]. This technology was developed during World War II for military applications but has since found numerous civilian uses [19].

Radar systems typically consist of the transmitter and receiver system and corresponding antennas and display system components. Transmitter emits electromagnetic waves, usually in the form of radio waves or microwaves, into the surrounding space, directionally. Antenna sends out these waves in a particular direction and receives the echoes when they bounce off objects. Receiver processes the received signals, extracting information about the distance, speed, and direction of the objects that reflected the waves. And finally, the display system shows the information obtained from the received signals, often in the form of a visual representation, such as a screen showing the positions of detected objects.

Radar operates on the principle of sending out radio waves, which travel at the speed of light. When these waves encounter an object, some of the waves are reflected back towards the receiver of the radar system. By measuring the time, it takes for the waves to return and analyzing the frequency shift of the returning waves (Doppler effect), radar systems can determine the distance and speed of the object.

Statistical analysis techniques are fundamental in radar data analysis, helping extract valuable information from radar returns, mitigate noise, and make informed decisions.

CFAR dynamically adjusts detection thresholds based on local statistical properties, reducing false alarms caused by clutter [20]. Cell Averaging CFAR (CA-CFAR) and Ordered Statistics CFAR (OS-CFAR) are variants of CFAR techniques that compute thresholds using statistics of nearby cells or ordered data to improve clutter rejection. Probability Density Functions (PDFs) and Cumulative Distribution Functions (CDFs) are used to model and analyze radar returns, aiding in estimating signal and noise distributions. Maximum Likelihood Estimation (MLE) estimates parameters of probability distributions, applied in target detection and parameter estimation. Bayesian

Methods utilizes Bayesian inference to update probabilities based on prior knowledge and incoming data, often used for target tracking and parameter estimation.

Kalman Filters are widely used in radar systems for recursive statistical filters that predict and correct estimates of target state variables, crucial for tracking moving objects in cluttered environments [21]. Particle Filters (Sequential Monte Carlo Methods) estimate a target's state by maintaining a set of particles representing possible states, particularly useful for nonlinear and non-Gaussian systems.

Matched Filtering maximizes signal-to-noise ratio by convolving received radar signals with a filter matched to the expected signal waveform, aiding in target detection. Pulse Compression, on the other hand, uses statistical signal processing to compress radar pulses, enhancing range resolution and improving target detection in clutter.

Adaptive Beamforming adjusts radar beam patterns based on statistical environmental characteristics, improving target detection and clutter rejection [22]. And Adaptive Thresholding and Filtering dynamically adjusts thresholds or filter parameters based on changing environmental conditions to optimize radar performance.

2.1.2. Types of radar

Radar technology has diversified into various types to suit different applications and operational needs. There are various kinds of radar classified in different ways. They can be classified as primary and secondary radars based on its specific function. There are two types of primary radars as Continuous Wave (CW) or Pulse radar. CW radars can be modulated or unmodulated and Pulse radars can be Moving Target Indicator (MTI) and Doppler radars.

Also, radars can be classified with their applications such as search radars (surface, air), warning radars such as weather forecast radars, airborne radar, ground penetrating radar (GPR), synthetic aperture radar (SAR), over the horizon (OTH) radar, etc.

Primary radar also known as "pulse radar," emits short pulses of radio waves (in time domain) and detects the echoes (reflected signals) from objects. It determines the range and bearing of targets but typically doesn’t provide detailed information about the target's identity or composition (this is due to the nature of relatively long wavelength electromagnetic signals). Secondary surveillance radar (SSR) often used in conjunction with primary radar, SSR transmits an interrogation signal to an aircraft's transponder.

The transponder replies with encoded information such as the aircraft's identity and altitude. This radar is widely used in air traffic control. This is a negotiated application of radar systems, like wireless link between systems. CW radar emits a continuous wave of radio frequency. By measuring changes in the frequency of the reflected wave, CW radar can determine the speed of an object, but it doesn't provide distance information because it can't identify the corresponding transmitted signal of the received signal that is reflected from the object/objects. Doppler radar measures the velocity of targets by detecting changes in frequency (frequency shift) caused by the Doppler effect. It's often used in weather monitoring in aerial vehicles to track precipitation and wind speed. SAR utilizes microwave signals to create high-resolution images of the Earth's surface. SAR is used in remote sensing applications like mapping terrain, monitoring agriculture, observing changes in the environment [23]. Weather radar specialized radar used to track weather phenomena like precipitation, storms, and severe weather conditions [24]. It helps in forecasting and monitoring weather patterns. Phased array radar employs an array of antennas that can steer the radar beam electronically without physically moving the antenna. This allows for rapid scanning, flexible beam shaping, and tracking multiple targets simultaneously. It is also massively used in directional, high speed wireless communication links. Over-the-horizon radar (OTH) is designed to detect targets beyond the line of sight by bouncing signals off the ionosphere. OTH radar is used for long-range surveillance, especially in maritime and defense applications. Ground Penetrating Radar (GPR) emits short electromagnetic waves into the ground and measures the reflected signals to detect subsurface structures. GPR is used in archaeology, geology for object detection and classification [25]. Airborne radar is installed on aircraft for various purposes such as navigation, weather monitoring, terrain mapping, and military reconnaissance. It doesn't introduce any unique capabilities advantages than any radar technologies above, but it is just mobilized versions of aforementioned techniques. These radar types vary in their technology, operating principles, and applications, catering to specific needs across military, civilian, scientific, and commercial domains.

2.1.3. Advantages and challenges of radar technologies

There are many methods to detect, identify and track objects in various use cases like visual monitoring, acoustic tracking in fully autonomous, semi-autonomous or

manual manner. But radar systems have unique capabilities like versatility, long-range detection, all-weather operation, non-intrusive and remote sensing, speed and accuracy and continuous monitoring.

Radar systems can be tailored for a wide array of applications, including defense, navigation, weather monitoring, and remote sensing. Many radar systems can detect objects at significant distances, providing early warnings and surveillance capabilities. Radars are effective in various weather conditions, including fog, rain, and darkness, making them reliable for continuous monitoring. They allow for non-invasive detection and sensing, enabling surveillance and analysis without physical contact. Radar systems can measure the speed and position of objects with high precision and speed, crucial for air traffic control, military applications, and weather forecasting. Some radar systems enable continuous and real-time monitoring, providing ongoing surveillance capabilities and can do this fully autonomously.

2.1.4. Applications of radar

Radar technology has numerous applications, including weather forecasting, air traffic control, military defense, navigation for ships and aircraft, speed enforcement (such as police radar guns), and various scientific and civilian uses like mapping and remote sensing. Different types of radar systems have been developed to suit specific purposes, ranging from long-range surveillance to short-range imaging and sensing applications.

2.1.5. Range finding in radar

Radars do determine how far the target is by measuring the time it takes for a transmitted signal (electromagnetic waves) to travel to the target and return as an echo. The process involves the transmission, travel time, reception, time measurement and range calculation steps.

The radar system emits a short pulse of electromagnetic waves using its transmitter. These waves travel at the speed of light, and once they encounter an object, a portion of the waves reflects back toward the radar. The radar's receiver detects the reflected waves. By precisely measuring the time interval between the transmission of the signal and the reception of the echo, the radar system calculates the time taken for

the signal to make the round trip. Knowing that the signal traveled at the speed of light and having measured the time it took to return, the radar system uses the following formula:

$$Distance = Speed\ of\ Light \times \frac{Time\ Taken}{2} \tag{2.1}$$

This method of determining range is fundamental to most radar systems and enables them to detect and track objects at various distances accurately. Factors like the speed of light and precise timing measurements are crucial for the accuracy of range calculations in radar technology.

2.1.6. Doppler effect and velocity estimation in radar

The Doppler effect is a phenomenon observed in waves, including sound waves and electromagnetic waves like light or radio waves. It describes the apparent change in frequency or wavelength of a wave when there is relative motion between the source of the wave and the observer. In simpler terms, it's the change in the perceived frequency or pitch of a wave when the source of the wave or the observer is in motion relative to each other.

When a source of waves is moving towards an observer, the observer experiences a higher frequency or pitch than the actual emitted frequency. Conversely, if the source is moving away from the observer, the perceived frequency or pitch is lower than the actual emitted frequency. It can be formulated as:

$$f' = f \cdot \left(\frac{v + v_0}{v - v_s}\right) \tag{2.2}$$

Where f' is the observed frequency, f is the emitted frequency, v is the speed of the wave in the medium, v_0 is the speed of the observer relative to the medium and v_s is the speed of the source relative to the medium.

The radar receiver detects the reflected waves and measures the frequency shift caused by the target's motion. This shift is used to calculate the radial component of the target's velocity (velocity toward or away from the radar).

The Doppler frequency shift is directly related to the velocity of the target. The formula for Doppler frequency shift is:

$$f_D = \frac{2\,f\,v}{c}\cos\theta \tag{2.3}$$

Where f_D is the Doppler frequency shift, f is the frequency of the transmitted radar waves, v is the velocity of the target relative to the radar, c is the speed of light, and θ is the angle between the principal axis of the antenna and the velocity vector of the target object.

By measuring the Doppler frequency shift, radars can calculate the speed of the target along the radar line-of-sight. The sign of the frequency shift (positive or negative) indicates whether the target is approaching or moving away from the radar. This Doppler shift method allows radars to estimate the velocity of moving targets, which is crucial in various applications such as air traffic control, weather monitoring, speed enforcement, and military surveillance.

2.1.7. Introduction to terahertz technology

Terahertz (THz) radar works by emitting and detecting electromagnetic waves in the terahertz frequency range. General concepts still persist like any other radar systems, but systems are built and operated significantly differently.

The range is significantly lower than other relatively long wave radar systems, but it can detect small velocities, objects, rate of change of movement and change in angle. THz radar is a new concept, and it is adopting really quickly in various industries because of their advantages and its low cost, small sizes, and unharmful nature. Following figure shows the terahertz range in the electromagnetic spectrum.

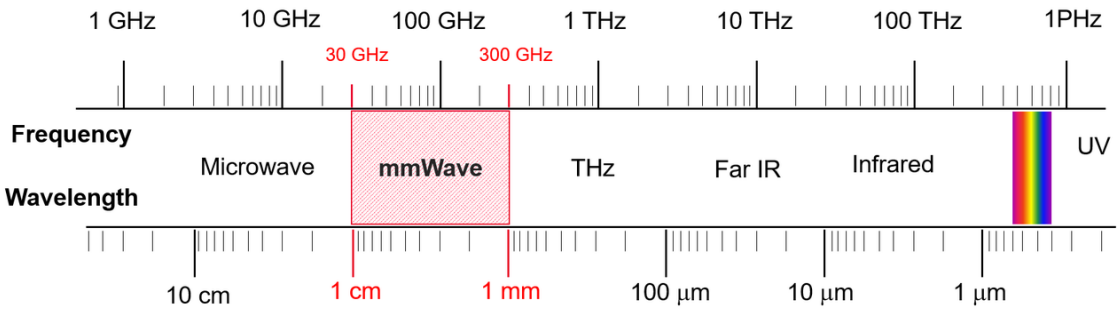

Figure 2.1. *Frequency and wavelength regions of the electromagnetic spectrum*

2.1.8. Applications of terahertz

As stated in the motivation section, THz radar technology operates within the electromagnetic spectrum between microwaves and infrared light, at frequencies ranging from 0.1 to 10 terahertz as shown in the Figure 2.1. It's gaining attention due to its unique capabilities in imaging, sensing, and communication. This technology offers advantages like high-resolution imaging without the harmful effects of ionizing radiation, making it useful in various fields such as security screening, medical imaging, material characterization, and even in cultural heritage preservation.

THz radar technology has a wide range of applications across various fields due to its unique properties. Some of the key applications include security screening, medical imaging, quality control in manufacturing, astronomy and remote sensing, spectroscopy and chemical analysis, non-destructive testing and wireless sensing and imaging.

THz radar can penetrate clothing and packaging, enabling the detection of concealed weapons, explosives, or illicit substances. It's used in airport security, border control, and high-security areas. THz waves can penetrate biological tissues without causing damage, allowing for non-invasive imaging. It holds promise for diagnosing skin diseases, detecting tumors, and imaging within teeth without the use of harmful ionizing radiation. THz radar helps assess material properties, detect defects in manufactured goods (such as ceramics, plastics, and composites), and monitor processes like pharmaceutical tablet coating. This radar type aids in studying celestial objects, atmospheric monitoring, and environmental sensing. It enables astronomers to study star formations and interstellar gasses. THz radar is used in spectroscopic techniques to identify and analyze specific molecules based on their unique spectral fingerprints. This is valuable in chemistry, biology, and environmental science. THz radars allow for non-

destructive evaluation of structures, like inspecting hidden layers in paintings, detecting structural defects in materials, and assessing the integrity of coatings and layers [26]. THz radar enables imaging through various materials, allowing for applications in robotics, automotive collision avoidance systems, and enhanced vision in poor visibility conditions.

As technology advances and researchers delve deeper into understanding and harnessing the capabilities of THz waves, the potential for new applications continues to expand. The technology is still evolving, facing challenges like limited range due to absorption by atmospheric moisture and the need for advancements in detector sensitivity and signal processing. Nonetheless, ongoing research and development continue to expand its applications and improve its capabilities.

2.1.9. Range measurements in terahertz radars

The main concept is still the same, transmit electromagnetic waves (signals), and receive the reflected ones. In FMCW terahertz radar, frequency of the transmitted signal is increased linearly by time. This is called a chirp signal. Time domain, FFT magnitude, and PSD representation of this signal is shown in the following figures.

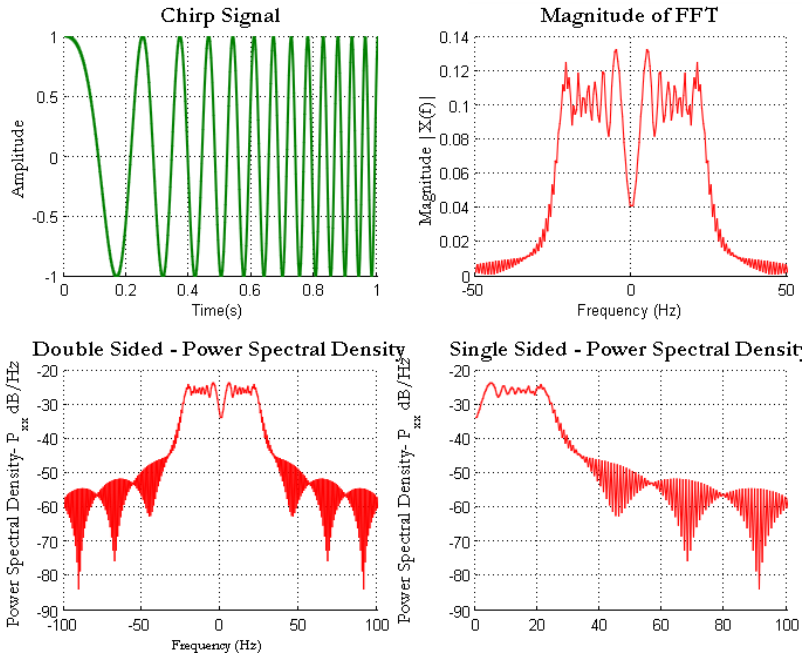

Figure 2.2. *Representation of the chirp signal in different domains*

These figures are just examples but on terahertz radar systems, this linear increment of the frequency in time domain happens in gigahertz per nanoseconds and repeats. An FMCW radar system transmits these signals and receives reflected systems from objects just like any other radar systems.

There are many reasons to choose chirp signals above others in terahertz radar systems. Chirp signals have a wide frequency range that allows for better range resolution in radar systems. By using a chirp signal, these systems can achieve finer distance measurements compared to simpler waveforms. Chirp signals provide better SNR and allow for more advanced signal processing techniques. They offer better discrimination against noise and interference, enabling clearer detection of targets.

Chirp signals enable pulse compression techniques. These signals can be compressed after transmission, which helps in maintaining a longer pulse duration for higher energy while still achieving fine range resolution. Chirp signals offer flexibility in bandwidth. By changing the chirp rate or duration, radar systems can adjust the effective bandwidth, allowing for versatility in different operational scenarios.

Overall, chirp signals enhance the capabilities of terahertz radar systems by providing better resolution, improved signal processing, and flexibility in bandwidth, which are crucial for various applications like imaging, sensing, and communication in the terahertz frequency range.

The Intermediate Frequency (IF) signal refers to an intermediate stage in the signal processing chain of various electronic systems, including radios, communication devices, and radar systems. In systems like radios and radar, the IF signal is an intermediate step between the received or transmitted signal and the final output or information extraction stage. In the context of a radar system, for example, the radar receives echoes or reflected signals from targets. The received signal undergoes initial amplification and filtering stages to extract relevant information and reduce noise. In many systems, the received signal is converted to a lower frequency, known as the intermediate frequency (IF). This conversion from the original received frequency to the IF frequency simplifies further signal processing stages. At the IF stage, the signal is easier to process, filter, and manipulate. This allows for better selectivity, easier amplification, and improved signal-to-noise ratio. After the IF stage, additional signal processing, demodulation, or further analysis is performed to extract specific information, such as target range, velocity, or other characteristics in radar systems.

The use of an IF signal offers several advantages, including easier amplification, simpler filtering, and improved signal processing capabilities. It allows for the separation of different stages of processing and enhances the overall performance of the system.

The generation of the IF signal is shown in the following figure. First, a synthesizer generates a chirp. Next, the chirp is transmitted by the antenna. Then, the reflection of the chirp by an object generates a reflected chirp captured by the receive antenna. At last, a mixer combines the RX and TX signals to produce an IF signal. The IF signal is created by mixing the carrier signal with a local oscillator signal in a process called heterodyning, resulting in a signal at the difference or beat frequency.

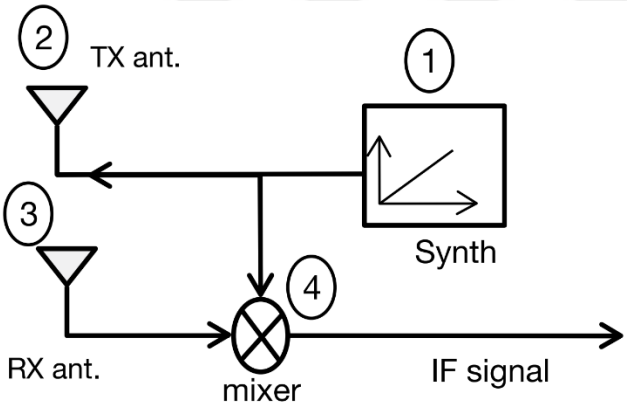

Figure 2.3. *FMCW radar block diagram*

The relationship between the distance(range) and IF signal can be expressed with the following formula:

$$Asin(2\pi f_0 t + \Phi_0) \tag{2.4}$$

where $f_0 = \frac{S2d}{c}$ and $\Phi_0 = \frac{4\pi d}{\lambda}$.

Range resolution is the ability to identify multiple objects in received reflected signal. If two or more objects move in the same direction (getting further or closer), radar systems mostly fail to separate them. In the defense industry, multiple military aerial vehicles fly in formation to fool the radar about the fleet and hide the objects from radar. Fourier theory implies that, if the length of the IF signal is increased, the resolution also increases. But the bandwidth of the IF signal must also be increased

proportionally. This results in two peaks in the IF signal spectrum. According to Fourier theory, again, an observation window can resolve this as long as it is less than the frequency difference between the two peaks of the IF signal. As a short-range resolution only depends on the bandwidth of the chirp signal. On a typical FMCW radar, chirp bandwidth is a couple of gigahertz and a 4GHz chirp bandwidth has a 3.75 range resolution.

The range resolution can be expressed with the following formula:

$$d_{res} = \frac{c}{2B} \tag{2.5}$$

where B is bandwidth of the transmitted pulse and c is the speed of light.

2.1.10. Velocity measurements in terahertz radars

To measure velocity, terahertz radars transmit two chirp signals that are separated with the observation interval. Radar system captures both reflected signal, takes FFT of them. Each chirp has chirps on the same location in the frequency domain with different phases. Velocity can be calculated using this phase difference as:

$$v = \frac{\lambda \Delta\phi}{4\pi T_c} \tag{2.6}$$

$$\text{where } \Delta\phi = \frac{4\pi T_c}{\lambda} \tag{2.7}$$

Since there is uncertainty measuring velocity based on phase difference, phase difference is less than pi. Since it mostly depends on observation interval, the calculated velocity is always less than wavelength over 4 times observation interval, stated as:

$$v < \frac{\lambda}{4T_c} \tag{2.8}$$

$$V_{max} = \frac{\lambda}{4T_c} \tag{2.9}$$

If multiple objects are moving with different velocities at the same distance, two chirps are not enough to estimate unique velocities. because they have reflected chirps

with equal IF frequencies. Because of that, phase difference can not be calculated correctly. To overcome this issue, the radar system must transmit more than two chirp signals, called chirp frames. The velocity resolution is inversely proportional to the frame period whose total period corresponds to N many observation intervals and it can be expressed as the following formula:

$$v_{res} = \frac{\lambda}{2T_f} \tag{2.10}$$

where T_f is the frame time.

2.1.11. Angle estimation in terahertz radars

A FMCW radar system is able to estimate angle of arrival (AoA) of the reflected signal if it has more than one receiving antenna. The distance difference between antennas and an object result in phase change in FFT peaks. With this phase difference, the angle of arrival can be estimated. This concept has shown in the following figure.

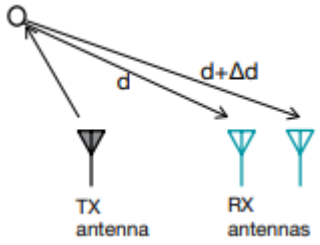

Figure 2.4. *Two antennas are required to estimate AoA*

$$\theta = \sin^{-1}\left(\frac{\lambda \Delta\phi}{2\pi l}\right) \tag{2.11}$$

$$\text{where } \Delta\phi = \frac{2\pi \Delta d}{\lambda} \tag{2.12}$$

The phase difference depends on sine of angle non-linearly. Sine of small values are equals to themselves, furthermore, the angle of arrival estimation accuracy increases significantly on smaller ones as shown in (2.11) and (2.12).

2.2. Unmanned Aerial Vehicle Detection

Detection of UAVs, commonly known as drones, is a crucial task in various fields due to their increasing use in both beneficial and potentially risky applications. Detecting UAVs involves identifying and tracking these flying objects to ensure safety, security, and regulatory compliance.

It is really important to detect, identify and track UAVs uncooperatively. Drones can pose security threats by carrying out unauthorized surveillance, smuggling contraband, or potentially causing harm in sensitive areas. To prevent collisions with manned aircraft, especially near airports or restricted airspace, monitoring UAVs is crucial. For privacy protection against unauthorized filming or intrusion of private spaces by drones, UAV detection using several techniques gains much attention in today's world.

There are many ways to detect, track, and identify unmanned aerial vehicles. Radar technology detects drones by analyzing their RCS and flight characteristics. This method provides long-range detection and works in various weather conditions. Detecting drones through the sound they produce, especially useful in urban environments where background noise does not exist, is also a common method. Monitoring radio frequencies used by drones for communication or control signals, detecting their presence or activity is another way of UAV detection in some cases. Using cameras or visual sensors to detect and track drones based on their visual signatures or characteristics is an example for optical UAV detection. It is also possible to detect the heat signature emitted by drones, especially useful for night-time detection or in environments with low visibility.

There are many challenges using above techniques to detect UAVs. Some drones are small, lightweight, and designed to have low radar visibility, making their detection challenging. Drones can use evasion techniques or adapt their flight patterns to avoid detection, requiring continuous advancement in detection technologies. High levels of background noise, buildings, and other obstructions can make detection more complex. Nevertheless, proposed method in this thesis is to bring new perspective and try to overcome these challenges regarding to solve this issue.

2.3. Machine Learning

Machine learning is a field of artificial intelligence (AI) that focuses on creating algorithms and systems capable of learning and making predictions or decisions without explicit programming. Essentially, it enables computers to learn and improve from experience automatically.

Machine learning algorithms use data to identify patterns, learn from examples, and make predictions or decisions without being explicitly programmed for each scenario. These algorithms learn from labeled or unlabeled data through a training process, generating models that can generalize and make predictions on new, unseen data. The system improves its performance iteratively by receiving feedback on its predictions and adjusting its model accordingly.

There are many machine learning methods [27]. Supervised learning algorithms learn from labeled data, mapping input to output. For instance, in image recognition, the algorithm learns to associate images of cats with the "cat" label. Unsupervised learning algorithms learn patterns and structures from unlabeled data, identifying inherent relationships or clusters within the data without specific guidance. Reinforcement learning agents learn by interacting with an environment, receiving rewards or penalties for actions, aiming to maximize cumulative reward over time.

Although there are many machine learning methods that exist, core concepts do not differ much. The first step is the data collection. Gather and prepare relevant data for training and testing. Next is feature extraction and selection. Identify meaningful features from the data to feed into the algorithm. Then, select the model choose an appropriate algorithm, train it on the data, and evaluate its performance.

Although machine learning algorithms are widely used in various industries, there are certain challenges and considerations. Machine learning heavily relies on data quality; inaccurate or biased data can lead to flawed models. Some complex models lack interpretability, making it challenging to understand how they make decisions. Bias in models, privacy issues, and potential societal impacts are important considerations in deploying machine learning systems.

There are many applications of neural networks in various areas. Recognizing patterns and features in images, audio, or video, processing and understanding human language, used in chat-bots, translation services, sentiment analysis, predicting user preferences in products or content based on past behavior and assisting in diagnosis,

predicting disease outcomes, and personalized medicine are some of the widely used machine learning applications.

2.3.1. Neural networks

Neural networks are fundamental component of artificial intelligence and machine learning, inspired by the structure and function of the human brain's neural networks. They consist of interconnected nodes, or artificial neurons, organized in layers that process information, allowing machines to learn patterns and make decisions.

The key components are neurons, layers and connections. Neurons in neural networks mimic biological neurons and perform computations. Each neuron receives inputs, processes them using activation functions, and produces an output. Layers typically composed of an input layer, one or more hidden layers, and an output layer. The input layer receives data, the hidden layers process it, and the output layer generates the final result. Connections (weights and biases) are the connections between neurons that have a weight that determines the strength of the connection, influencing the information flow. Biases are added to neuron inputs, affecting their activation thresholds.

There are many types of neural networks. In Feed-forward Neural Networks (FNN) information moves in one direction, from input to output, without loops or cycles. These are basic networks used in tasks like classification and regression. Recurrent Neural Networks (RNN) allow connections to form cycles, enabling them to process sequential data by retaining memory of past inputs. Widely used in tasks like language modeling, speech recognition, and time series analysis. CNNs are primarily for image recognition and processing. They use specialized layers called convolutional layers that detect patterns in spatial data. Generative Adversarial Networks (GAN) are composed of two neural networks, a generator, and a discriminator, competing against each other to generate and distinguish synthetic data from real data.

Although the types of neural networks differ, the core concepts are still valid. These concepts are called forward propagation and back propagation. In forward propagation the data is passed through the network layer by layer, computing weighted sums and applying activation functions to produce an output.

Neural networks are capable of learning complex patterns, adaptability to various data types, and their ability to improve with more data. But prone to over-fitting (fitting

too closely to training data), require large amounts of data for training, and can be computationally intensive. Nevertheless, neural networks are a powerful tool in machine learning, driving significant advancements in various fields by enabling machines to learn from data, recognize patterns, and make intelligent decisions. Application areas are the same since it is still a machine learning type.

Neural networks can be highly effective for radar data analysis, providing valuable insights, improving target detection, and aiding in various radar-related tasks [28]. Neural networks can automatically learn relevant features from radar returns, identifying patterns in complex radar data that might be challenging to specify manually. They can classify radar returns into different categories, such as distinguishing between various types of targets (e.g., aircraft, birds, clutter) or identifying specific weather patterns in weather radar data. Neural networks can learn to filter out noise and clutter from radar returns, enhancing the signal-to-noise ratio and improving the accuracy of target detection. Combining neural networks with CFAR techniques can optimize clutter rejection by dynamically adjusting thresholds based on learned patterns.

In SAR, neural networks can reconstruct high-resolution images from raw radar data, improving image quality and detail. In Inverse Synthetic Aperture Radar (ISAR), neural networks assist in processing and analyzing radar returns to generate detailed images of moving targets like ships or aircraft.

Neural networks integrated with Kalman filters or other tracking algorithms can improve target tracking accuracy by predicting target trajectories based on historical radar data. They aid in tracking multiple targets simultaneously, predicting movements, and maintaining a clear picture of the monitored area. They can assist in optimizing pulse compression techniques, improving range resolution and target discrimination in radar systems. Employing neural networks for adaptive beam-forming in phased array radars, optimizing radar beam patterns based on learned environmental characteristics.

Neural networks can detect abnormal patterns or unexpected events in radar data, aiding in identifying potential threats or irregularities. Neural networks offer powerful capabilities in learning complex patterns and relationships from radar data, enhancing the accuracy, efficiency, and capabilities of radar systems across multiple applications and domains.

2.3.2. Classification methods

Data classification techniques are methods used to categorize or assign labels to data based on their attributes or features. These techniques are fundamental in machine learning and are applied to various types of data for tasks such as predicting outcomes, grouping similar items, or identifying patterns.

Decision Trees are hierarchical structures that make decisions based on features, dividing data into classes with simple if-else rules. Support Vector Machines (SVM) separates data by finding the optimal hyperplane that maximizes the margin between different classes. K-Nearest Neighbors (KNN) classifies data points based on the majority class of their k-nearest neighbors in feature space. Naive Bayes Classifier utilizes Bayes' theorem to predict the probability of a class based on the features' conditional probabilities.

MLP classification, employing a multi-layered architecture with interconnected nodes and activation functions, tackles classification problems by learning complex patterns in data. Comprising input, hidden, and output layers, it utilizes back-propagation for iterative weight adjustments, aiming to minimize prediction errors. With its ability to handle non-linear relationships and approximate various functions, MLPs serve as powerful tools across domains like image recognition, natural language processing, and diverse predictive analytics applications. Tuning parameters such as layer structure, activation functions, and regularization techniques impacts their performance in achieving accurate classification results.

Principal Component Analysis (PCA) and classification are interconnected in machine learning, particularly in preprocessing and feature engineering for classification tasks. PCA is a dimensionality reduction technique that identifies the most informative features while reducing the dimensionality of the dataset by transforming correlated variables into a set of linearly uncorrelated ones called principal components. In classification, PCA serves to preprocess and enhance the data by reducing noise and redundancy, aiding in feature selection and visualization, and speeding up training. By condensing the data into fewer dimensions, PCA can help classifiers focus on the most relevant information, potentially improving their performance, especially when dealing with high-dimensional datasets. Moreover, PCA can be used in conjunction with classifiers to enhance their efficiency, especially when the original feature space is large

and contains redundant or irrelevant information, thereby contributing to more accurate and efficient classification models.

2.3.3. Regression methods

Regression is a statistical method used in machine learning to model and analyze the relationships between a dependent (target) variable and one or more independent (predictor) variables. Its primary goal is to understand how the value of the dependent variable changes concerning changes in the independent variable(s). Regression models seek to find the best-fit line, curve, or surface that represents these relationships, allowing for prediction or estimation of the dependent variable based on the independent variables.

KNN regression is a machine learning algorithm used for regression tasks, wherein it predicts the continuous output variable for a new data point based on the average or weighted average of the output values of its k-nearest neighbors in the feature space. Unlike KNN classification, which determines the class based on the majority vote of neighbors, KNN regression computes the output by averaging the target values of the neighboring data points. The prediction for a new data point is calculated as the mean or weighted mean of the target values of its k nearest neighbors, often assigning more influence to closer neighbors through weighted averaging schemes. KNN regression doesn't assume any underlying functional form between the input and output variables but relies on the proximity of data points in the feature space, making it particularly useful for nonlinear relationships and localized patterns within the data. Adjusting the value of k impacts the smoothness and flexibility of the regression curve, with smaller k values leading to more complex and potentially over-fitted models, while larger k values result in smoother, simpler models.

Linear regression is a statistical method used to model the relationship between a dependent variable and one or more independent variables by fitting a linear equation to the observed data. The model assumes a linear relationship, where the dependent variable is a linear combination of the independent variables, with coefficients that define the slope and intercept of the line. The goal of linear regression is to find the best-fit line that minimizes the difference between the predicted and actual values of the dependent variable. This technique serves to understand the direction and strength of the relationship between variables, make predictions, and estimate the impact of changes in

independent variables on the dependent variable, providing a straightforward and interpretable approach to modeling continuous relationships in data.

Multilayer Perceptron (MLP) regression is a form of artificial neural network used for regression tasks, employing multiple layers of interconnected nodes to predict continuous output values based on input features. Unlike traditional linear regression, MLP regression can capture complex nonlinear relationships between input and output variables. It comprises an input layer, hidden layers with numerous neurons, and an output layer, where each neuron utilizes activation functions to introduce non-linearities and transform input data through weighted connections. Through an iterative process involving forward propagation and back-propagation, the network adjusts its weights to minimize the difference between predicted and actual output values, aiming to optimize its predictive accuracy. MLP regression, with its ability to model intricate patterns and relationships in data, is utilized across various domains for predicting continuous outcomes, such as in finance for stock price forecasting or in engineering for predicting values of continuous variables based on complex inputs.

2.3.4. Ensemble neural networks

An ensemble neural network model combines multiple individual neural networks to improve predictive performance and generalization. Instead of relying on a single neural network's predictions, ensembles leverage the collective knowledge of multiple models to make more accurate predictions.

There are various ways to create ensemble neural network models. Bagging (Bootstrap Aggregating) involves training multiple neural networks independently on different subsets of the training data. Each network has equal weight in the final prediction. Boosting method trains multiple neural networks sequentially, where each subsequent network pays more attention to the examples that the previous ones misclassified. Gradient Boosting Machines (GBM) and AdaBoost are popular techniques in this category. In Stacking Model, the predictions from multiple individual networks become inputs for a meta-learner (another neural network or a different model) that combines these predictions to make the final prediction.

Ensemble methods often lead to better performance than individual models [29] because they can compensate for biases and errors in each individual model.

3. PROPOSED METHOD

3.1. General Structure of The System

The proposed method is to detect UAVs and extracting specific propeller-related features (presence, speed, diameter, distance, thrust level) from mmWave Doppler radar data involves a multi-step process combining radar signal analysis, feature extraction, and machine learning techniques.

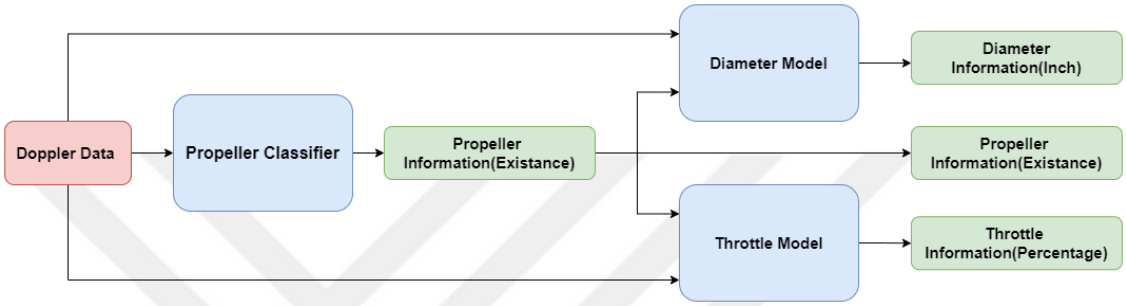

Figure 3.1. *System overview of the proposed method for propeller recognition*

The IWR1843 FMCW Doppler radar module supplies pre-processed 2-Dimensional (2D) FFT Doppler data (as range bins and FFT Doppler bins of 4096×4 bytes). The system gets this data and feeds into the propeller classifier. This neural network is one of many in the system that can be called an “Ensemble Model” and feeds other neural networks whether a propeller exists or not in this Doppler data recording as shown in the Figure 3.1.

At the next step, there are two more neural networks to predict diameter and throttle level of this propeller also shown in the Figure 3.1. The overall score is enhanced by the propeller classifier output since if there are no propeller in the data, the diameter and throttle should be 0.

3.2. Data Acquisition and Preprocessing

Employed a IWR1843 FMCW Doppler radar module operating in the millimeter-wave frequency band capable of capturing Doppler shifts caused by moving objects.

At each test setup, captured radar returns from the environment where UAVs are expected to operate, recording a variety of UAV flight scenarios and distances. This is simplified with scenarios with only one propulsion system (single motor, electronic

speed controller, propeller, battery and RC receiver) with different propeller sizes (to estimate the propeller size to identify which commercial drone it might be).

Preprocessing is mostly done in the module itself. This step is about cleaning the radar data, removing noise, clutter, and artifacts. Apply filtering techniques to isolate moving objects and extract relevant signals associated with propeller movements.

3.3. Feature Extraction

Propeller related features must be extracted to predict propeller existence, diameter, distance and thrust level. Analyzes must be done for the presence of Doppler shifts in radar returns indicating moving objects, identifying potential propellers in the radar data using neural networks. This neural network must process the frequency shift in the radar returns to estimate the speed of the detected propellers.

For diameter estimation, periodic frequency components corresponding to propeller rotations must be extracted. To do that, signal processing methods (e.g. Fourier analysis) should be utilized to estimate the diameter based on these periodic components. Thrust level estimation is about correlating Doppler shifts and changes in velocity over time to estimate the thrust level of the UAV's propellers.

3.4. Performance and Quality Analysis

There are well-defined methods for performance and quality analysis methods for neural networks called validation and testing. Cross validation evaluates the trained classifier's performance using cross-validation techniques on labeled radar data to assess accuracy, precision, and recall for each extracted feature. Real-time testing is about implementing the trained classifier in a real-time system to process live radar data, continuously detecting UAVs and extracting their propeller-related features. During the implementation of the proposed method, many neural network types and their corresponding parameters were investigated and fine-tuned to increase the overall result.

Model optimization was done to fine-tune the classification and regression models and feature extraction techniques to improve accuracy, reduce false positives, and enhance feature estimation precision. Parameters of the models were tuned to optimize the model's hyper-parameters and feature extraction parameters for better performance.

4. IMPLEMENTATION

Many Doppler radar data were recorded in a controlled environment. Each recording session, only one of the features of the UAV propeller such as diameter, thrust level, distance, angle, presence was changed.

4.1. Data Gathering

Developed a mobilized data gathering setup to collect Doppler data in various conditions both indoor and outdoor. It consists of a radar module (IWR1843) with a laptop to collect the data and a mobilized thrust test stand consisting of a motor, electronic speed controller (ESC), propeller, battery, RC transmitter and receiver as shown in the Image 4.1.

The recordings were done in a controlled environment, each recording session, only controlled condition was changed like location (indoor, outdoor), distance between thrust stand and radar module, rotational speed of the propeller and angle of the radar module with respect to rotation plane of the propeller.

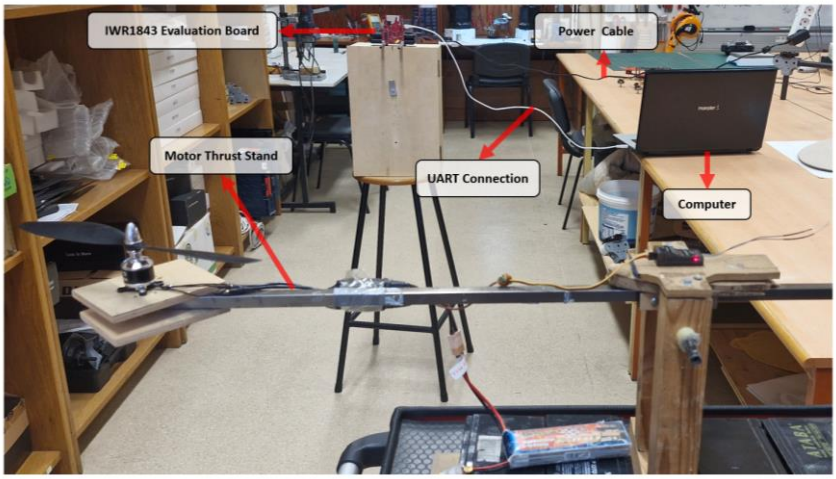

Image 4.1. *Data gathering setup*

4.1.1. Tools

This section provides more information about hardware and software tools included in the system to implement the proposed method.

4.1.1.1. IWR1843

The IWR1843 is a specific type of radar sensor manufactured by Texas Instruments (TI). It's a part of TI's mmWave sensor portfolio, specifically designed for industrial applications, robotics, automotive, and other areas where high-performance radar sensing is required. The IWR1843 is based on mmWave radar technology, operating at frequencies above 60 GHz. These frequencies enable higher resolution and accuracy in object detection, tracking, and classification compared to lower frequency radar systems. The front and back of the module has shown in the Image 4.2.

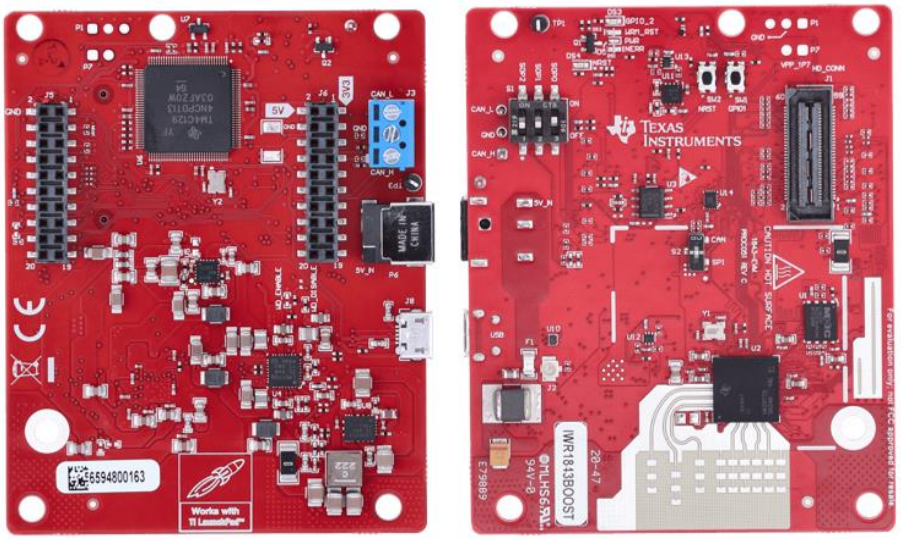

Image 4.2. *IWR1843 module*

Key features of the IWR1843 include high-resolution sensing, multiple antenna configuration, digital signal processing, integrating processing and interface and adaptive parameter configuration.

It provides high-resolution sensing capabilities, allowing for accurate detection and tracking of objects in its field of view. The sensor integrates multiple transmitters and receivers (antennas), which enable it to perform advanced radar signal processing techniques like beam-forming and MIMO (Multiple Input, Multiple Output). The IWR1843 incorporates a powerful digital signal processor, which processes the received radar signals to extract meaningful information about the surrounding objects. It features integrated processing capabilities, reducing the need for external processing units. It also offers various interfaces for easy integration into different systems. The

sensor is capable of adapting its parameters dynamically to optimize performance based on the environment and application requirements.

Applications of the IWR1843 include industrial automation, robotics, advanced driver-assistance systems (ADAS) in automotive, traffic monitoring, and more. Its capabilities in detecting and tracking objects, even in challenging environmental conditions, make it suitable for various real-time sensing applications.

FMCW sensors are able to determine velocity, range and angle of multiple objects by using their RF, analog and digital electronic components. This module has integrated digital signal processing, micro controller unit, transmitting and receiving antennas, analog and digital components on a single chip. It is able to store 512 chirps. This significantly increases the effective range and resolution of range, velocity, and angle.

Below is the antenna front-end of the IWR1843 FMCW radar module.

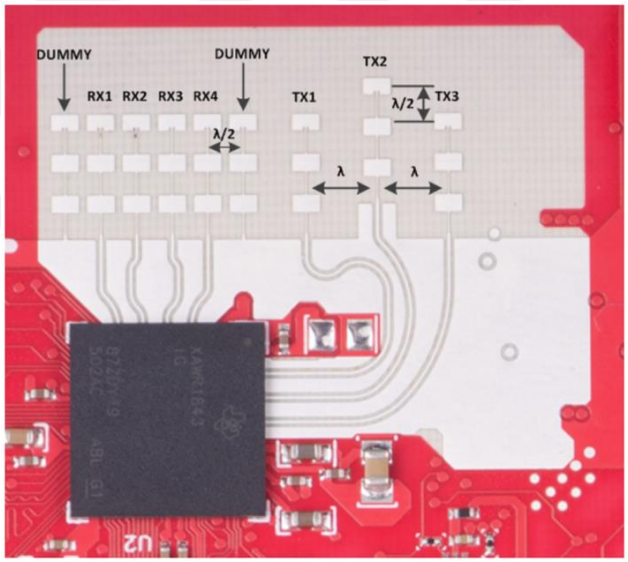

Image 4.3. *IWR1843 chip and PCB antenna*

FMCW transceiver of the IWR1843 has features of integrated Phase Lock Loop (PLL), transmitter, receiver, base-band, and Analog-to-Digital Converters (ADC), 76-81GHz coverage with 4 GHz available bandwidth, 4 receive channels, 3 transmit channels, ultra-accurate chirp engine based on fractional-N PLL, 12 dBm TX power and 25000 ksps ADC sampling rate.

The configurations of this module during this thesis supports velocities up to 5 m/s and range up to 11 meters. These spans depend on the configurations, maximum

detecting velocities can be increased but this decreases the detection range, and vice-versa. It can be calculated using the following formula:

$$d_{max} = \frac{300 \times fs}{2S \times 10^3} \tag{4.1}$$

where fs is sample rate expressed as ksps and S is the frequency slope constant expressed as $MHz / \mu s$.

$$v_{max} = \frac{3 \times 10^8}{4 \times f_{start} \times 10^9 \times (t_{idle} + t_{end}) \times 10^{-6} \times n_{tx}} \tag{4.2}$$

where f_{start} is start frequency in GHz, t_{idle} is idle time in μs, t_{end} is ramp end time in μs and n_{tx} is the number of transmitting antennas.

The configurations of this module during this thesis were given in the appendix.

4.1.1.2. Propellers

Since data that will be investigated mostly depends on the motion of the propeller, several propeller types were used for data gathering. These includes 9, 10, 11 and 12 inches of propellers. These propellers were chosen because they are widely used in the commercial drones (most of the DJI brand drones) that are big enough to have potential danger. In popular Mini UAVs on the market, generally propeller sizes of 9-12 inches are used. That's why these propeller sizes were prioritized during data recording sessions. Some popular drone models and the propeller sizes used are shown Table 4.1.

Table 4.1. *Various propeller sizes of drones*

Drone	Propeller Diameter(inch)
3DR Solo-Smart Drone	10
Skydio 2	13
DJI Phantom 3	9
Helimax Form500	12

In the following image, propellers used during the data gathering step were shown.

Image 4.4. *Various propeller types used for propeller size identification*

4.1.1.3. Motor thrust stand

During the data gathering step, it is crucial to safely operate the drone propeller attached to motor. To do that, a motor thrust stand was developed during the implementation. It is highly portable to collect data in various environmental conditions. It has shown in the Image 4.5.

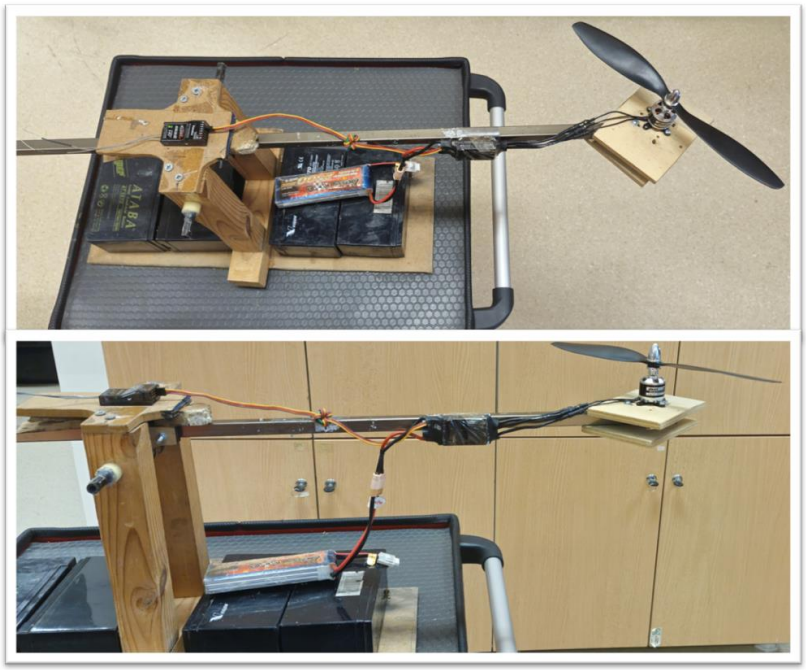

Image 4.5. *Motor thrust stand*

4.1.1.4. Motor, esc, battery & rc system

Each propeller and electric powered UAVs have motors, propeller, ESCs and batteries. The motor, ESC and propeller part called “Propulsion System” of the UAV. To simply implementing the proposed method, Doppler data of the single propeller were investigated.

Investigating the affects of the rotational speed of the propeller to the Doppler data was also a part of this implementation since during a flight, there are infinitely many maneuvers for a UAV based on the rotational velocity difference of the propellers. To mimic this, the rotational speed of each propeller must be changed during time. For this, an RC link was established between RC transmitter and receiver. These are shown in the Images 4.6-4.8.

Image 4.6. *RC transmitter*

Image 4.7. *Li-Po battery*

Image 4.8. *RC receiver – motor – esc*

4.1.2. Environments

Two environments, one indoor and one outdoor were chosen for recording sessions. Thanks to the mobilized motor thrust stand, data was collected in various environments, angle of arrivals, distance and thrust levels. During some of the recording sessions, other moving objects were in the environments, too, like walking people, moving cars, standing objects to introduce undesired noise and non-relational data to the recordings. Recording environments can be seen in the Image 4.9.

Image 4.9. *Different environments for data collecting.*

4.2. Data Processing

4.2.1. Environment

The provided SDK (Software Development Kit) with the module only works in Windows, which is not the best fit for embedded development and machine learning tasks. Several software components were implemented for data gathering, pre-processing, visualizing, neural network model training, evaluating in Linux environment. Software components were implemented with portability, being embedded

and ease of use in mind. Libraries were used in the scripts are easy to find and version controlled, so they are compatible with each other and highly portable.

All the scripts are written in Python programming language. Some example source codes and libraries (with corresponding versions) were given in the appendix.

4.2.2. Tools

4.2.2.1. Data gathering

Significant efforts were made to decode data communication protocol of this module. There were an SDK, written in C++ as a proprietary software, but in order to put together machine learning libraries in Python programming language and supplied data from this module, the Doppler data recordings must be decoding.

Below schema is how to decode live serial data from IWR1843 FMCW radar module.

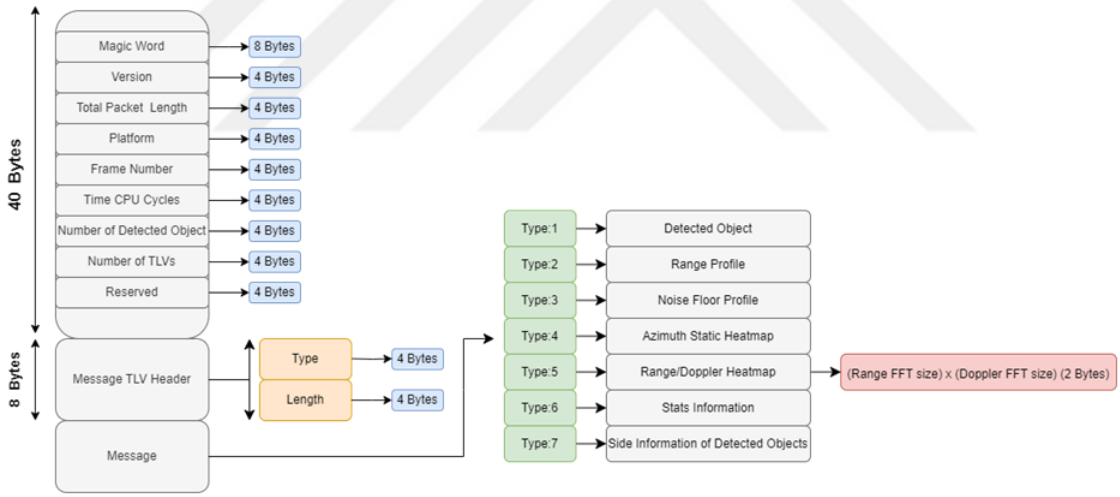

Figure 4.1. *Output packet structure of IWR1843*

The IWR1843 module can be connected to a computer using the supplied USB cable. When it is connected, the Operating System (OS) sees two ports related to this module. One port is for configuring the module (at 115200 baud) and another one is for real time data stream (at 921600). Default configurations used during Doppler data recordings can be found in the appendix. There is also a sample Python code about how to send these configurations to the radar module.

At the data port, Doppler data is supplied as a serial stream with a predefined packet format. Each packet starts with 40 bytes of header that holds information (type, length, etc.) about the incoming data in this packet. In the appendix, example codes can be found about how to decode these packets.

For future use in other codes, the data processing code exports the Doppler data to Comma Separated Values (CSV) files. These files can be processed with any programming language.

4.2.2.2. Data plotting

It is also important to be able to visualize the captured data for randomly selecting and manually (visually) processing whether everything is fine regarding data capturing and processing. Because, if the input data is not good enough, machine learning models will perform poorly. To overcome these issues, data plotting Python codes were written. Below is some outputs of data plotting tools.

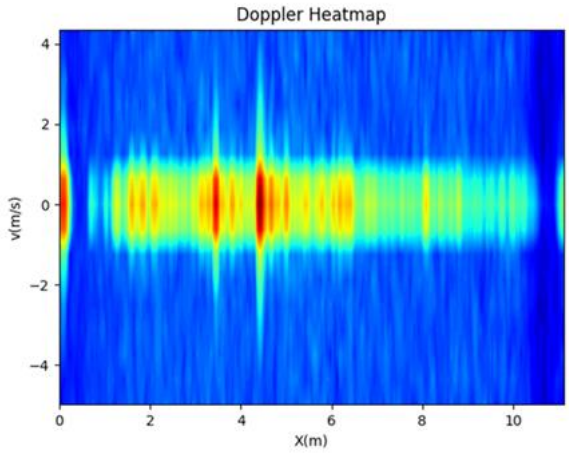

Figure 4.2. *Doppler heatmap without propeller movement*

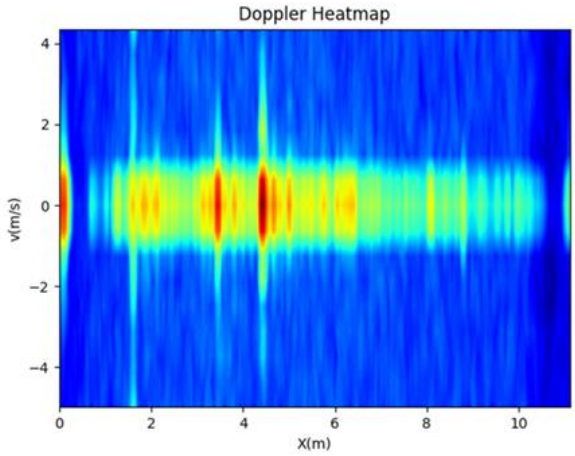

Figure 4.3. *Doppler heatmap with propeller movement*

Figure 4.2 is an empty test environment, and the Figure 4.3 contains a 9-inch propeller rotated 1.5m far away from the radar module, rotating at 50% throttle. It is clearly seen that the Doppler heatmap contains a velocity source around 1.5 meters. Other sources seen in the figure that have velocities close to 0 are the stationary objects located at the test environment.

4.2.2.3. Neural network training and evaluating

Data training models were implemented in Python programming language using Scikit-learn, Keras and TensorFlow. Data manipulations were done using NumPy library and for manual evaluation and plotting, matplotlib library was used.

To configure the labeled input data for later use in the data training models, first, converted from binary data to the CSV data. Then, in scikit-learn, the “StandardScaler” operation was used. It is a preprocessing technique used to standardize the features of a dataset. Standardization is the process of re-scaling the features so that they have the properties of a standard normal distribution with a mean of 0 and a standard deviation of 1. This is often important in machine learning algorithms that are sensitive to the scale of the input features.

The “StandardScaler” class in scikit-learn provides a convenient way to perform standardization. The fit method computes the mean and standard deviation of each feature from the training data, and the transform method then applies the standardization to any set of data using the computed mean and standard deviation.

After standardization, each feature will have a mean of 0 and a standard deviation of 1. This is especially important when using algorithms like support vector machines, k-nearest neighbors, or principal component analysis, where the scale of the features can significantly impact the performance of the model.

The data at each training session was splitted with 80% training data and 20% test data. The validation data was taken from the 20% of the training data. As a result, there are 26566 training data, 5313 validation data and 6642 testing data. At each session, before splitting, the data was shuffled in order to eliminate the bias.

There are several data training models were trained using aforementioned libraries. For classification models, MLP Classifier, KNN Classifier, SVM SVC, SVM Linear SVC, SVM Nu SVC.

In scikit-learn, the “MLPClassifier” stands for Multi-Layer Perceptron Classifier, which is a type of artificial neural network used for classification tasks. An MLP is a type of feed-forward neural network with multiple layers, including an input layer, one or more hidden layers, and an output layer. Each layer consists of nodes (neurons), and there are connections (weights) between these nodes. The “MLPClassifier” in scikit-learn is an implementation of a supervised neural network classifier, and it uses the back-propagation algorithm for training. The back-propagation algorithm minimizes the error by adjusting the weights in the network based on the gradient of the error with respect to the weights. During the experiments, activation function, solver type and hidden layer size parameters were the controlled variables to fit the model to the data. The generated models were evaluated with training loss and validation score results.

KNN Classifier, K-Nearest Neighbors, is a simple algorithm that stores all available cases and classifies new cases based on a similarity measure (e.g., distance functions) to the nearest neighbors in the training set. In scikit-learn, the “KNeighborsClassifier” class provides an implementation of the KNN algorithm for classification. During the experiments, number of neighbors parameters is the variable to observe which model suits best to the Doppler data. The generated models were evaluated with training accuracy, test loss and test accuracy.

SVM SVC, Support Vector Machine with the C-Support Vector Classification (SVC), is a supervised learning model used for classification tasks. It tries to find the best decision boundary (hyperplane) that separates different classes by maximizing the margin between them. In scikit-learn, this model is provided as the SVC class. SVM

Linear SVC is another form of SVM, specifically uses a linear kernel for classification tasks. It works well when the data is linearly separable. SVM Nu SVC is similar to SVC but with a slightly different formulation. In scikit-learn, this model is provided as the “LinearSVC” class. Nu SVC allows setting a parameter "nu" to control the number of support vectors and errors in the training set. In scikit-learn, this model is provided as the “NuSVC” class.

During the support vector machine model training sessions, no parameter was changed except the training data size. The epoch size was 1000 samples and trainings were done up to 25000 data, step by step. The generated SVC models were evaluated with F1 score, accuracy, precision, and recall that commonly used metrics to evaluate the performance of classification algorithms. Each metric provides a different perspective on the model's performance based on different aspects of the confusion matrix, which summarizes the classification results.

Accuracy represents the overall correctness of the model and is calculated as the ratio of correctly predicted instances to the total number of instances. While accuracy is a commonly used metric, it may not be suitable for imbalanced datasets. Precision, on the other hand, focuses on the number of positive predictions. It is the ratio of correctly predicted positive observations to the total predicted positives. Recall measures the ability of the model to capture all the positive instances. It is the ratio of correctly predicted positive observations to the actual positives. F1 Score is the harmonic mean of precision and recall. It provides a balance between precision and recall, making it useful when there is an uneven class distribution. Formulation of accuracy, precision, recall and F1 score are shown in the equations 4.3 – 4.6.

$$Accuracy = \frac{Number\ of\ Correct\ Predictions}{Total\ Number\ of\ Predictions} \tag{4.3}$$

$$Precision = \frac{True\ Positives}{True\ Positives + False\ Positives} \tag{4.4}$$

$$Recall = \frac{True\ Positives}{True\ Positives + False\ Negatives} \tag{4.5}$$

$$F1\ Score = \frac{2\ x\ Precision\ x\ Recall}{Precision + Recall} \tag{4.6}$$

In summary, accuracy measures overall correctness, precision focuses on the accuracy of positive predictions, recall measures the ability to capture all positive instances and F1 score provides a balance between precision and recall.

For classifier methods, additional to the above methods, confusion matrix was also used to evaluate the overall performance of the model instances. Confusion matrix is a table used to evaluate the performance of a classification algorithm. It provides a detailed breakdown of the model's predictions, showing the number of instances that were correctly or incorrectly classified into different classes. The confusion matrix is a useful tool to assess the performance of a model, especially in binary classification problems.

The members of confusion matrix are true positive, true negative, false positive and false negative. True Positive (TP) is the number of instances that were correctly predicted as positive. True Negative (TN) is the number of instances that were correctly predicted as negative. False Positive (FP) is the number of instances that were incorrectly predicted as positive (Type I error). And False Negative (FN) is the number of instances that were incorrectly predicted as negative (Type II error). As mentioned above, accuracy, precision, F1 score, and recall were derived in terms of these elements.

There are several regression models were trained using aforementioned libraries. For regression tasks, MLP Regressor, KNN Regressor, Linear Regressor models were used.

In scikit-learn, the “MLPRegressor” class stands for Multi-layer Perceptron Regressor, which is a type of artificial neural network used for regression tasks. Input, output and hidden layers, activation function, solver type, learning rate and other parameters can be changed to fine tune to the input data.

The “KNeighborsRegressor” (K-Nearest Neighbors) model in scikit-learn is a simple yet effective algorithm used for regression tasks. Instead of learning a function from the training data, it predicts the output of a new instance by averaging the target values of the k closest instances in the training set. Only the neighbor count was changed during the learning step.

The “LinearRegression” class in scikit-learn is a fundamental linear model used for regression tasks. It models the relationship between the dependent variable and one or more independent variables by fitting a linear equation to the observed data. During

the training, number of samples were the controlled variable to create a regression instance that fits best to the training data.

It is crucial to evaluate the trained models using several methods. To evaluate the regression models, outputs of the R2 score, Mean Squared Error (MSE), loss curve, validation score, test score, train score and train loss were used.

The R-squared (R^2) score, also known as the coefficient of determination, is a metric used to evaluate the performance of a regressor model. It assesses how well the independent variables (features) explain the variance in the dependent variable (target) within a regression model. R-squared measures the proportion of variance in the target variable that is predictable from the independent variables. It's a value between 0 and 1. A higher R-squared value indicates that a larger proportion of variance is captured by the model. An R-squared value of 0 indicates that the model fails to explain any variance in the target variable. An R-squared value of 1 indicates that the model perfectly explains the variance in the target variable. And higher R-squared values (closer to 1) suggest that the model better fits the data. R-squared doesn't indicate whether the model predictions are biased or accurate. A high R-squared does not necessarily mean the model's predictions are always accurate. It can be artificially inflated by adding more variables, even if they're not truly related to the target variable. The R-squared score provides a convenient way to quantify how well a regression model fits the data, but it's crucial to use it alongside other evaluation metrics and consider the context of the problem to draw accurate conclusions about the model's performance. The calculation of R-squared value can be expressed as:

$$R^2 = 1 - \frac{\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{N}(y_i - \bar{y})^2} \tag{4.7}$$

MSE calculates the average of the squared differences between predicted and actual values. MSE is always a non-negative value, with lower values indicating better model performance. Smaller MSE implies that the model's predictions are closer to the actual values on average. MSE penalizes larger errors (due to squaring) more than smaller errors, making it sensitive to outliers. When evaluating a regressor model, a lower MSE signifies better predictive performance. MSE can be be formulated as:

$$MSE = \frac{1}{n}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2 \tag{4.8}$$

The loss curve, also known as the training curve or learning curve, is a visual representation of how the loss function changes over epochs or iterations during the training process of a regressor model. It helps in evaluating the performance and behavior of the model during training. In the context of training a regressor model, the loss function (or cost function) measures the discrepancy between the predicted values and the actual target values. The loss function used during training can be MSE, Mean Absolute Error (MAE), or other regression-specific loss functions. During the training of a regressor model, the model iteratively adjusts its parameters (weights and biases) to minimize the loss function. At each iteration (or epoch), the model makes predictions on the training data, calculates the loss using the chosen loss function, and updates the parameters using optimization algorithms like Gradient Descent. A steep decline in the loss curve initially indicates rapid learning or improvement in the model's performance. A plateau or slow decrease in the loss curve might suggest that the model is converging or that further training might not significantly improve performance. Abrupt spikes or irregularities in the loss curve might indicate issues like over-fitting (if observed in validation loss) or learning instability.

Validation score is a crucial part of evaluating the model's generalization ability and is used to avoid over-fitting. During the training of a regressor model, a portion of the data (not used for training) is set aside as a validation dataset. The validation dataset acts as an independent set of data to assess how well the model generalizes to unseen examples. A higher validation score (e.g., higher R-squared or lower MSE/MAE) indicates better performance of the model on unseen data. Consistently high performance on both the training and validation datasets suggests the model has learned to generalize well. Discrepancies between training and validation scores can indicate over-fitting. If the model performs significantly better on the training set compared to the validation set, it might imply over-fitting.

The test score indicates how well the model generalizes to new, previously unseen data. A higher test score (e.g., higher R-squared or lower MSE/MAE) signifies better performance of the model on real-world data. The test score provides a final assessment of the model's performance before deploying it in a real-world scenario. It serves as an

unbiased estimate of the model's performance, giving insights into its effectiveness in making predictions on new, future observations.

The training score is calculated as after training the regressor model on the training dataset, the same dataset is used to evaluate the model's performance. The model makes predictions on the training dataset, and the chosen evaluation metric (such as R-squared, MSE, MAE) is computed using these predictions and the actual target values from the training dataset. The train score reflects how well the model fits the training data it has seen during the learning process. A high train score (e.g., high R-squared or low MSE/MAE) suggests that the model fits the training data well. The train score indicates how effectively the model captures the patterns and relationships present in the training data. Achieving a high train score doesn't guarantee good performance on unseen data; an excessively high train score might indicate over-fitting.

During the training process of a regressor model, a chosen loss function (such as Mean Squared Error, Mean Absolute Error, etc.) quantifies the model's error or deviation from the actual target values. In each iteration or epoch of training, the model makes predictions on the training dataset. The predictions are compared with the actual target values from the training dataset, and the loss function computes the discrepancy or error. The train loss is the cumulative or average value of this loss across the entire training dataset or within a training batch. The train loss indicates how well the model is fitting the training data by measuring the average error between predicted and actual values during training. A decrease in train loss over epochs suggests the model is improving in fitting the training data.

Since the training and test data is more suitable to be interpreted as discrete values, overall evaluation scores of the prediction of the regression models is relatively low to the classification methods. To overcome this issue, the output (prediction of the regression models) was discretized with thresholding methods. For example, the model, trained to predict propeller existence was set to 0.35. Means that, if the output of the model is less than this value, it is interpreted as a propeller that does not exist. Any values above that indicate propeller existence. Similarly, since if there is no propeller, the diameter output should be zero at the corresponding model, since the output were thresholded with 3. The output values are also rounded to the nearest integers since almost all the commercial drone propellers have a diameter as inches as integers. The

throttle level is a manual input from the user or an autopilot that is a decimal value. So, output of this value was not thresholded nor discretized.

5. EXPERIMENTAL RESULTS

In this chapter, the results of the experiments were discussed. During initial models, trained a single model that takes 4096 uint32 values of Doppler samples (range and doppler FFT values) and outputs three values such as propeller existence, diameter of the propeller and thrust level. As seen at the multi output model section, it has seen that it performs poorly, especially on the thrust level estimation. To overcome this issue, separate models were trained for propeller existence, diameter and thrust level individually as a member of ensemble network model. It has seen that, it performs much better than the multi output model, shown at the following corresponding section.

5.1. Multi Output Model

This model is developed as a 4096 Doppler samples and outputs (predicts) three values as propeller existence, diameter of the propeller and thrust level. MLP regressor, KNN regressor and linear regressor models were investigated to find which model suits best to the Doppler data and predicts the required outputs.

5.1.1. MLP regressor

During MLP regressor training, hidden layer size is chosen as 3×100 means that it consists of 3 hidden layers with 100 neurons at each layer. Note that the input layer consists of 4096 neurons and output layer consists of 3 neurons which receives the Doppler data and outputs propeller presence, size and thrust level, respectively. During the training sessions, activation function was chosen as ReLu, and optimizer (solver) was chosen as Adam. Maximum iteration of this learning was constrained to the 35 iterations. The results are shown and in the following plots.

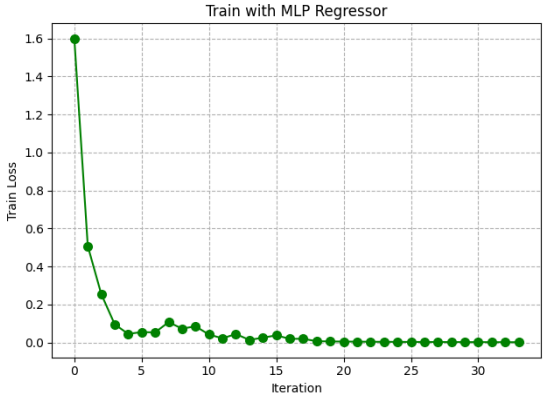

Figure 5.1. *Training loss graph for multi output MLP regressor model*

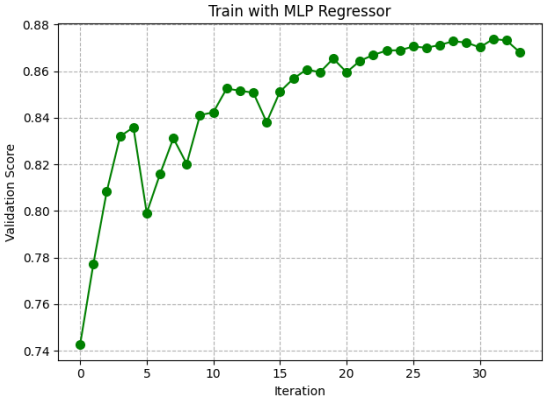

Figure 5.2. *Validation score of multi output MLP regressor model*

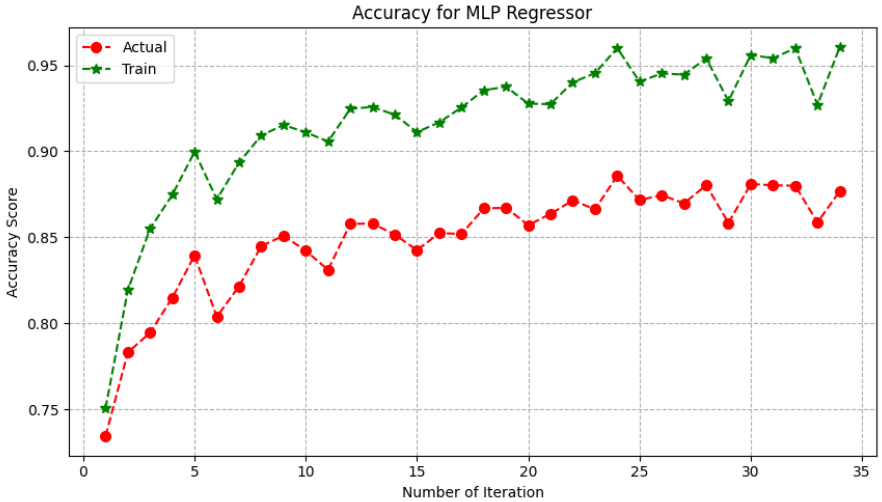

Figure 5.3. *Comparison of overall accuracy score for test and train data of MLP regressor model*

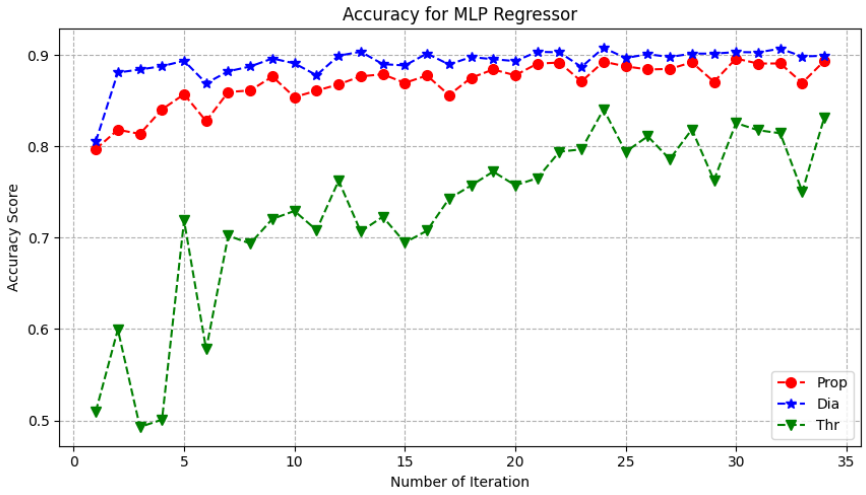

Figure 5.4. *Accuracy score of MLP regressor for propeller presence, propeller size and throttle level on test data*

As seen in the figures, best propeller accuracy was reached at the 30th iteration with a score of 0.896, best diameter and thrust level accuracy were reached at the 24th iteration with scores of 0.908 and 0.840, respectively. Also, best train accuracy was at the 34th iteration with 0.961 and best test accuracy was at the 24th iteration with a value of 0.886. Although these scores are tempting and might be considered as good, as seen at the following chapters, ensemble model outperforms this MLP regressor multi-output model.

5.1.2. KNN regressor

In KNN Regressor model, the input consisted of 4096-dimensional Doppler data, and the output consisted of 3-dimensional, which takes the Doppler data and extracts the outputs propeller presence, size and thrust level, respectively. During the KNN Regressor training, only the number of neighbors was changed during the learning step. For this model, MSE and test accuracy were calculated at each k values. The results are shown and in the following plots.

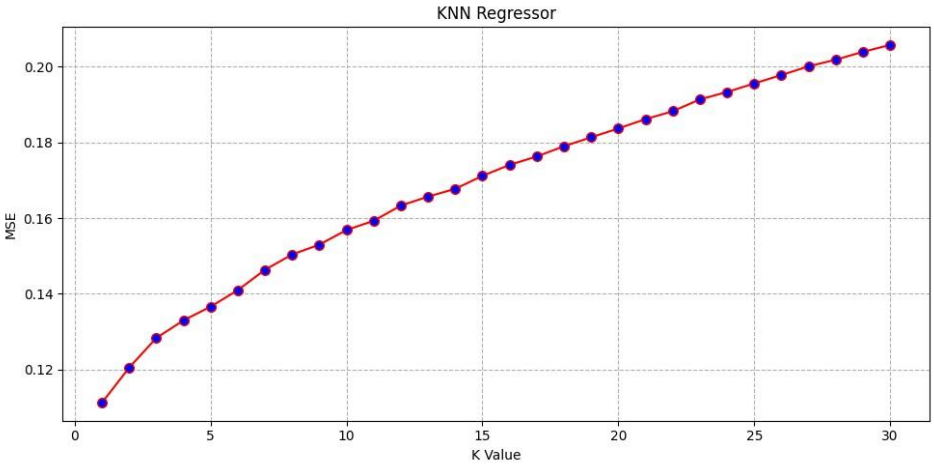

Figure 5.5. *MSE of training for multi output KNN regressor model*

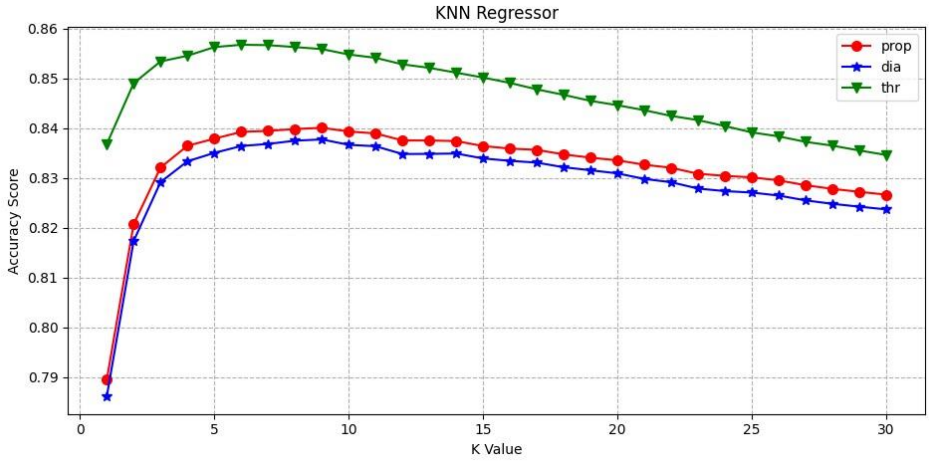

Figure 5.6. *Accuracy score of KNN regressor for propeller presence, propeller size and throttle level on test data*

As seen in the figures, the best propeller and diameter accuracy were reached when k value was chosen as 8 with scores of 0.841 and 0.837, respectively. Also, the best throttle level accuracy was reached when k value was chosen as 6 with score of 0.855.

5.1.3. Linear regressor

In Linear Regressor model, the input consisted of 4096-dimensional Doppler data, and the output consisted of 3-dimensional, which takes the Doppler data and extracts the outputs propeller presence, size and thrust level, respectively. During the Linear

Regressor training, only the training data size was changed. The epoch size was 1000 samples and trainings were done up to 25000 data step by step. For this model, MSE and test accuracy were calculated at each sample for model. The results are shown and in the following plots.

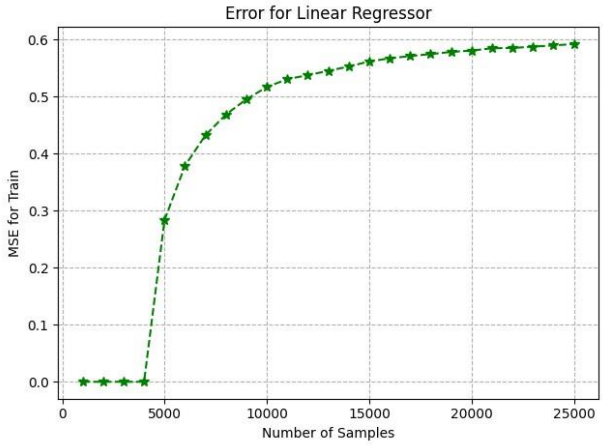

Figure 5.7. *MSE of training for multi output Linear regressor model*

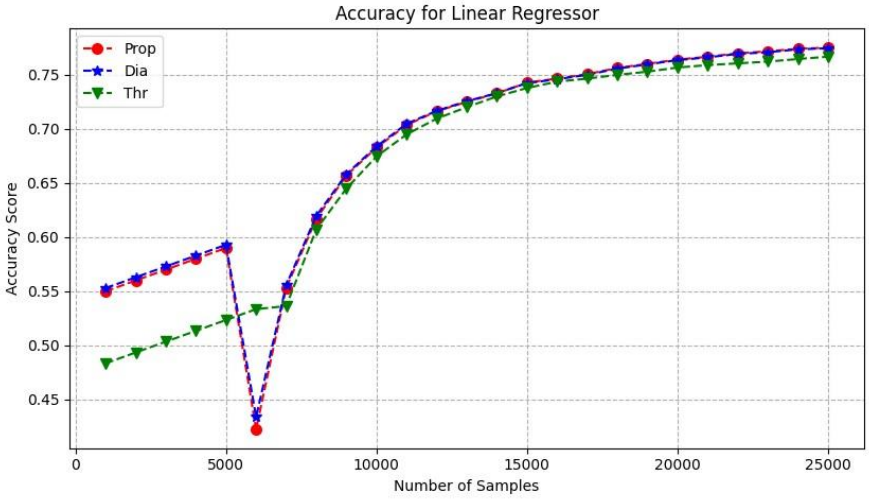

Figure 5.8. *Accuracy score of Linear Regressor for propeller presence, propeller size and throttle level on test data*

As seen in the figures, the best propeller, diameter and throttle level accuracy were reached when train dataset size was chosen as 25000 with a scores of 0.781, 0.778 and 0.769, respectively.

5.2. Propeller Model

For this model, classifier and regressor models was tested. Results of different models were shown at the corresponding sections.

5.2.1. Classifier models

In this section, MLP, KNN and different SVM Classifier models were tested.

5.2.1.1. MLP classifier

During MLP classifier training, the input layer consisted of 4096 neurons and the output layer consisted of 1 neuron, which takes the Doppler data and extracts the presence of the propeller. The maximum iteration of this learning was limited to 25 iterations. The results are shown in the graphs below. Initially, to decide on the activation function and optimizer, the hidden layer was kept constant and selected as 1×100. Choosing the hidden layer size as 1×100 means that it consists of 1 hidden layer with 100 neurons in each layer. Training was carried out by changing activation functions and optimizers respectively. Validation score and training loss were calculated at each iteration for all models. The results are shown and in the following figures and Table 5.1.

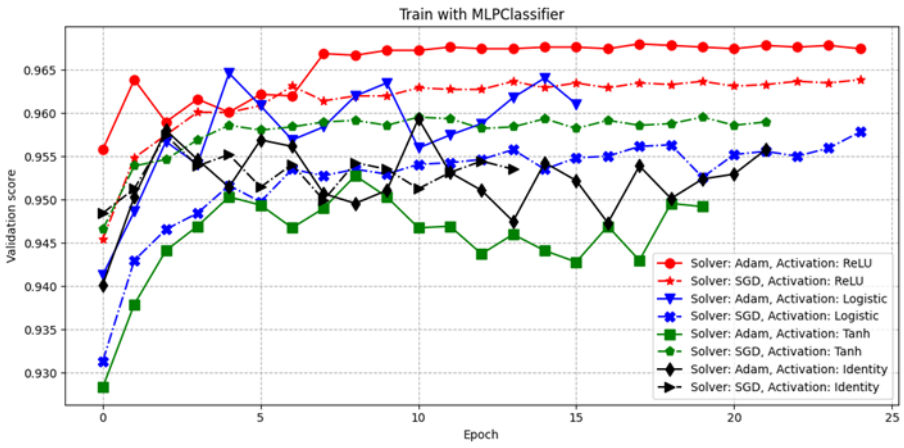

Figure 5.9. *Validation score of training for MLP classifier using different solvers and activation functions*

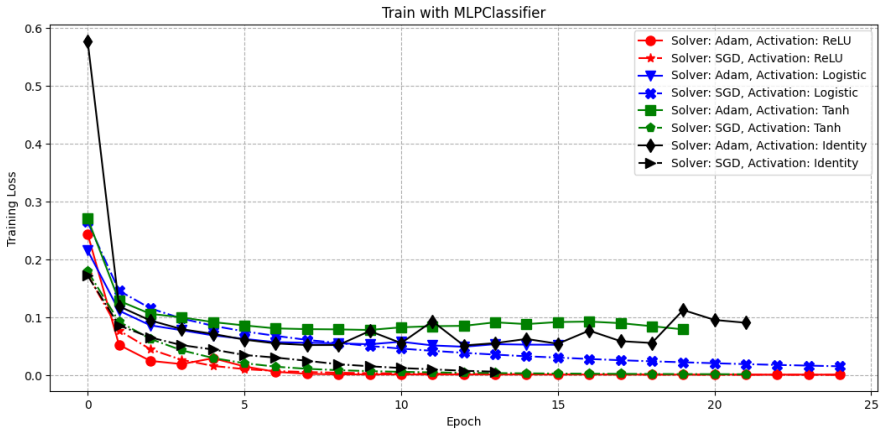

Figure 5.10. *Training loss for MLP classifier using different solvers and activation functions*

Table 5.1. *Comparison of MLP Classifier models performance with different solvers and activation functions*

Activation	Solver	Train Score	Train Loss	Test Score
ReLu	Adam	0,993563	0,001332	0,964619
ReLu	SGD	0,992773	0,000958	0,961608
Logistic	Adam	0,976737	0,05286	0,954532
Logistic	SGD	0,991455	0,01574	0,955736
Tanh	Adam	0,967703	0,07947	0,949112
Tanh	SGD	0,991907	0,001986	0,954983
Identity	Adam	0,982082	0,090914	0,951069
Identity	SGD	0,980501	0,00643	0,948811

According to these results, the best test score was obtained when the activation function was chosen as "ReLu", and the optimizer was chosen as "Adam". After choosing the activation function and optimizer, training was carried out by changing the number of hidden layers and hidden layer size. Validation score and training loss were calculated at each iteration for all models. The results are shown and in the following plots and Table 5.2.

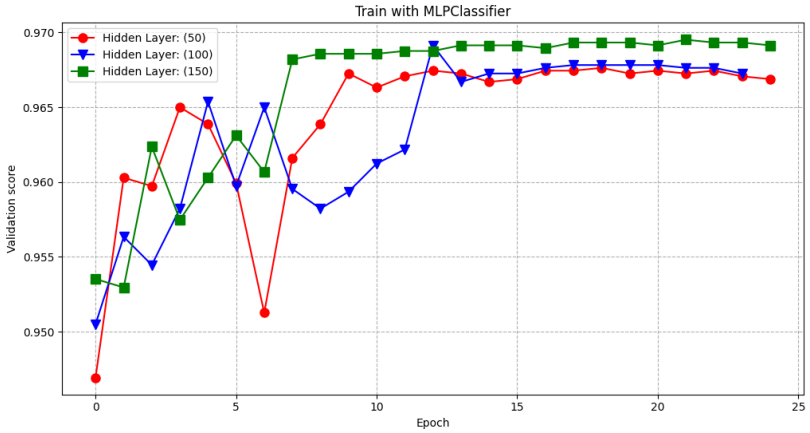

Figure 5.11. *Validation score of training for MLP classifier with one hidden layer different number of neurons*

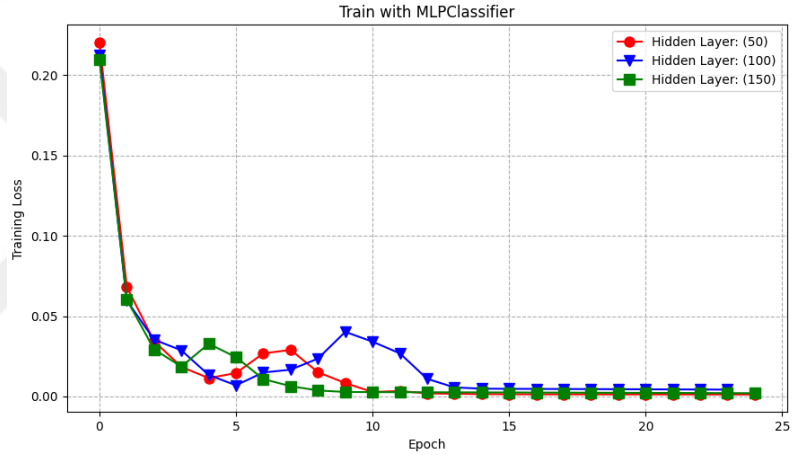

Figure 5.12. *Training loss for MLP classifier with different hidden layer size*

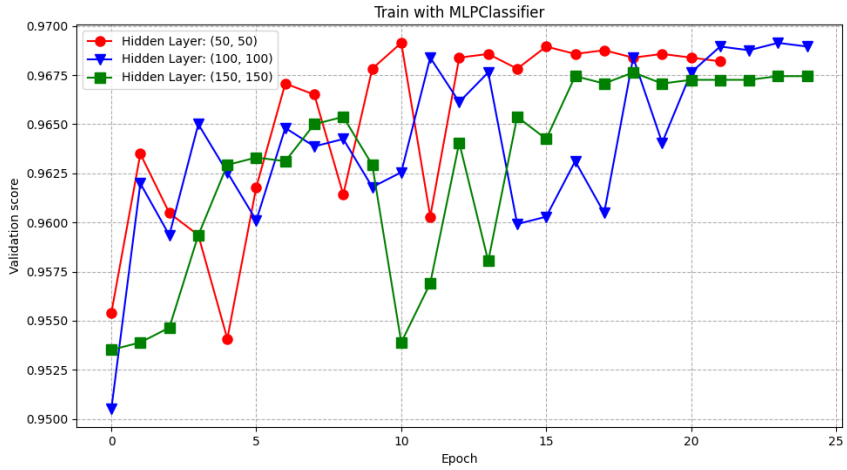

Figure 5.13. *Validation score of training for MLP classifier with two hidden layer different number of neurons*

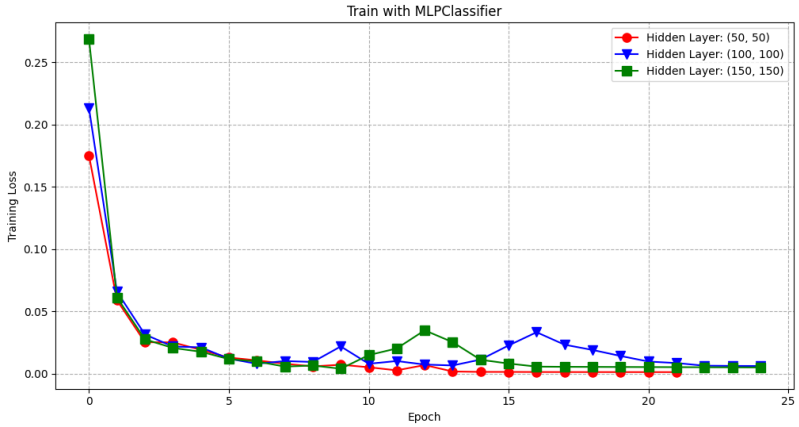

Figure 5.14. *Training loss for MLP classifier with two hidden layer different number of neurons*

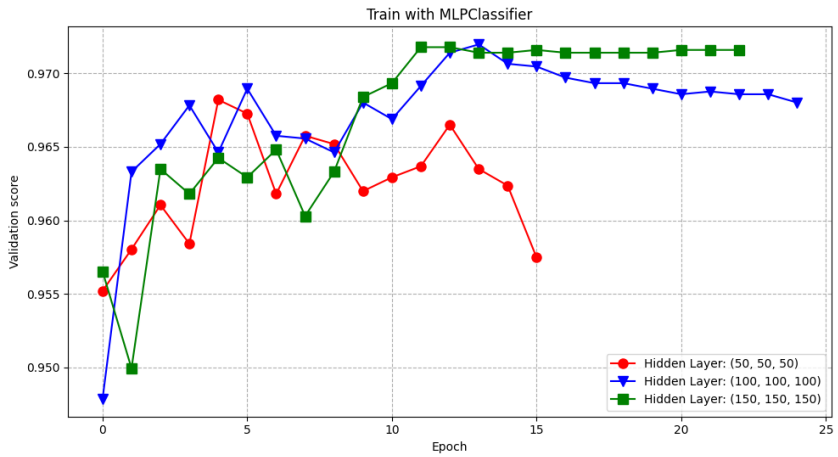

Figure 5.15. *Validation score of training for MLP classifier with three hidden layer different number of neurons*

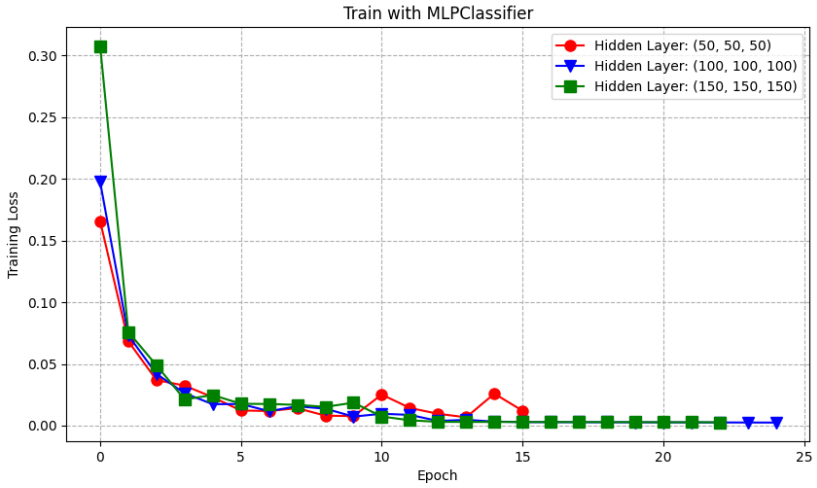

Figure 5.16. *Training loss for MLP classifier with three hidden layer different number of neurons*

Table 5.2. *Comparison of MLP Classifier models performance with different hidden layer sizes and different number of neurons*

Hidden Layer	Train Score	Train Loss	Test Score
(50)	0,993526	0,001224	0,962511
(100)	0,993714	0,004276	0,964167
(150)	0,993902	0,002038	0,965673
(50,50)	0,993714	0,001251	0,965974
(100,100)	0,993827	0,006141	0,965071
(150,150)	0,993526	0,004989	0,965974
(50,50,50)	0,990326	0,01203	0,964017
(100,100,100)	0,994391	0,002446	0,964469
(100,100,100)	0,994354	0,002658	0,963415

According to these results, the best test score was obtained when the hidden layer size was chosen as 1×150. The confusion matrix of the model trained with the chosen parameters was plotted using the test data set, as shown in the following figure.

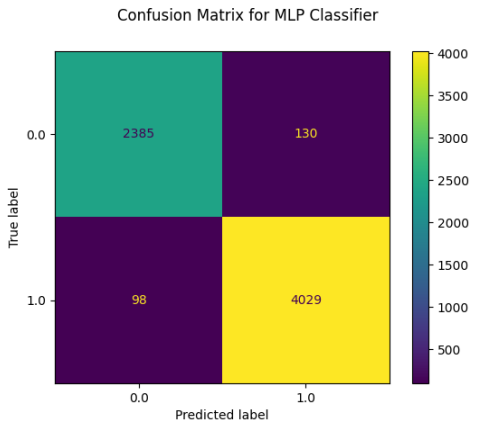

Figure 5.17. *Confusion matrix of the best MLP Classifier model among others*

5.2.1.2. KNN classifier

During KNN classifier training, the input consisted of 4096-dimensional Doppler data, and the output consisted of 1-dimensional, which takes the Doppler data and extracts the presence of the propeller. During the experiments, number of neighbors parameters is the variable to observe which model suits best to the Doppler data. The generated models were evaluated with training accuracy, test loss and test accuracy. Training was carried out by increasing the number of neighbors (k value) parameter one by one up to 30. Training accuracy, testing loss and testing accuracy were calculated at each k value for all models. The results are shown and in the following plots.

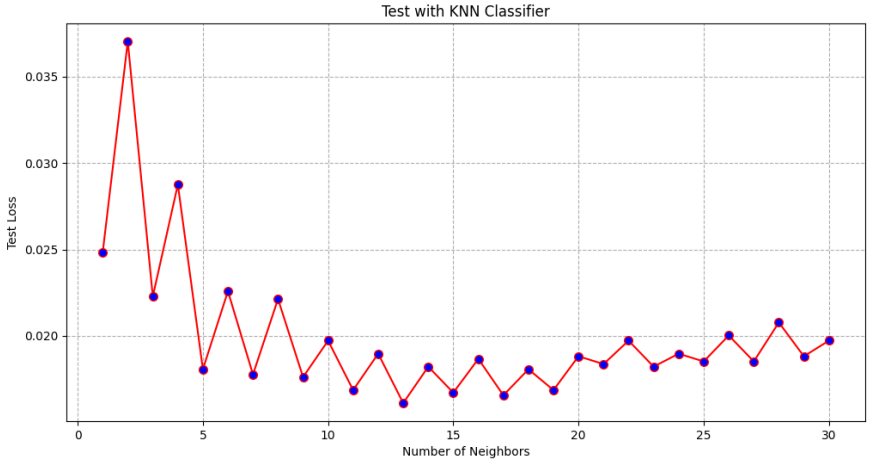

Figure 5.18. *Test loss of KNN classifiers with different number of neighbors*

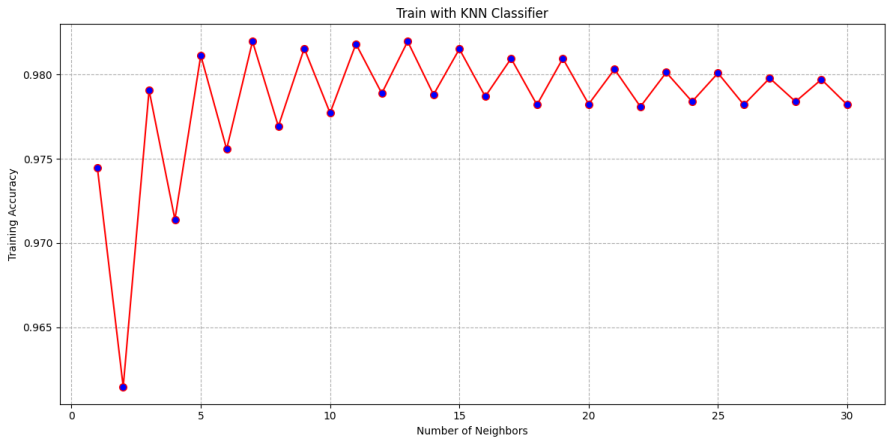

Figure 5.19. *Training accuracy of KNN classifiers with different number of neighbors*

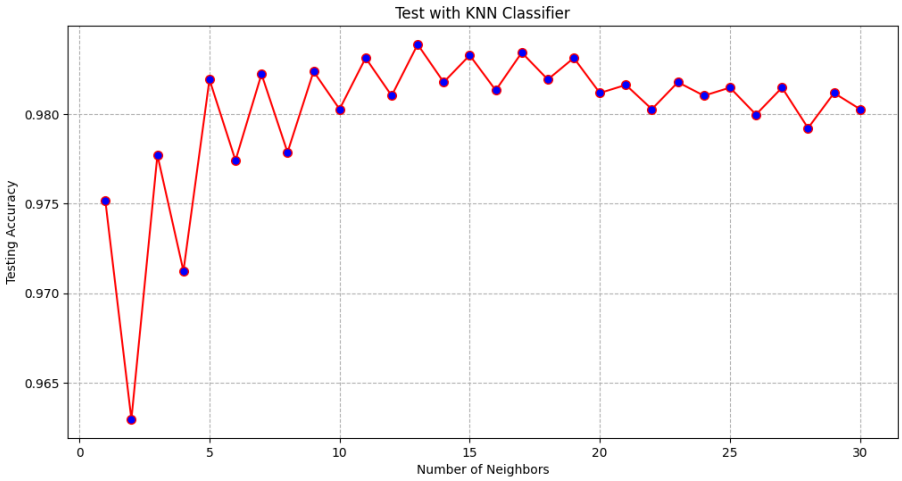

Figure 5.20. *Testing accuracy of KNN classifiers with different number of neighbors*

As seen in the figures, best training accuracy was reached at the k value of 13 with a score of 0.981, best testing accuracy was reached at the k value of 13 with a score of 0.983 and the minimum training loss was reached at the k value of 13 with a score of 0.016.

According to these results, the best test score was obtained when the number of neighbors was chosen as 13. The confusion matrix of the model trained with the chosen parameter was plotted using the test data set can be seen in the Figure 5.21.

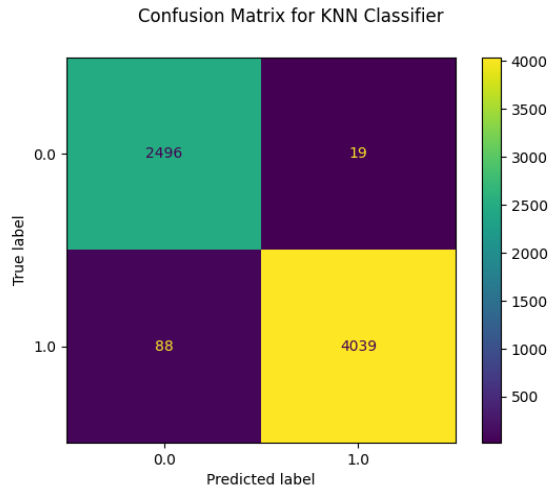

Figure 5.21. *Confusion matrix of the best KNN Classifier model among others*

5.2.1.3. SVM support vector classifier

During the SVM SVC model training session, no parameter was changed except the training data size. The epoch size was 1000 samples and trainings were done up to 25000 data step by step. F1 score, accuracy, precision, and recall were calculated at each sample for model. The results are shown and in the following plots.

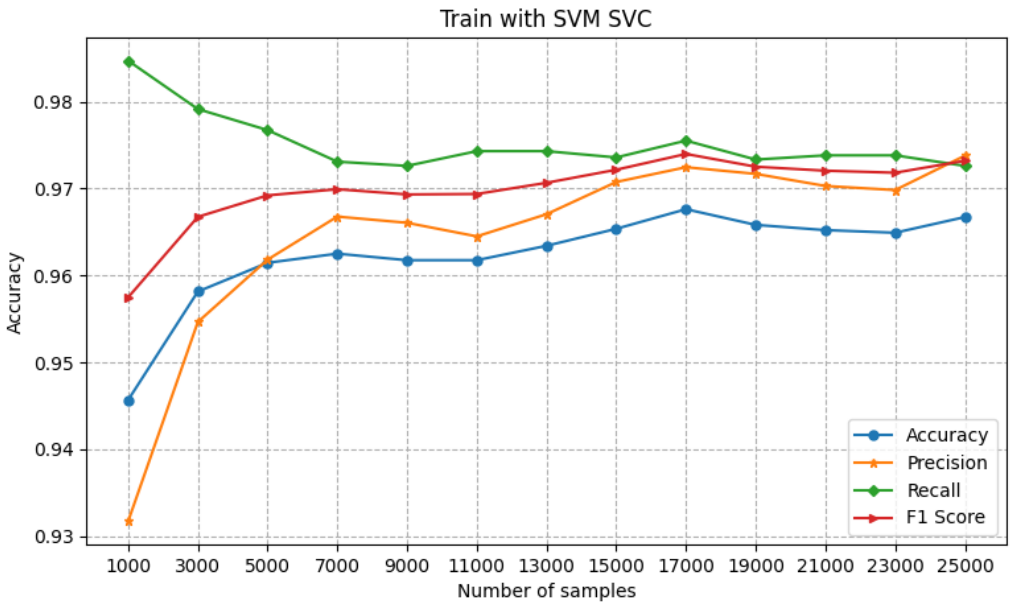

Figure 5.22. *Test accuracy of SVM SVC model with different number of samples*

The most important of these metrics is F1 score. The main reason for using the F1 Score value is to avoid making a wrong model selection in unevenly distributed data sets. In addition, F1 Score is very important because a measurement metric that includes not only False Negative or False Positive but also all error costs is needed. Therefore, according to these results, the best F1 score was obtained when train dataset size was chosen as 17000 with a score of 0.973.

5.2.1.4. SVM linear support vector classifier

During the SVM Linear SVC model training session, also no parameter was changed except the training data size. The epoch size was 1000 samples and trainings were done up to 25000 data step by step. F1 score, accuracy, precision, and recall were calculated at each sample for model. The results are shown and in the following plots.

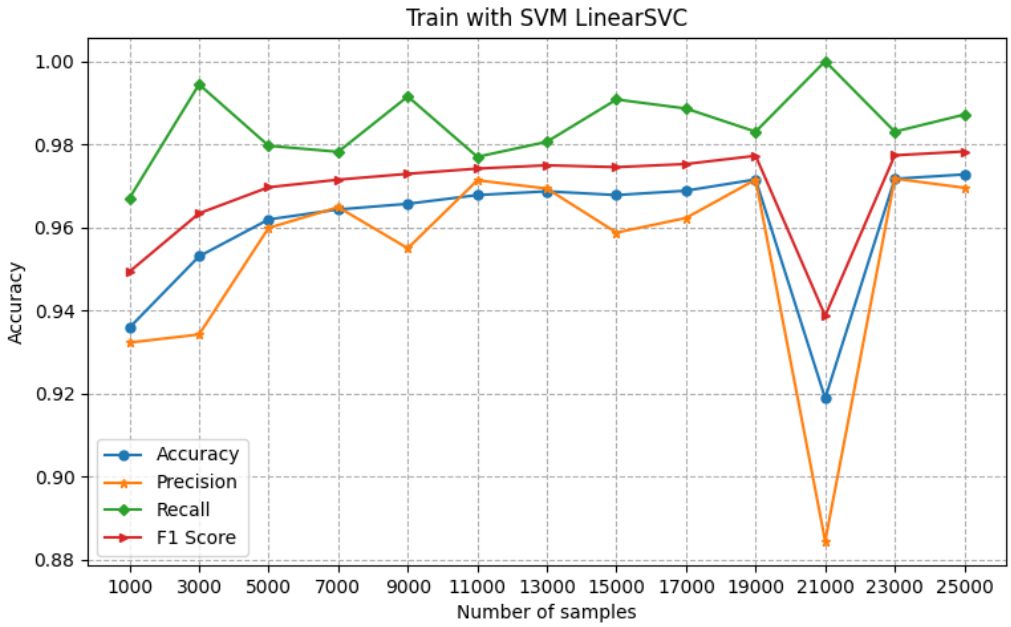

Figure 5.23. *Test accuracy of SVM linear SVC model with different number of samples*

As seen in the figure, best F1 score was reached at the train data set length of 25000 with a score of 0.975.

5.2.1.5. SVM nu support vector classifier

During the SVM Nu SVC model training session, no parameter was changed except the training data size. The epoch size was 1000 samples and trainings were done up to 25000 data step by step. F1 score, accuracy, precision, and recall were calculated at each sample for model. The results are shown and in the following plots.

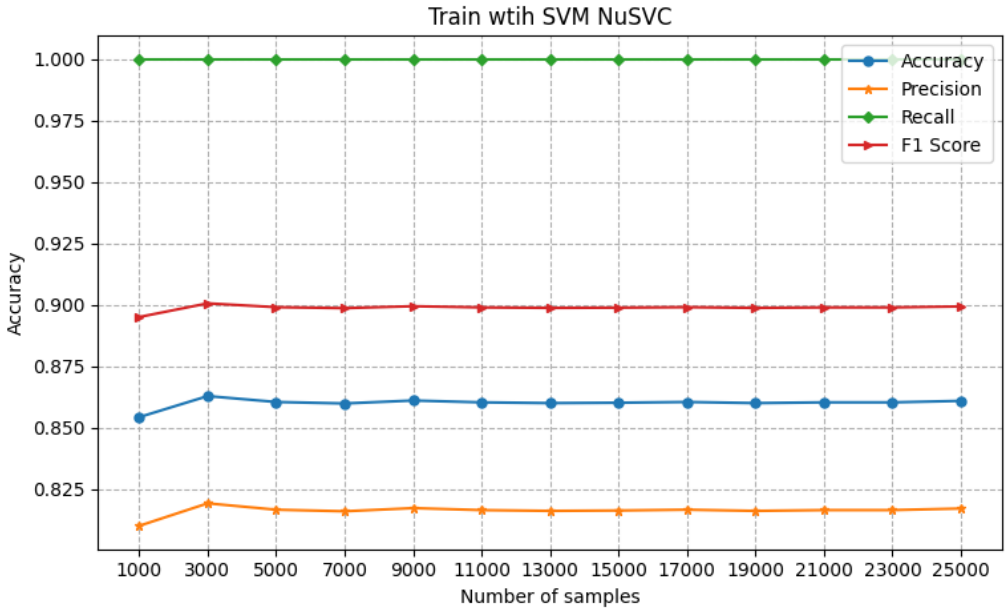

Figure 5.24. *Test Accuracy of SVM Nu SVC Model with different number of samples*

According to these results, increasing the training dataset was not affect model performance after 3000. The best F1 score was obtained when train dataset size was chosen as 3000 with a score of 0.90.

5.2.2. Regressor models

In this section, MLP, KNN and different Linear regressor models were tested for propeller presence estimation.

5.2.2.1. MLP regressor

During MLP regressor training, hidden layer size is chosen as 3×100 means that it consists of 3 hidden layers with 100 neurons at each layer. Note that the input layer consists of 4096 neurons and output layer consists of 1 neuron which receives the Doppler data and output propeller presence. During the training sessions, activation function was chosen as ReLu, and optimizer (solver) was chosen as Adam. Maximum iteration of this learning was constrained to the 35 iterations. The results are shown and in the following plots.

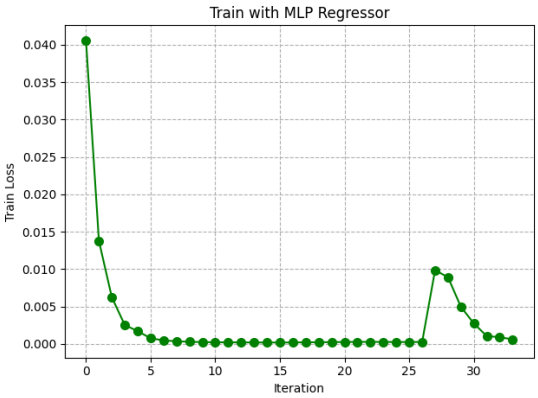

Figure 5.25. *Training loss of MLP regressor for propeller presence*

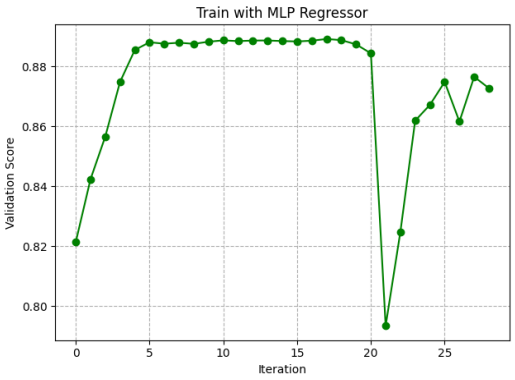

Figure 5.26. *Validation score of MLP regressor for propeller presence*

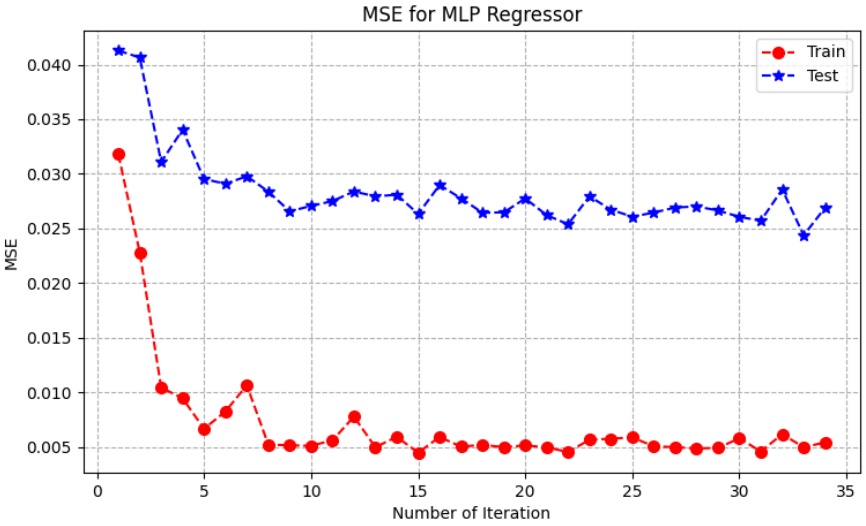

Figure 5.27. *MSE for test and train of MLP regressor for propeller presence*

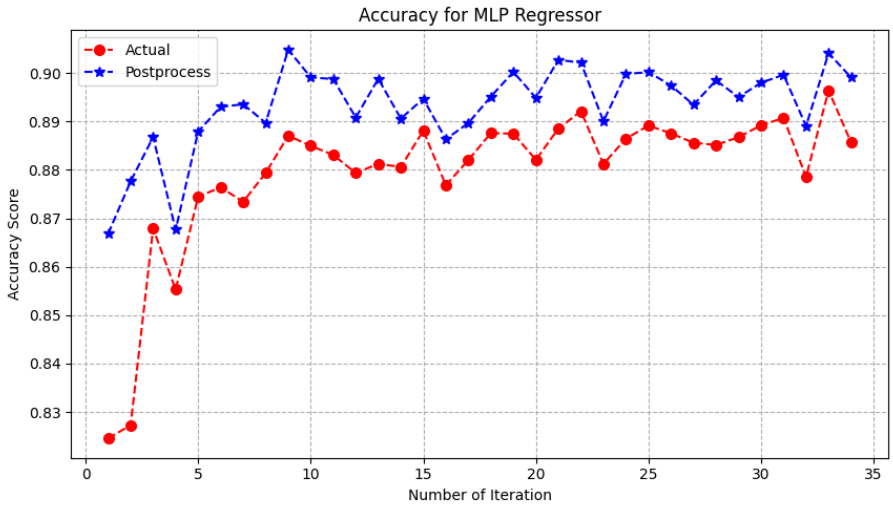

Figure 5.28. *Accuracy score with and without postprocessing of MLP regressor for propeller presence on test data*

As seen figures, the best test accuracy was obtained as 0.895 when the number of iterations was chosen as 33 without applying a threshold to the predicted value. Test accuracy improved after thresholding the predicted value. The final test accuracy of the propeller presence was recorded as 0.905.

5.2.2.2. KNN regressor

In KNN Regressor model, the input consisted of 4096-dimensional Doppler data, and the output consisted of 1-dimensional, which takes the Doppler data and extracts the presence of the propeller. During the KNN Regressor training, only the number of neighbors was changed during the learning step. For this model, MSE and test accuracy were calculated at each k values. The results are shown and in the following plots.

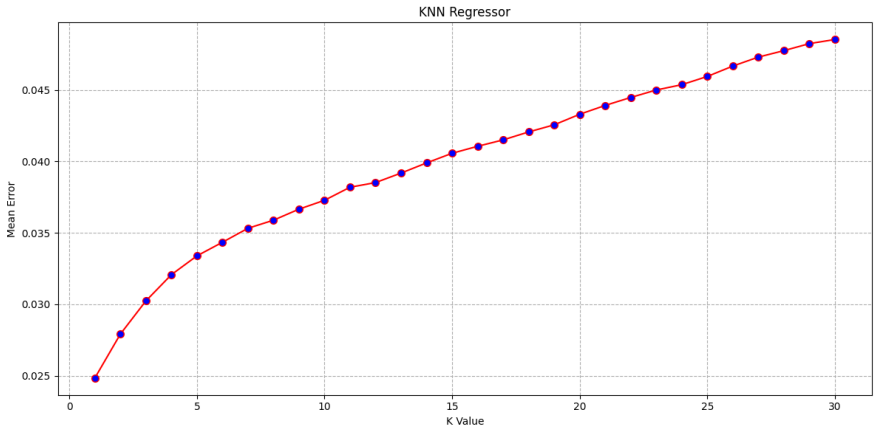

Figure 5.29. *MSE of KNN regressor with different number of neighbors for propeller presence*

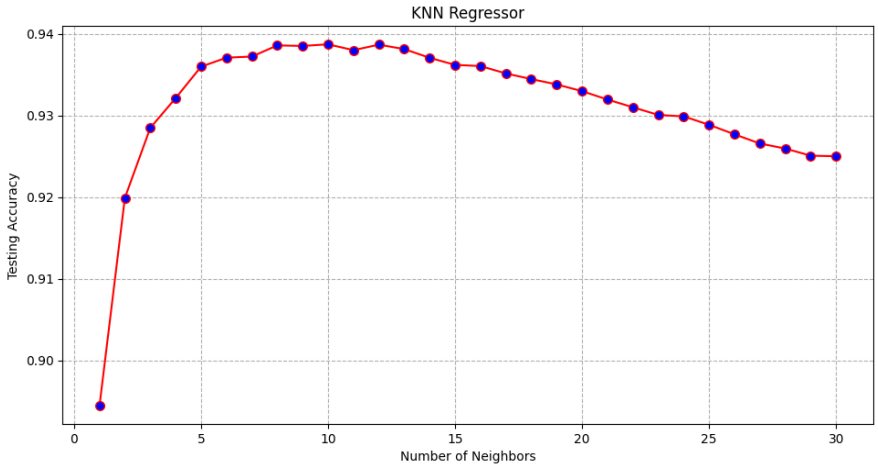

Figure 5.30. *Testing accuracy of KNN regressor with different number of neighbors for propeller presence*

According to these results, the best test accuracy was obtained when k value was chosen as 10 with a value of 0.939.

5.2.2.3. Linear regressor

In Linear Regressor model, the input consisted of 4096-dimensional Doppler data, and the output consisted of 1-dimensional, which takes the Doppler data and extracts the output propeller presence. During the Linear Regressor training, only the training data size was changed. The epoch size was 1000 samples and trainings were done up to 25000 data step by step. For this model, MSE and test accuracy were calculated at each sample for model. The results are shown and in the following plots.

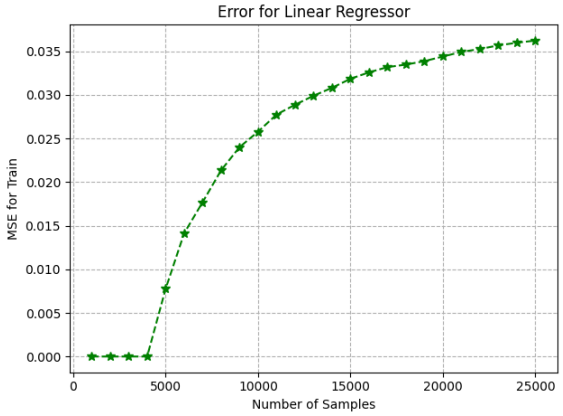

Figure 5.31. *MSE of Linear regressor with different number of samples for propeller presence*

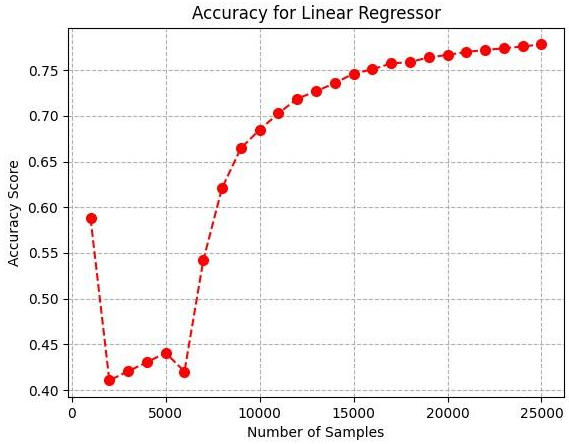

Figure 5.32. *Testing accuracy of Linear regressor with different number of samples for propeller presence*

As seen in the figures, the best propeller presence accuracy was reached when train dataset size was chosen as 25000 with a score of 0.785.

5.3. Diameter Model

For this model only the MLP regressor was used. In particular, the effect of the presence of a propeller as a new input on the model has been observed.

In the first model, hidden layer size is chosen as 3×100 means that it consists of 3 hidden layers with 100 neurons at each layer. Note that the input layer consists of 4096 neurons and output layer consists of 1 neuron which receives the Doppler data and output propeller size. During the training sessions, activation function was chosen as

ReLu, and optimizer (solver) was chosen as Adam. Maximum iteration of this learning was constrained to the 35 iterations. The results are shown and in the following plots.

Figure 5.33. *Validation score of MLP regressor for propeller size without propeller presence input*

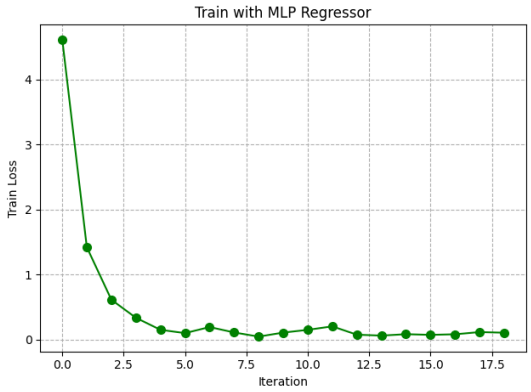

Figure 5.34. *Training loss of MLP regressor for propeller size without propeller presence input*

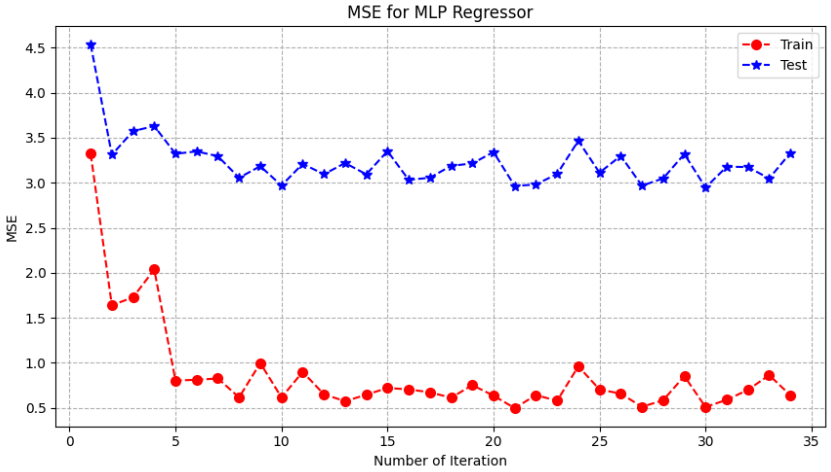

Figure 5.35. *MSE for test and train of MLP regressor for propeller size without propeller presence input*

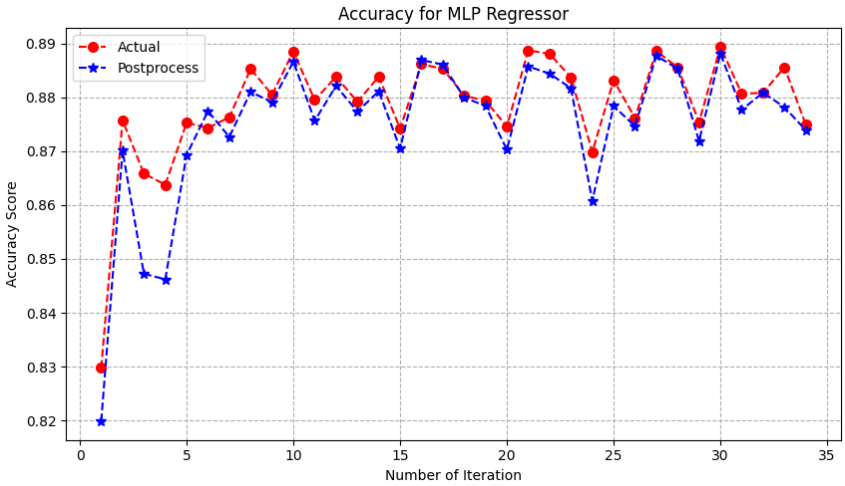

Figure 5.36. *Accuracy score of MLP regressor for propeller size with and without postprocessing without propeller presence input*

As seen in the figures, the best test accuracy was reached at the 30th iteration with a score of 0.889. Then, after using thresholding, best test accuracy was reached at the 30th iteration with scores of 0.888. Also, minimum train MSE was reached at the 21st iteration with a value of 0.496 and minimum test MSE was reached at the 30th iteration with a value of 2.94.

In the second model, only input layer size was changed. The presence of the propeller data was added to the input data. Note that the input layer consists of 4097 neurons and output layer consists of 1 neuron which receives the Doppler data and propeller presence and output propeller size. The results are shown and in the following plots.

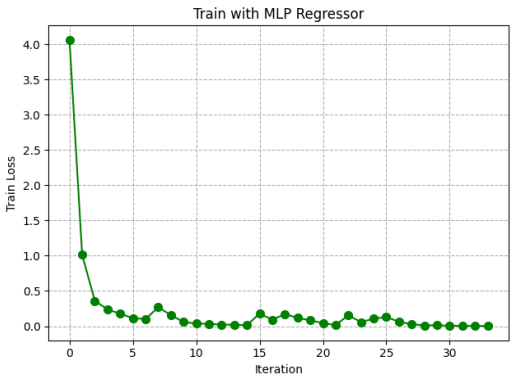

Figure 5.37. *Training loss of MLP regressor for propeller size with propeller presence input*

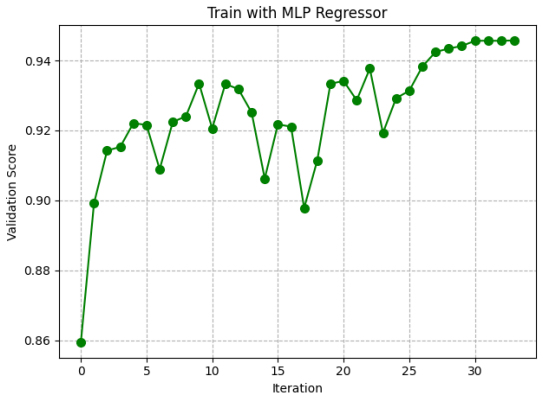

Figure 5.38. *Validation score of MLP regressor for propeller size with propeller presence input*

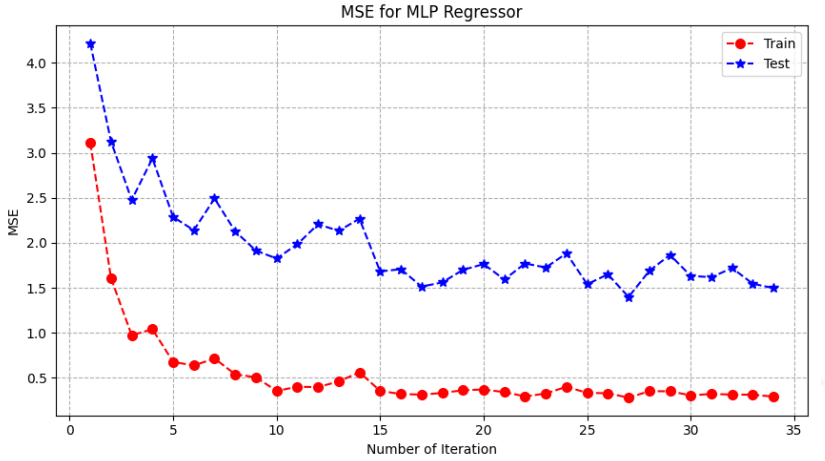

Figure 5.39. *MSE for test and train of MLP regressor for propeller size with propeller presence input*

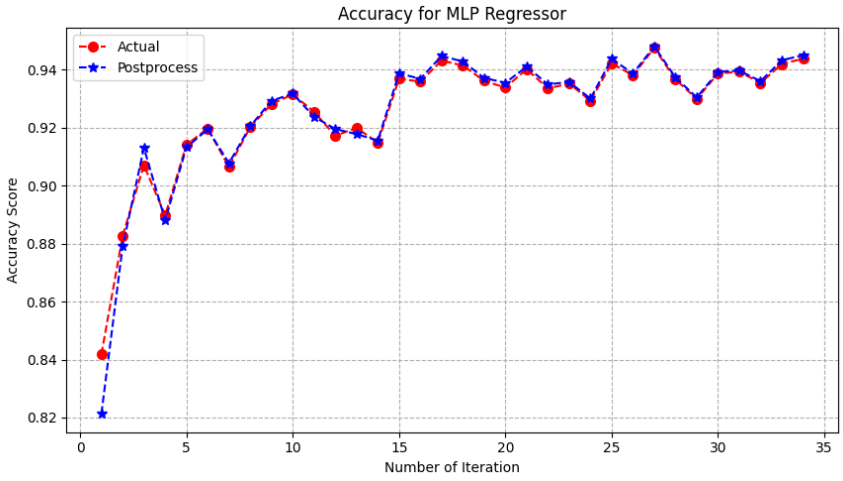

Figure 5.40. *Accuracy Score of MLP regressor for propeller size with and without postprocessing with propeller presence input*

As seen in the figures, the best test accuracy was reached at the 27th iteration with a score of 0.947. Then, after using thresholding, best test accuracy was reached at the 27th iteration with scores of 0.948. Also, minimum train MSE was reached at the 27th iteration with a value of 0.280 and minimum test MSE was reached at the 27th iteration with a value of 1.39.

By adding the presence of the propeller data to the input data, the test success rate was improved by approximately 6%.

5.4. Thrust Model

For this model only MLP regressor was used. In particular, the effect of the presence of a propeller as a new input on the model has been observed.

In the first model, hidden layer size is chosen as 3×100 means that it consists of 3 hidden layers with 100 neurons at each layer. Note that the input layer consists of 4096 neurons and output layer consists of 1 neuron which receives the Doppler data and output thrust level. During the training sessions, activation function was chosen as ReLu, and optimizer (solver) was chosen as Adam. Maximum iteration of this learning was constrained to the 35 iterations. The results are shown and in the following plots.

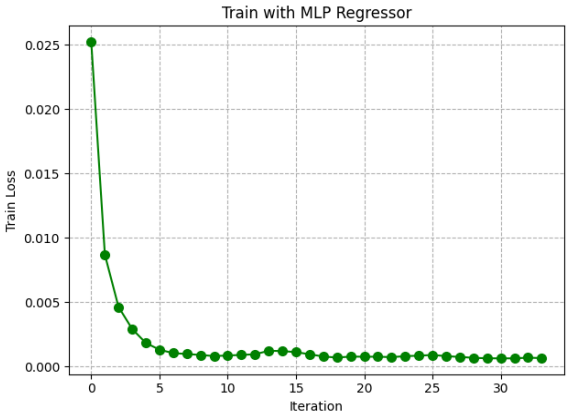

Figure 5.41. *Training loss of MLP regressor for throttle level without propeller presence input*

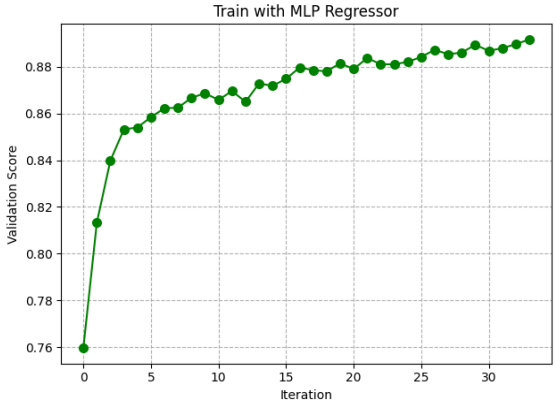

Figure 5.42. *Validation score of MLP regressor for throttle level without propeller presence input*

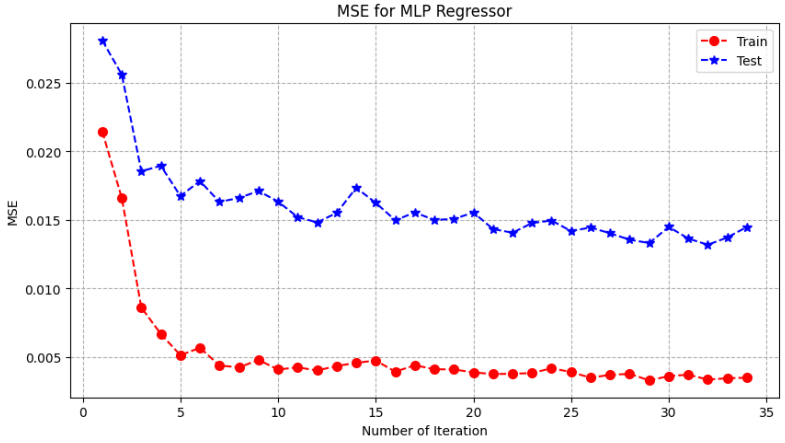

Figure 5.43. *MSE for test and train of MLP regressor for throttle level without propeller presence input*

Figure 5.44. *Accuracy score for test (actual) and train of MLP regressor for throttle level without propeller presence input*

As seen in the figures, the best test accuracy was reached at the 32nd iteration with a score of 0.901 and the best train accuracy was reached at the 29th iteration with scores of 0.975. Also, minimum train MSE was reached at the 29th iteration with a value of 0.0032 and minimum test MSE was reached at the 32nd iteration with a value of 0.0132.

In the second model, only input layer size was changed. The presence of the propeller data was added to the input data. Note that the input layer consists of 4097 neurons and output layer consists of 1 neuron which receives the Doppler data and propeller presence and output propeller size. The results are shown and in the following plots.

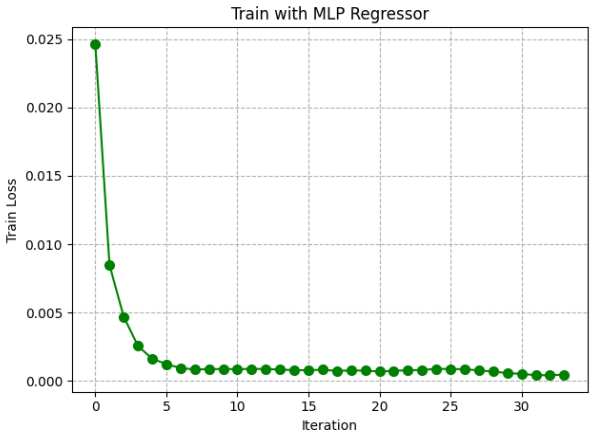

Figure 5.45. *Training loss of MLP regressor for throttle level with propeller presence input*

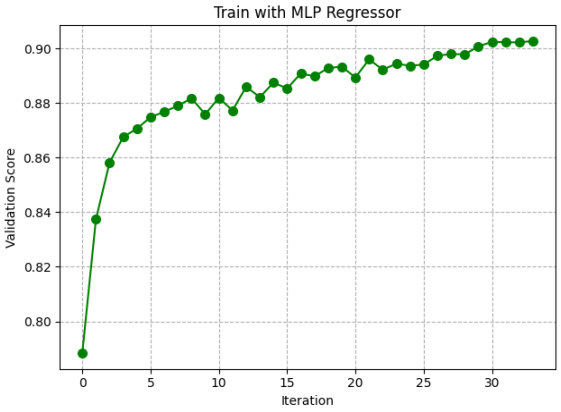

Figure 5.46. *Validation score of MLP regressor for throttle level with propeller presence input*

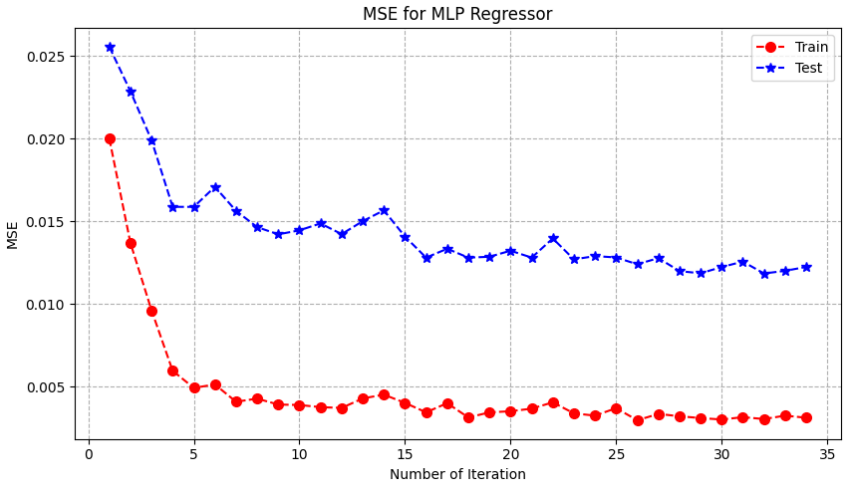

Figure 5.47. *MSE for test and train of MLP regressor for throttle level with propeller presence input*

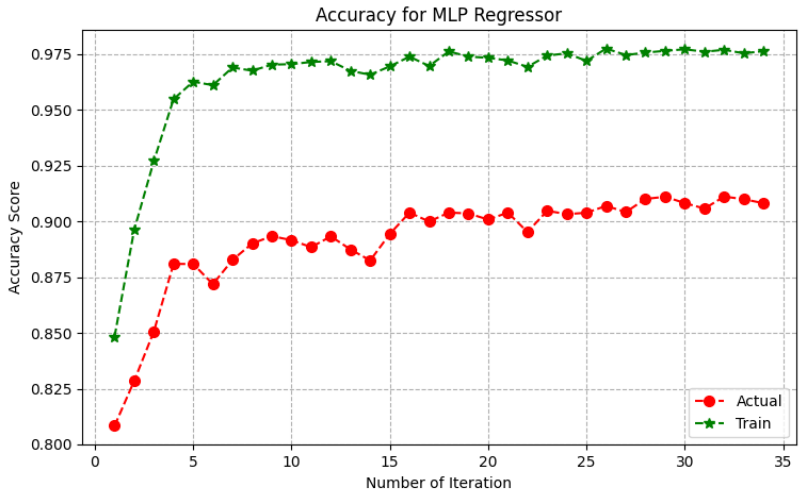

Figure 5.48. *Accuracy score for test (actual) and train of MLP regressor for throttle level with propeller presence input*

As seen in the figures, best test accuracy was reached at the 32nd iteration with a score of 0.911 and best train accuracy was reached at the 26th iteration with scores of 0.977. Also, minimum train MSE was reached at the 26th iteration with a value of 0.0029 and minimum test MSE was reached at the 32nd iteration with a value of 0.0118.

By adding the presence of the propeller data to the input data, the test success rate was improved by approximately 1% for thrust level model.

5.5. Final Proposed Ensemble Model

The final proposed model is an ensemble network model which consists of multiple learning models. The model is shown in the following figure.

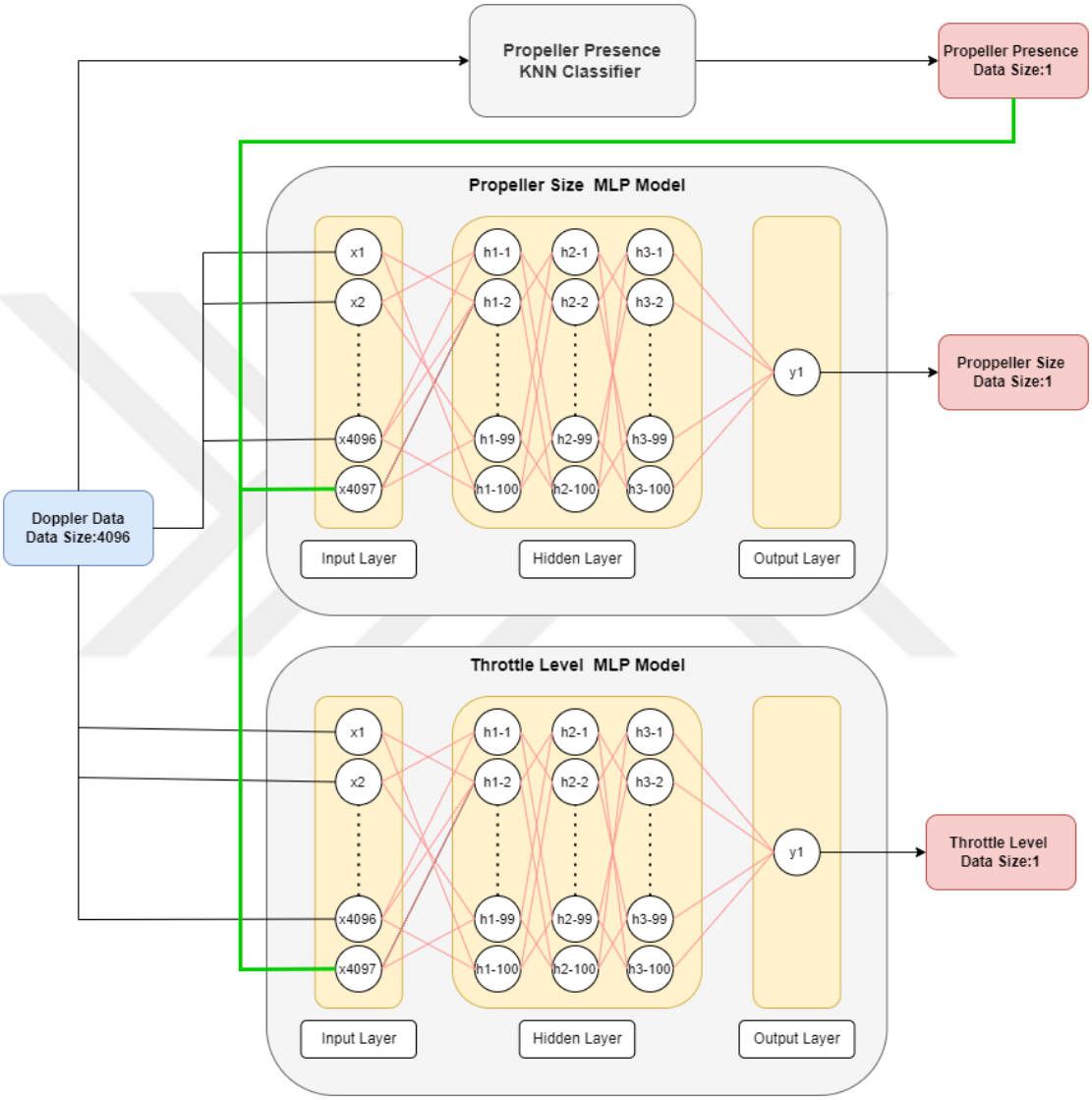

Figure 5.49. *System overview of final proposed ensemble model*

KNN Classifier was chosen for propeller existence estimation among others because it outperforms the estimation accuracy of other, as seen in the Table 5.3. Choosing the right model for this task is really important since it effects others at a critical level because it was used to fed other models (thrust level and propeller diameter estimation models) as input.

First, the Doppler data is being fed to the propeller detection model. This is a KNN Classifier previously trained network model with 4096 inputs and a single output that corresponds whether propeller does exist in this recording or not. This is the first input of the ensemble model and output of this model also fed as an input to the other models. Output of this classifier ranges from 0 to 1 indicating 0 is there is not propeller and 1 is propeller does exist in the output. Closer to those values means higher confidence level.

Table 5.3. *Comparison for test accuracy of propeller presence model*

Type of Model	**Test Accuracy**
MLP Classifier	0.965
KNN Classifier	0.981
SVM SVC	0.973
SVM Linear SVC	0.975
SVM Nu SVC	0.90
MLP Regressor	0.905
KNN Regressor	0.939
Linear Regressor	0.788

After the KNN Classifier, there are two MLP Regressor models specifically trained for propeller size and throttle level estimation, respectively. They both consists of 4097 inputs (range Doppler data and output of the propeller detection classifier) and one output (diameter size and throttle level separately). Diameter output could be from 0 to infinity. 0 indicates there is no propeller in the data. Any reasonable values indicate the propeller sizes, but too big numbers can be interpreted as invalid (model is failed to estimate the propeller size) although this was never seen during the experiments. Throttle output could be from 0 to 1. Closer to 0 values indicates zero throttle or propeller does not exist, values closer to 1 indicates full throttle (100%).

As seen in the Table 5.4, feeding the propeller size estimation model with propeller presence estimation model output significantly increases the test accuracy. But the result of applying post process to the output of this model is negligible since the inaccurate values are out of thresholding range before post processing.

Table 5.4. *Comparison for test accuracy of propeller size model*

Input	Post Process	Test Accuracy
Doppler Data	Not Applied	0.889
Doppler Data + Propeller Presence	Not Applied	0.947
Doppler Data	Applied	0.888
Doppler Data + Propeller Presence	Applied	0.948

As seen in the Table 5.5, feeding the throttle level estimation model with propeller presence estimation model output increases the test accuracy.

Table 5.5. *Comparison for test accuracy of throttle level model*

Input	Test Accuracy
Doppler Data	0.901
Doppler Data + Propeller Presence	0.911

Following table lists the test accuracy of the multi output models (first model in the experiments) and ensemble (final proposed) model.

Table 5.6. *Comparison for test accuracy of multi output and ensemble models*

Accuracy	Multi Output MLP Model	Multi Output KNN Model	Multi Output Linear Model	Ensemble Model
Proppeller Presence	0.896	0.841	0.781	0.984
Proppeller Size	0.908	0.837	0.778	0.922
Throttle Level	0.840	0.855	0.769	0.861
Overall	0.881	0.844	0.776	0.922

As seen on the above table, using ensemble model improves all the outputs, and as a result, overall score from 0.776 to 0.922, by more than 14% (w.r.t. worst case of the non-ensemble models).

6. CONCLUSION

In conclusion, the utilization of Doppler radar technology with machine learning for detecting and identifying drones via their propeller signatures represents a promising frontier in drone surveillance and security. Doppler radar systems have shown potential in capturing and analyzing the distinct frequency patterns generated by drone propellers. These advancements enable the differentiation of drone movement from other objects in the airspace.

As shown in the experimental results of this thesis, the implemented method is capable of processing Doppler radar data to extract unique propeller signatures. Aforementioned machine learning models that are developed during this thesis, play a critical role in identifying and classifying drones based on their specific frequency patterns.

Controlled experiments of this thesis have demonstrated the feasibility of using Doppler radar for drone detection and identification using machine learning methods, offering promising results in real world applications. Experimental results show that a well-trained machine learning model is able to detect and identify drone propellers with Doppler data at a greatly acceptable success rate. It is also shown that, creating an ensemble model has better predictions and success rate than the multi output single neural network model.

However, challenges and considerations stand still. Factors like wind, background noise, and variations in drone applications pose challenges to accurate and reliable detection using Doppler radar. Integrating Doppler radar technology into practical drone detection systems requires validation in real-world scenarios to assess its efficacy, reliability, and feasibility in diverse environments. Deployment of radar-based drone detection systems must adhere to regulatory guidelines, addressing concerns related to privacy, data collection, and the civilian use of radar technology.

As a result of this thesis, a 98.4% success rate in propeller recognition, a 92.2% success rate in estimating the propeller size, an 86.1% success rate in throttle level estimation was achieved. The overall accuracy score of the was measured by 92.2%. Based on these results, the proposed method and developed models can be used to detect drones and it is possible to predict and proactively do counter-measures to the safety risks that a this drone may pose.

In addition, during this thesis process, different machine learning models were developed with software and development environment that is developed in the Linux operating system. A user can use the radar module used during the thesis directly with this software development environment. It is possible to instantly monitor the Doppler data provided by this module in real-time plots.

In essence, works in this thesis shows promise, ongoing research and development efforts are needed to refine Doppler radar-based drone detection systems with machine learning methods. Advancements in signal processing, radar technology, and real-world validation will be crucial in enhancing the accuracy, reliability, and practical applicability of these systems in ensuring airspace security and safety.

REFERENCES

[1] S. Roychowdhury and D. Ghosh, "Machine Learning Based Classification of Radar Signatures of Drones," 2021 2nd International Conference on Range Technology (ICORT), Chandipur, Balasore, India, 2021, pp. 1-5.

[2] M. Jian, Z. Lu and V. C. Chen, "Drone detection and tracking based on phase-interferometric Doppler radar," 2018 IEEE Radar Conference (RadarConf18), Oklahoma City, OK, USA, 2018, pp. 1146-1149.

[3] J. Drozdowicz et al., "35 GHz FMCW drone detection system," 2016 17th International Radar Symposium (IRS), Krakow, Poland, 2016, pp. 1-4.

[4] V. G. Rizzi Varela, M. C. Brown and C. Li, "Drone Movement Detection Using V-band FMCW MIMO Radar with Digital Beamforming," 2023 16th International Conference on Advanced Technologies, Systems and Services in Telecommunications (TELSIKS), Nis, Serbia, 2023, pp. 122-125.

[5] Á. D. De Quevedo, F. I. Urzaiz, J. G. Menoyo and A. A. López, "Drone Detection and RCS Measurements with Ubiquitous Radar," 2018 International Conference on Radar (RADAR), Brisbane, QLD, Australia, 2018, pp. 1-6.

[6] P. Karpovich, S. Kareneuski and T. P. Zieliński, "Practical Results of Drone Detection by Passive Coherent DVB-T2 Radar," 2020 21st International Radar Symposium (IRS), Warsaw, Poland, 2020, pp. 77-81.

[7] M. Ezuma, O. Ozdemir, C. K. Anjinappa, W. A. Gulzar and I. Guvenc, "Micro-UAV Detection with a Low-Grazing Angle Millimeter Wave Radar," 2019 IEEE Radio and Wireless Symposium (RWS), Orlando, FL, USA, 2019, pp. 1-4.

[8] M. Caris, W. Johannes, S. Sieger, V. Port and S. Stanko, "Detection of small UAS with W-band radar," 2017 18th International Radar Symposium (IRS), Prague, Czech Republic, 2017, pp. 1-6.

[9] Y. Liu, X. Wan, H. Tang, J. Yi, Y. Cheng and X. Zhang, "Digital television based passive bistatic radar system for drone detection," 2017 IEEE Radar Conference (RadarConf), Seattle, WA, USA, 2017, pp. 1493-1497.

[10] B. Knoedler, C. Steffes and W. Koch, "Detecting and Tracking a Small UAV in GSM Passive Radar Using Track-before-Detect," 2020 IEEE Radar Conference (RadarConf20), Florence, Italy, 2020, pp. 1-6.

[11] F. Yang, K. Qu, M. Hao, Q. Liu, X. Chen and F. Xu, "Practical Investigation of a MIMO Radar System for Small Drones Detection," 2019 International Radar Conference (RADAR), Toulon, France, 2019, pp. 1-5.

[12] V. Mehta, F. Dadboud, M. Bolic and I. Mantegh, "A Deep Learning Approach for Drone Detection and Classification Using Radar and Camera Sensor Fusion," 2023 IEEE Sensors Applications Symposium (SAS), Ottawa, ON, Canada, 2023, pp. 01-06.

[13] H. Haifawi, F. Fioranelli, A. Yarovoy and R. van der Meer, "Drone Detection & Classification with Surveillance 'Radar On-The-Move' and YOLO," 2023 IEEE Radar Conference (RadarConf23), San Antonio, TX, USA, 2023, pp. 1-6.

[14] H. Lee et al., "CNN-Based UAV Detection and Classification Using Sensor Fusion," in IEEE Access, vol. 11, pp. 68791-68808, 2023.

[15] Rahman, Samiur & Robertson, Duncan. (2018). Radar micro-Doppler signatures of drones and birds at K-band and W-band. Scientific Reports.

[16] S. S. Yadav, R. Agarwal, K. Bharath, S. Rao and C. S. Thakur, "tinyRadar: mmWave Radar based Human Activity Classification for Edge Computing," 2022 IEEE International Symposium on Circuits and Systems (ISCAS), Austin, TX, USA, 2022, pp. 2414-2417.

[17] S. S. Yadav, R. Agarwal, K. Bharath, S. Rao and C. S. Thakur, "tinyRadar for Fitness: A Contactless Framework for Edge Computing," in IEEE Transactions on Biomedical Circuits and Systems, vol. 17, no. 2, pp. 192-201, April 2023.

[18] G. Galati and P. van Genderen, "History of radar: The need for further analysis and disclosure," 2014 11th European Radar Conference, Rome, Italy, 2014, pp. 25-28.

[19] M. E. Davis, "A history of battlefield surveillance radar," 2015 IEEE Radar Conference (RadarCon), Arlington, VA, USA, 2015, pp. 1345-1350.

[20] H. Rohling, "Radar CFAR Thresholding in Clutter and Multiple Target Situations," in IEEE Transactions on Aerospace and Electronic Systems, vol. AES-19, no. 4, pp. 608-621, July 1983.

[21] D. Reid, "An algorithm for tracking multiple targets," in IEEE Transactions on Automatic Control, vol. 24, no. 6, pp. 843-854, December 1979.

[22] Tie-Jun Shan and T. Kailath, "Adaptive beamforming for coherent signals and interference," in IEEE Transactions on Acoustics, Speech, and Signal Processing, vol. 33, no. 3, pp. 527-536, June 1985.

[23] S. -J. Liu, H. Luo and Q. Shi, "Active Ensemble Deep Learning for Polarimetric Synthetic Aperture Radar Image Classification," in IEEE Geoscience and Remote Sensing Letters, vol. 18, no. 9, pp. 1580-1584, Sept. 2021.

[24] Fan Hui, Huang Xingyu, Su Tao and Gao Zhonghui, "A study on the application of millimeter-wavelength cloud radar," Proceedings of 2011 IEEE CIE International Conference on Radar, Chengdu, 2011, pp. 1008-1011.

[25] R. Mehta et al., "CNN-Based Sub-Surface Object Detection Using Ground Penetrating Radar," 2021 11th International Workshop on Advanced Ground Penetrating Radar (IWAGPR), Valletta, Malta, 2021, pp. 1-5.

[26] S. M. Sabery, A. Bystrov, P. Gardner and M. Gashinova, "Surface Classification Based on Low Terahertz Radar Imaging and Deep Neural Network," 2020 21st International Radar Symposium (IRS), Warsaw, Poland, 2020, pp. 24-27.

[27] Robert Caiming Qiu; Zhen Hu; Husheng Li; Michael C. Wicks, "Machine Learning," in Cognitive Radio Communication and Networking: Principles and Practice , Wiley, 2013, pp.283-321.

[28] D. Tm, R. Verma, R. Rajesh and S. Varughese, "Single Shot Radar Target Detection and Localization using Deep Neural Network," 2022 IEEE International Conference on Electronics, Computing and Communication Technologies (CONECCT), Bangalore, India, 2022, pp. 1-9.

[29] A. T. Purnomo, K. S. Komariah, D. -B. Lin, W. F. Hendria, B. -K. Sin and N. Ahmadi, "Non-Contact Supervision of COVID-19 Breathing Behaviour With

FMCW Radar and Stacked Ensemble Learning Model in Real-Time," in IEEE Transactions on Biomedical Circuits and Systems, vol. 16, no. 4, pp. 664-678, Aug. 2022.

APPENDIX

1. Configurations for IWR1843

```
% ***************************************************************
% Created for SDK ver:03.05
% Created using Visualizer ver:3.6.0.0
% Frequency:77
% Platform:xWR18xx
% Scene Classifier:best_range_res
% Azimuth Resolution(deg):15 + Elevation
% Range Resolution(m):0.044
% Maximum unambiguous Range(m):9.02
% Maximum Radial Velocity(m/s):4.98
% Radial velocity resolution(m/s):0.63
% Frame Duration(msec):33.333
% RF calibration data:None
% Range Detection Threshold (dB):15
% Doppler Detection Threshold (dB):15
% Range Peak Grouping:enabled
% Doppler Peak Grouping:enabled
% Static clutter removal:disabled
% Angle of Arrival FoV: Full FoV
% Range FoV: Full FoV
% Doppler FoV: Full FoV
% ***************************************************************
sensorStop
flushCfg
dfeDataOutputMode 1
channelCfg 15 7 0
adcCfg 2 1
adcbufCfg -1 0 1 1 1
profileCfg 0 77 8 7 57.14 0 0 70 1 256 5209 0 0 30
chirpCfg 0 0 0 0 0 0 0 1
chirpCfg 1 1 0 0 0 0 0 4
chirpCfg 2 2 0 0 0 0 0 2
frameCfg 0 2 16 0 33.333 1 0
lowPower 0 0
guiMonitor -1 0 0 0 0 1 0
cfarCfg -1 0 2 8 4 3 0 15 1
cfarCfg -1 1 0 4 2 3 1 15 1
multiObjBeamForming -1 1 0.5
clutterRemoval -1 0
calibDcRangeSig -1 0 -5 8 256
extendedMaxVelocity -1 0
lvdsStreamCfg -1 0 0 0
compRangeBiasAndRxChanPhase 0.0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1
0 1 0 1 0 1 0
measureRangeBiasAndRxChanPhase 0 1.5 0.2
CQRxSatMonitor 0 3 5 121 0
CQSigImgMonitor 0 127 4
analogMonitor 0 0
aoaFovCfg -1 -90 90 -90 90
cfarFovCfg -1 0 0 8.92
cfarFovCfg -1 1 -4.98 4.98
calibData 0 0 0
sensorStart
```

2. Prepare The Data for Training

```
import glob
import pathlib
import csv
import numpy as np

# list all the files in the directory
data_dir = pathlib.Path("../data/")

# get all the files in the directory matches the pattern m{}_p{}_d{}_a{}_t{}.csv
file_list = list(data_dir.glob("m*_p*_d*_a*_t*.csv"))

# print the file list
expected = {}
for file in file_list:
    kv, dia, dist, ang, thr = [float(x[1:]) for x in file.stem.split("_")]
    expected[file.stem] = {}
    expected[file.stem]["prop"] = 1.0
    expected[file.stem]["kv"] = float(kv)
    expected[file.stem]["dia"] = int(float(dia) * 1e-2)
    expected[file.stem]["dist"] = float(dist)
    expected[file.stem]["ang"] = float(ang)
    expected[file.stem]["thr"] = float(thr) * 1e-2

    # check if thr is 0
    if expected[file.stem]["thr"] == 0.0:
        # all data is 0
        expected[file.stem]["prop"] = 0.0
        expected[file.stem]["kv"] = 0.0
        expected[file.stem]["dia"] = 0.0
        expected[file.stem]["dist"] = 0.0
        expected[file.stem]["ang"] = 0.0
        expected[file.stem]["thr"] = 0.0

# get all the data from the files
all_data = []
for file in file_list:
    with open(file, "r") as f:
        reader = list(csv.reader(f))
        all_exp = [expected[file.stem]["prop"], expected[file.stem]["dia"], expected[file.stem]["thr"]]
        for read in reader:
            all_data.append([int(x) for x in read] + all_exp)

# split the data into train and test
train_weight = 0.8
all_data = np.array(all_data)
np.random.shuffle(all_data)
train_all_data = all_data[:int(train_weight * len(all_data))]
test_all_data = all_data[int(train_weight * len(all_data)):]
print("train all data:", len(train_all_data))
print("test all data:", len(test_all_data))

# save the all data
np.save("train_all_data.npy", train_all_data)
np.save("test_all_data.npy", test_all_data)
```

3. Train and Test The Final Ensemble Model

```
import numpy as np
from sklearn.neighbors import KNeighborsClassifier
from sklearn.preprocessing import StandardScaler
from sklearn.neural_network import MLPRegressor
from sklearn.metrics import accuracy_score, r2_score, f1_score
from sklearn.metrics import plot_confusion_matrix
import matplotlib.pyplot as plt
import pickle


def train_prop_model(train_data_x, train_data_y, test_data_x,
test_data_y):
    prop_model = KNeighborsClassifier(n_neighbors=13)
    prop_model.fit(X=train_data_x, y=train_data_y)

    # test
    actual = test_data_y
    predict = prop_model.predict(X=test_data_x)
    print("KNN", accuracy_score(actual, predict))

    # plot confusion matrix
    fig = plot_confusion_matrix(prop_model, test_data_x,
test_data_y,
                                display_labels=prop_model.classes_)

    fig.figure_.suptitle("Confusion Matrix for KNN Classifier")
    plt.show()

    # save model to file
    with open("../model/model_knn.pkl", "wb") as file:
        pickle.dump(prop_model, file)


# train with mlp
def train_mlp_model(train_data_x, train_data_y, test_data_x,
test_data_y, max_iter=35, name="model"):
    # preprocess the data
    scaler = StandardScaler()
    train_data_x = scaler.fit_transform(train_data_x)
    test_data_x = scaler.transform(test_data_x)

    # train with MIMO neural network
    model = MLPRegressor(hidden_layer_sizes=(100, 100, 100),
max_iter=max_iter, verbose=True,
                         activation='relu', solver='adam',
                         validation_fraction=0.2,
early_stopping=True, tol=0)
    model.fit(train_data_x, train_data_y)
    print("score:", model.score(test_data_x, test_data_y))
    print(model.score(train_data_x, train_data_y))
    plt.plot(model.loss_curve_)
    plt.plot(model.validation_scores_)
    plt.show()

    # test the model
    print("score:", model.score(test_data_x, test_data_y))
```

```
        # save the model with name
        model_name = "../model/model_" + name + "_mlp.pkl"
        with open(model_name, "wb") as file:
            pickle.dump(model, file)


    # test ensemble model
    def       ensemble_model_test(test_data_x,       prop_output_test,
dia_output_test, thr_output_test, dia_threshold=0.5):
        # load the model
        with open("../model/model_knn.pkl", "rb") as file:
            prop_model = pickle.load(file)

        with open("../model/model_dia_mlp.pkl", "rb") as file:
            dia_model = pickle.load(file)

        with open("../model/model_thr_mlp.pkl", "rb") as file:
            thr_model = pickle.load(file)

        # test prop model
        prop_predict = prop_model.predict(test_data_x)

        # get prop score
        prop_score    =    f1_score(prop_output_test,    prop_predict,
average='macro')
        print("prop score:", prop_score)

        # add prop_predict to test_data_x to get test_data_processed as
4097
        test_data_processed       =       np.concatenate((test_data_x,
prop_predict.reshape(-1, 1)), axis=1)

        # preprocess the data
        scaler = StandardScaler()
        test_data_processed = scaler.fit_transform(test_data_processed)

        # test dia model
        dia_predict = dia_model.predict(test_data_processed)

        print("Before      dia      score:",      r2_score(dia_predict,
dia_output_test))

        # apply threshold to dia_predict
        for i in range(len(dia_output_test)):
            if (dia_output_test[i] - dia_threshold < dia_predict[i] <
dia_output_test[i] + dia_threshold) or (
                    dia_predict[i]  <  4.0  and  dia_output_test[i]  ==
0.0):
                dia_predict[i] = dia_output_test[i]

        # get dia score
        dia_score = r2_score(dia_predict, dia_output_test)
        print("dia score:", dia_score)

        # test thr model
        thr_predict = thr_model.predict(test_data_processed)
```

```
        # get thr score
        thr_score = r2_score(thr_predict, thr_output_test)

        print("thr score:", thr_score)


    # load the data
    train_data = np.load("./train_all_data.npy")
    test_data = np.load("./test_all_data.npy")

    # get doppler data 4096
    doppler_data_train = train_data[:, :-3]
    doppler_data_test = test_data[:, :-3]

    # get 4097 data for dia and thr model
    input_4097_train = train_data[:, :-2]
    input_4097_test = test_data[:, :-2]

    # get output data
    output_train = train_data[:, -3:]
    output_test = test_data[:, -3:]

    # get prop output
    prop_output_train = output_train[:, :1]
    prop_output_test = output_test[:, :1]

    # get dia output
    dia_output_train = output_train[:, 1:2]
    dia_output_test = output_test[:, 1:2]

    # get thr output
    thr_output_train = output_train[:, -1:]
    thr_output_test = output_test[:, -1:]

    # train prop model with knn
    train_prop_model(doppler_data_train,            prop_output_train,
doppler_data_test, prop_output_test)

    # train dia model with mlp
    train_mlp_model(input_4097_train,               dia_output_train,
input_4097_test, dia_output_test, max_iter=35, name="dia")

    # train thr model with mlp
    train_mlp_model(input_4097_train,               thr_output_train,
input_4097_test, thr_output_test, max_iter=35, name="thr")

    # test ensemble model
    ensemble_model_test(doppler_data_test,          prop_output_test,
dia_output_test, thr_output_test, dia_threshold=0.5)
```

CURRICULUM VITAE

ORCID ID **:** 0009-0000-9304-9505

Name Surname **:** Fatma ÖZÜDOĞRU

Foreign Language **:** English

Education and Professional Background:

- Eskisehir Technical University, Master's Degree, Engineering Faculty, Electrical and Electronics Engineering, Circuit and Systems Theory, 2022-Present
- Eskisehir Technical University, Engineering Faculty, Electrical and Electronics Engineering (%100 English), 2016-2021 GNO: 3.74/4, High Honour Degree
- Anadolu University, Open Education Faculty, Business 2017-2021 GNO:3.38/4, Honour Degree
- Hardware Design Engineer, Erendiz Superbilgisayar Industry and Trade Limited Company, Eskisehir, September 2021- March 2022
- Software Engineer, RE-SCIENCE Technological Consultancy Industry Trade Limited Company, Eskisehir, April 2022-(Present)

Publications and/or Scientific/Artistic Activities:

- 23rd International Symposium on Quality Electronic Design (ISQED'22), Paper: Multilayered Triboelectric Energy Harvester as a Smart Floor Mat

Awards:

- 2021 Xilinx Open Hardware Competition, Winner of Student Category
- 2021 Eskisehir Technical University 14th Project Fair and Competition, Project: Triboelectric Energy Harvester Design and Potential Sensing Applications, Rank: 1
- 2018 TÜBİTAK UAV Competition, Fixed Wing Category, Anadolu UAV Team, Rank:3