DEEP NEURAL NETWORKS ALGORITHMS FOR ACOUSTIC DRONE DETECTION

Master of Science Thesis

Hussam KANAAN

Eskişehir, 2019

DEEP NEURAL NETWORKS ALGORITHMS FOR ACOUSTIC DRONE DETECTION

Hussam KANAAN

MASTER OF SCIENCE THESIS

Electrical and Electronics Department

Supervisor: Assoc. Prof. Dr. Tansu FİLİK

Eskişehir

Anadolu University

Institute of Graduate Programs

November 2019

FINAL APPROVAL FOR THESIS

This thesis titled "Deep Neural Networks Algorithms for Acoustic Drone Detection" has been prepared and submitted by Hussam Kanaan in partial fulfillment of the requirements in "Anadolu University Directive on Graduate Education and Examination" for the Degree of Master of Science in Electrical and Electronics Engineering Department has been examined and approved on 12/11/2019

Committee Members	**Title Name Surname**	**Signature**
Member (Supervisor)	: Assoc. Prof. Dr. Tansu FİLİK	.....
Member	: Prof. Dr. Ömer Nezih GEREK	.....
Member	: Dr. Öğr. Üyesi Cahit PERKGÖZ	.....

Prof. Dr. Murat TANIŞLI

Director of Institute of Graduate Programs

ABSTRACT

DEEP NEURAL NETWORKS ALGORITHMS FOR ACOUSTIC DRONE DETECTION

Hussam KANAAN

Department of Electrical and Electronics Engineering

Anadolu University, Institute of Graduate Programs, November 2019

Supervisor: Assoc. Prof. Dr. Tansu FİLİK

In 2010, the first commercial drone was presented at Consumer Electronics Show (CES), and since this date drones are becoming increasingly popular in various industrial, commercial, and public-safety areas. However, drones can be used in several illegal activities, and they pose serious challenges especially in highly security-sensitive areas such as airports and nuclear plants. As a consequence, effective counter measures are highly required in order to detect and report a drone flying over such restricted areas.

Recent advances and fast developments in the design and implementation of deep learning models lead us to apply them in several recognition tasks such as speech, music, environmental sounds, and image recognition. Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs) are two widely used deep learning models, where RNNs show remarkable performance in several sequence data related tasks such as natural language processing applications. On the other hand, convolutional models considerably success in image classification and object recognition for computer vision tasks.

The main goal of the thesis is to develop a powerful and efficient deep learning model for acoustic drone detection. In this context, we investigate the results of the drone's sound recognition scheme based on RNNs and CNNs that are trained using our collected dataset; we also investigate the influence of acoustic features extraction techniques such as MFCCs and Mel-Scale Filter Banks on model's classification performance with different sampling rates. It is verified with various experiments that

the CNN model with Mel-scale filter banks as a feature extraction technique with 32 KHz unified sampling rate gives the best classification performance.

Keywords: Deep Learning, Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs), Acoustic Features, Mel Frequency Cepstral Coefficients MFCCs, Mel-Scale Filter Banks.

ÖZET

AKUSTİK DRONE TESPİTİ İÇİN DERİN SİNİR AĞLARI ALGORİTMALARI

Hussam KANAAN

Elektrik Elektronik Mühendisliği Bölümü

Anadolu Üniversitesi, Lisansüstü Eğitim Enstitüsü, Kasım 2019

Danışman: Doç. Prof. Dr. Tansu FİLİK

2010 yılında düzenlenen Tüketici Elektroniği Fuarında ilk ticari drone'nun tanıtılmasından bu yana endüstriyel, ticari ve kamu güvenliği gibi alanlarda bu araçların popülerliği artmaktadır. Bunun yanında Drone'ların kötü amaçlarla kullanıldığı, özelikle havaalanı, nükleer tesisi gibi yüksek güvenlik gerektiren yerlerde büyük zararlara yol açabildiği bilinmektedir. Sonuç olarak bu tür kritik alanlardaki izinsiz uçan drone ihlallerini tespit etmek ve gerekli uyarıları oluşturmak önemli bir gereksinimdir.

Derin öğrenme modellerinin tasarım ve uygulamasındaki son teknolojik gelişmeler, bu modelleri konuşma, müzik, çevresel sesler ve görüntü tanıma gibi çeşitli klasik problemlerin çözümünde tekrar gündeme getirmiştir. Evrişimsel Sinir Ağları (CNN'ler) ve Tekrarlayan Sinir Ağları (RNN'ler) yaygın olarak kullanılan modelleridir. RNN'lerin doğal dil işleme uygulamaları gibi çeşitli sıra verileriyle ilgili görevlerde dikkate değer bir performans gösterdiği bilinmektedir. Öte yandan, evrişimsel modeller, bilgisayarlı görme görevleri için görüntü sınıflandırma ve nesne tanımada büyük başarı elde etmektedir.

Bu tezin ana amacı akustik drone tespitinde güçlü ve verimli bir derin öğrenme modeli önermektir. Bu bağlamda drone seslerinin tanınması için toplanan veri setleri üzerinde hem RNN hem de CNN modelleri oluşturulmuştur. Bunun yanında akustik özelliklerin çıkarılmasında MFCC ve Mel-ölçekli filtre bankaları kullanılmıştır. Yapılan çeşitli denemelerle, Mel-ölçekli filtre bankası kullanan CNN modeli ile 32 KHz örnekleme oranında en iyi sınıflandırma performansı elde edilmiş ve gösterilmiştir.

Anahtar Kelimeler: Derin öğrenme, Evrişimsel Sinir Ağları (CNN'ler), Tekrarlayan Sinir Ağları (RNN'ler), Akustik özellikleri, MFCC, Mel-ölçekli filtre bankaları.

12/11/2019

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 Anadolu 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.

Hussam KANAAN

.....

ACKNOWLEDGMENT

This work has not been done single handedly without the help of my advisor Assoc. Prof. Dr. Tansu FİLİK who guided me from the beginning to end. Although it took me so much time to finish, he understood my situation as a foreign student and kept motivating me. I also would like to thank Prof. Dr. Ömer Nezih GEREK and Dr. Öğr. Üyesi Cahit PERKGÖZ for serving in my committee. Without the support of my family especially may father, my mother, and most importantly my brother Dr. Ammar Kanaan who was the reason that I applied for this degree. In addition, I would like to thank YTB (Yurtdışı Türkler ve Akraba Topluluklar Başkanlığı) for their financial support with scholarship during my master education. Finally, I would like to thank my academic and life-time friends Eyad, Jihad, Youssef, Yassin, Rasha, Özge Çakır, and special thanks to my precious friends Adnan, Ghaith, and Dana A. Hawwa.

TABLE OF CONTENTS

Page

LIST OF FIGURES

Page

LIST OF TABLES

Page

1. INTRODUCTION

1.1. Background

At Consumer Electronics Show (CES) 2010, the first commercial drone was presented, and during the past nine years several businesses and even individuals use drones for various purposes motivated by the evolution of drones' technologies [1]. The unmanned air vehicles (drones), supplied with several types of sensors, are becoming increasingly widespread for several industrial, commercial, and public safety applications. Today we can say that we are living in drones' era where drones are adopted by different sectors for various tasks such as, search and rescue, filming, shipment of goods, and transportation soon. However, drones with uncontrolled deployment can be used for malicious purposes and they pose a critical challenge in terms of privacy and security for highly security-sensitive areas such as nuclear plants, airports, presidential houses, and other sensitive areas [2].

Recently, the world has witnessed several incidents caused by drones. For example, in 2016 an electricity blackout, which took 6 hours to fix, was caused by a drone collided with power lines in Sichuan in China. Another incident near to Cape Town in South Africa was reported when a drone crashed a nuclear facility. In 2016 also, unauthorized drone activity closed Dubai international airport three times so many flights were diverted [3]. The Venezuelan president has been assassinated by two drones while he was speaking at the 81st anniversary of the national army [4]. These frequently reported incidents raised attention to malicious drones' activities opening a research and development area for both academia and industry to develop methods for drones detecting and disabling. By 2024 the market of drones' detection systems is expected to reach $ 1.85 [1].

1.2. Problem Statement

A decade ago, drones were restricted technology specified only for official authorities. However in recent years since introducing the first commercial drone by Parrot at CES 2010, several public and private sectors started utilizing drones due to their diverse uses and affordable prices [5].

By considering privacy and security, drones constitute game-changing technologies. "Terrorism by Joystick" is an example of topics that have been reported

by the media describing the malicious effects of drones today [3]. The use of drones is no longer restricted by industrial and private sectors, malicious entities started using drones for their purposes so that the number of drone-related incidents has been increased and these incidents are reported daily. The frequently reported incidents have opened a new area of research and development for both industry and academia in order to develop anti-drone systems that are able to detect and disable unauthorized malicious drone [1].

Drones' detection is a critical challenge due to their small size and versatility, so their detection is more difficult than normal Unmanned Aerial Vehicle (UAV) detection. The detection methods can be classified into three major groups: (1) electromagnetic wave-based detection, (2) sound waves detection for the drones which do not emit any radio waves, and (3) drones detection using cameras [2].

Deep Neural Networks are discriminative classifiers that can model the highly non-linear relationships between the inputs and outputs, and which can be easily adopted to output multiple classes at a time (multi-label classification) [6]. With the remarkable achievements of deep learning methods in image classification tasks and natural language processing NLP, DNNs started attracting the attention of anti-drone researchers. Recent advanced DNNs models such as deep convolutional neural networks CNN and recurrent neural networks RNN can be used to detect the malicious drone attacks [7].

1.3.General Overview of the Thesis

In this thesis, a deep Convolutional Neural Network (CNN) is proposed for detecting the presence of a drone based on recorded environmental sound waves. In order to prove the efficiency of the proposed CNN model, we are going to test and compare it with another deep learning model that is especially designed for sequence data manipulating. The second model is a deep Recurrent Neural Network (RNN). The two models will be trained, tested and evaluated on the same dataset which is a combination of three different audio sets. The first set is recorded and collected in the lab using TASCAM 1800 US audio interface [8], the second set is UrbanSound8K which is an open source audio data for scientific research that consists of 8000 audio clips categorized in ten different classes [9], and the third dataset is DREGON [10] which includes a set of drone's records in different scenarios and different noise levels.

The combined dataset will be divided into two parts train and test sets. The test set is about 10% of the combined dataset. The performance of the two models will be evaluated on the test set using some evaluation metrics such as accuracy, precision, f1 score which is a harmonic mean of precision and recall, and confusion matrices. Python and some deep learning frameworks, such as Tensorflow and Keras, will be used for building and developing the models.

1.4. Thesis Outline

The outline of the thesis is given as follow:

In chapter 2, the background information for machine learning and deep learning are given, including convolutional neural networks CNNs and recurrent neural networks RNNs.

In chapter 3, drones background, malicious drone uses, and malicious drone detection in restricted flight areas are presented.

In chapter 4, the proposed methodology is presented, and the implementation results are introduced.

In chapter 5, the conclusion of the thesis is given.

2. DRONES BACKGROUND, MALICIOUS DRONE USES, AND DRONE DETECTION IN RESTRICTED FLIGHT AREAS

2.1. Introduction

In this chapter, the relevant background required for the rest of the thesis is provided. This background has an importance to understand drone functionalities, malicious drone activities, and drones' detection in restricted flight areas.

2.2. Drones Background

Drones are remotely controlled multirotor small aircrafts. They are grouped according to the number of rotors they have into tricoptor, quadcoptor, and hexacopter. Drones technologies and capabilities are highly important in order to understand challenges and issues related to them, and to develop anti-drone systems for counteracting these small flying threats.

Several organizations (NASA, NATO, and State Regularity Authority) have classified drones into various groups or classes. These classifications differ according to the classification criterions that have been used by each organization, but the most common classification is based on the drone's weight. According to this classification drones are grouped into four groups: Nano (less than 0.2 Kg), Micro (0.2-2 Kg), Mini (2-20 Kg), small (less than 150Kg), and Tactical (more than 150 Kg) [1].

Most drones are provided with video cameras which provide the operator with a video stream transmitted over a radio channel called First Person View channel (FPV-channel) Figure 2.1.

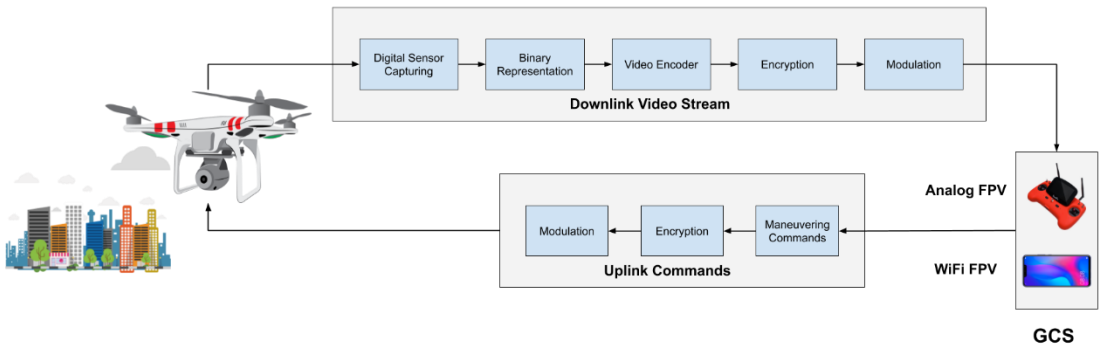

Figure 2.1. *FPV-channel uplink and downlink*

A typical FPV-channel include: (1) an uplink channel for drone's controlling and commanding, (2) a downlink channel for video streaming from drone's camera to

the operator. Digitalization, compression, encryption, and modulation are common stages in both uplink and downlink channels.

The majority of commercial drones' manufacturers (such as Parrot, Yuneec, and DJI) provide their sold drones with the following sensors [1]:

- GPS sensor- the drone uses this sensor for automatic navigation and localization.
- 4K/Full HD Camera- which is used to capture images and videos.
- Motion Sensors- such as gyroscopes, accelerometers, and magnetometers used for drone stabilization.
- Barometer- for computing the drone's flying altitude.
- Collision Avoidance System- vision systems based on ultrasonic sensors and monocular cameras are provided to recent drones to sense dangers in their vicinity.

2.3. Malicious Drone Uses

The FPV-channel capabilities allow modern drones' operators to fly drones far away from their locations. The capabilities that have convinced several sectors and entities to adopt drones for various legitimate purposes, have also encouraged malicious entities to misuse drones for malicious and illegitimate purposes such as, smuggling, physical attacks, spying and tracking, and so on.

2.3.1. Smuggling

The size, speed, flight range, and carrying capabilities of the current generation of drones have encouraged criminals to use drones for smuggling goods and drugs between countries and borders, and smuggling weapons and other contraband into prisons [11]. Smugglers were encouraged to use drones for their crimes because: (1) the difficulty in identifying the identity of the drone's operator even if the drone has been detected, (2) drones neglect the necessity to a human smuggler.

2.3.2. Physical attacks

The title "Terrorism by Joystick" can be given to the era we are living in according to several sources [3], [1], so shooting and exploding drones are no longer related to science fiction. Terrorists are led to adopt drones for various purposes by the same reasons that led criminals to utilize drones for smuggling. Recently, the

Venezuelan president has been assassinated by two drones while he was speaking at the 81st anniversary of the national army [4].

2.3.3. Spying and tracking

A malicious drone operator can use FPV channel capabilities for spying on people and tracking them without being detected due to the following reasons:

- The FPV channel supports HD resolution that enables the operator (spy) to capture high resolution pictures even if the target position is far from the drone.
- Using encryption, the FPV channel can be encrypted and secured.
- The FPV channel allows a malicious drone's operators to spy even if they are not close to their targets.

In addition to the previous reasons, drones can be bought by everyone and operated in populated areas. Drones can be also used by robbers to target empty houses [12].

2.3.4. Launching a cyber-attack

Today, drones can be used to launch a cyber-attack which was infeasible in the past for many reasons such as the distance and the line of sight [1]. A cyber-attack can be done by establishing a cryptic channel in order to infiltrate [13] and exfiltrate [14] data transmitted/received by an organization using a transmitter and a receiver carried by the drone.

2.4. Drones Detection in Restricted Flight Areas

Various methods have been presented for drones' detection in the last few years. Drones are high-speed flying objects, so their detection is much harder than manned aircrafts. In order to deal with drones' detection issue, we need to present dedicated methods, some of these methods are listed below:

2.4.1. RF scanner and spectrum analyzer

Drones have radio signatures that can be recognized using RF scanners and spectrum analyzers. RF scanners and spectrum analyzers can: (1) detect unauthorized malicious FPV-channel, (2) localize the source of this channel in space. In order to detect an approaching spying Wi-Fi drone, a study processed the Received Signal Strength Indication (RSSI) of Wi-Fi signals received by a simple Wi-Fi receiver, but

this method is only applicable if we have a line of sight between the drone and the receiver [15]. Although RF scanners can detect and localize a drone by matching its FPV-channel bands with stored known bands, an attacker can avoid those using dedicated bands that are not popular for FPV-channels.

2.4.2. Acoustic

Various studies have presented the use of a microphone array to detect drones by analyzing the noise emitted by rotors. The acoustic-based detection techniques compare the acoustic signature of a flying drone with other stored signatures, so acoustic-based methods don not need a line of sight condition which is required in RF-scanners based methods. One study [16] deployed machine learning techniques to recognize a drone's FFT signal captured by a microphone, while another study [17] utilized correlation for recognition. The major issue for acoustic detection is collecting the acoustic data need for comparison and recognition, because several factors and conditions like obstacles, wind, temperature, time of the day, and other environmental sounds can affect the sound waves and change their directions [18]. The collection of audio signals in a forest on a cold and windy night will be significantly different than the collection of audio signals in a plain with a little wind and on a hot day [18]. Although acoustic-based detection techniques can be deployed to detect the presence of a drone than localize it using multiple distributed microphones, acoustic-based drones' detection techniques suffer from false negative detection due to the increased models of drones. In addition, acoustic-based detection methods are restricted by the distance between drones and microphones [19].

2.4.3. Optical

Using cameras that have the ability to detect visible frequencies, many studies presented methods to detect a drone and its path from video streams by detecting visual masks [20], shape description [21], and motion cues. Other methods suggested using deep neural networks [22], [23]. Although the abovementioned methods can be deployed to detect and localize drones using databases, they suffer from false negative detection because of the increasing number of drones' models and ambient darkness. They also suffer from false positive detection because of the similarities between the movements of birds and drones. Several studies suggested using a thermal camera which can capture invisible wavelength to address the comprised drone detection rate in

dark conditions. A recent study suggested using short-wave infrared (SWIR) for drone detection in dark conditions [24].

2.4.4. Hybrid

A method or a sensor that can meet all requirements of a perfect drone detection system is not existed. All the above-mentioned methods have their advantages and disadvantages, so to overcome the limitations of using one sensor we need to suggest several sensor fusion methods. Several methods [25], [26] suggested using a camera with a microphone array, or an acoustic camera as a combination of acoustic and optical methods. Another study proposed a combination of optical and RF methods such as LiDAR and radar [27]. Other studies suggested combining the three methods (optical, RF, and acoustic) to detect and locate drones in space [28] [29]. In general, using more than one sensor is the technique used by several companies for drones' detection and localization. However, the greatest disadvantage associated with this technique is the high cost.

3. BACKGROUND AND DEFINITIONS

3.1. Introduction

In this chapter, a set of important definitions is presented. These definitions have tremendous importance to understand Machine Learning (ML), Artificial Intelligence (AI), and Deep Learning (DL), and to distinguish between these different concepts and terms.

3.2. Machine Learning

Programming computers so that they can learn from data is the science of machine learning. ML is not a futuristic science as many people think, it has been for decades. Optical Character Recognition (OCR) and filter spam are two examples of ML applications that back to the 1990s [6]. A more general definitions can be presented as follow:

"Machine learning is the field of study that gives computers the ability to learn without being specifically programmed". –Arthur Samuel, 1959 [6].

And a more engineering-oriented one:

"A computer program is said to learn from experience E with respect to some task T and some performance measure P, if its performance on T, as measured by P, improves with experience E". –Tom Mitchell, 1997 [6].

Artificial intelligence, machine learning, and deep learning are frequently mentioned terms when discussing advanced technology. AI can be understood as any technique that enables computers mimic human intelligence, a perfect example of AI is self-driving cars and natural language processing (NLP) applications such as machine translation. Machine learning (ML) can be understood as a subset of AI that includes algorithms and statistical models used to perform a specific task without explicit instructions. Deep learning can be understood as a subset of machine learning that uses multilayered neural networks to learn from vast amount of data. Figure 3.1 illustrates the main subsets of AI. Machine learning systems can be classified according to the type and amount of supervision during training into four major categories: supervised learning, unsupervised learning, semisupervised learning, and reinforcement learning.

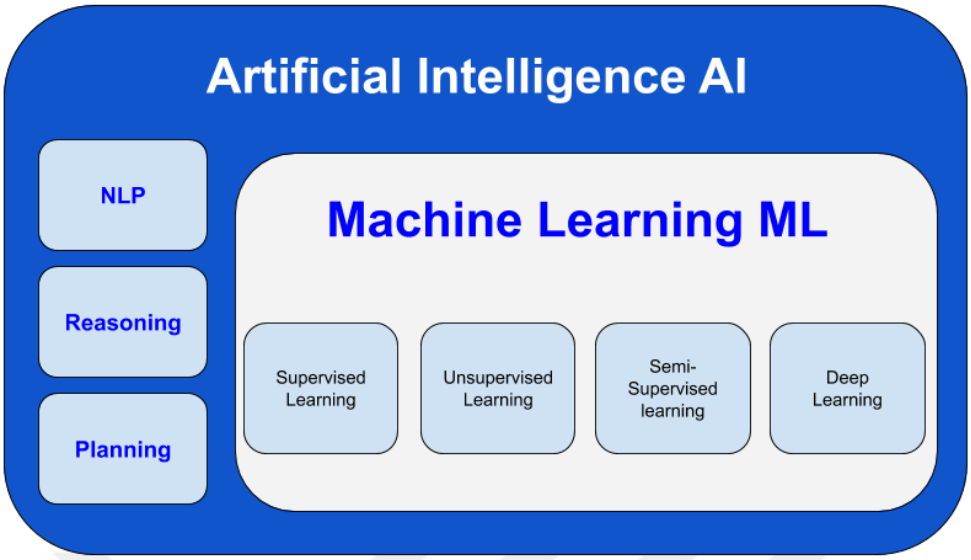

Figure 3.1. *AI is the main set that includes ML and DL*

3.2.1. Supervised learning

In supervised learning, the training data is labeled, in other words the training data we fit to the model includes the desired solutions, called labels. Figure 3.2 illustrates an example of supervised learning system (spam filter).

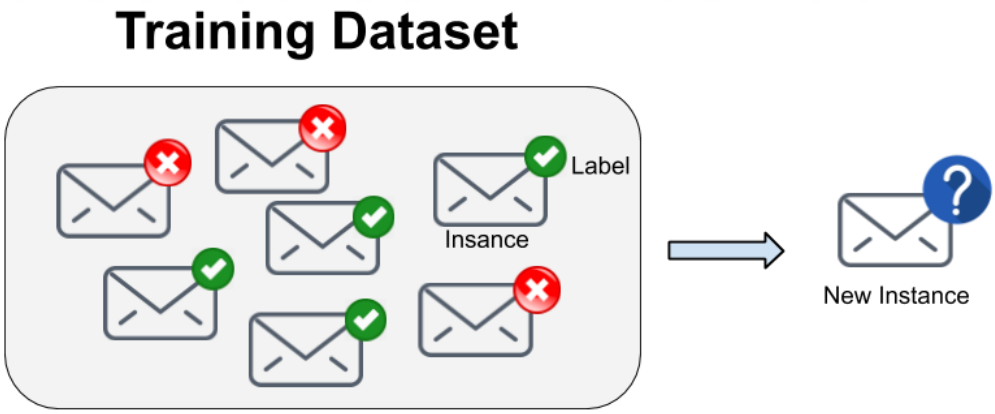

Figure 3.2. *Spam filter an example of supervised learning*

Classification and regression are two typical tasks of supervised learning. In classification tasks the output is a label (class), we have binary classification (the output is only two labels such as “drone” and “no drone” in our project) and multi-label classification (the output is more than two labels such as environmental sounds classification). Regression task is to predict a target numeric value corresponding to the

input features Figure 3.3. Predicting the price of a house given a set of features (location, number of rooms, etc.) called predictors is an example of regression tasks.

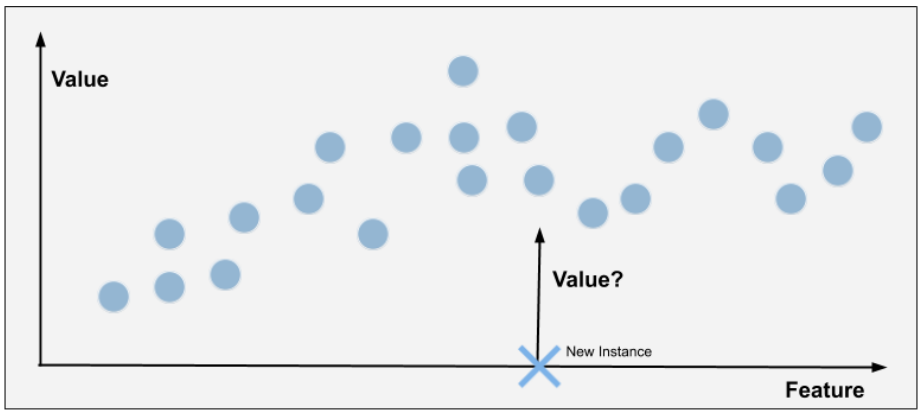

Figure 3.3. *Regression*

Some regression algorithms can be used for classification and vice versa. Logistic regression is an example of regression algorithm that can be used for classification. In the list below a set of the most important supervised learning algorithms:

- Linear Regression
- Logistic Regression
- Support Vector Machines (SVMs)
- K-Nearest Neighbors
- Neural Networks
- Decision Trees and Random Forests

3.2.2. Unsupervised learning

In unsupervised learning we have unlabeled training data, so the algorithm tries to answer on unlabeled or unknown data. Unsupervised learning is commonly used by data scientists for discovering patterns in new datasets. In the list below a set of the most important unsupervised learning algorithms:

I. Visualization and Dimensionality Reduction
   - Principal Component Analysis (PCA)
   - Locally-Linear Embedding (LLE)
   - T-distributed Stochastic Neighbor Embedding (t-SNE)
   - Kernel PCA

II. Clustering

- Hierarchical Cluster Analysis (HCA)
- Expectation Maximization
- K-Means

III. Association Rule Learning

- A priori
- Eclat

Visualization algorithms, dimensionality reduction, and anomaly detection are good examples of unsupervised learning. In visualization algorithms, we feed a lot of complex and unlabeled data, and the output is 2D or 3D representation of this data that can be easily plotted. In anomaly detection, the system is trained using normal training examples (normal instances), after that it can classify a new instance and tell whether it looks a normal one or whether it is likely an anomaly as illustrated in Figure 3.4.

Figure 3.4. *Anomaly detection*

3.2.3. Semisupervised learning

In this type of learning, algorithms can deal with partially labeled data. . In fact we have a lot of unlabeled data and a little bit of labeled data. In order to build a semisupervised learning algorithm we need to combine both supervised and unsupervised learning algorithms. Deep Belief Networks (DBNs) are considered a good example of semisupervised learning algorithms. These networks are built using stacked unsupervised components called Restricted Boltzman Machines (RBMs) where the whole system is tuned using supervised learning, while the RBMs are trained using

unsupervised learning [6]. Figure 3.5 illustrates the mechanism of semisupervised learning.

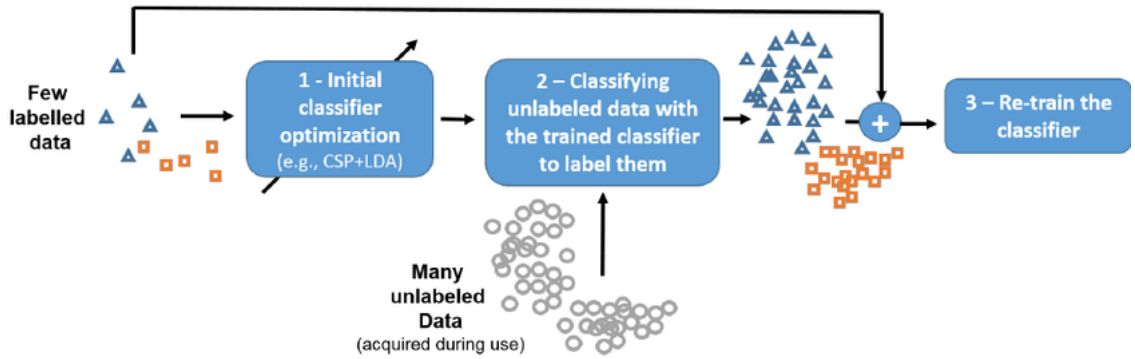

Figure 3.5. *Semisupervised learning*

3.2.4. Reinforcement learning

Reinforcement learning is a behavioral learning model. The algorithm receives feedback from the data so the user is guided to the best outcome. In reinforcement learning the system (model) learns through trial and error, so it is not trained with the training data set like other types of supervised learning. The learning system, called an agent in this context, can observe the environment, select and perform actions, and get rewards in return (or penalties in the form of negative rewards). Then, the model must learn the best strategy, called a policy, to get the most reward overtime. A policy represents the actions that are taken by the agent when it faces specific situations [6]. Figure 3.6 shows the reinforcement learning.

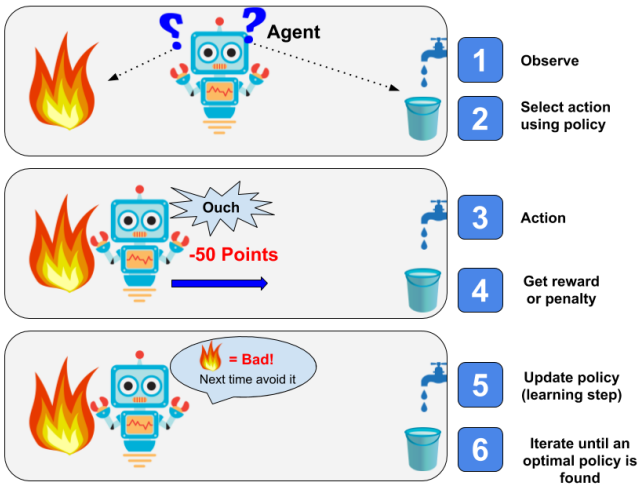

Figure 3.6. *Reinforcement learning*

3.3. Artificial Neural Networks

Among popular machine learning techniques, artificial neural networks come first. Artificial neural networks (ANNs) simulate the mechanism of learning in the human nervous system that contains cells, which are commonly known as neurons. These neurons are connected with each other using axons terminals and dendrites, and the connecting regions between dendrites and axons are referred to as synapses [6], these parts are illustrated in Figure 3.7. The major components of neurons with their tasks are listed below:

- Dendrites- receive input signal from other connected neurons in the form of electrical pulse.
- Cell body- decides what action to take base on generated interferences from inputs.
- Axon terminals- transmit output signals as electrical impulses.
- Synapses- connect adjacent neurons.

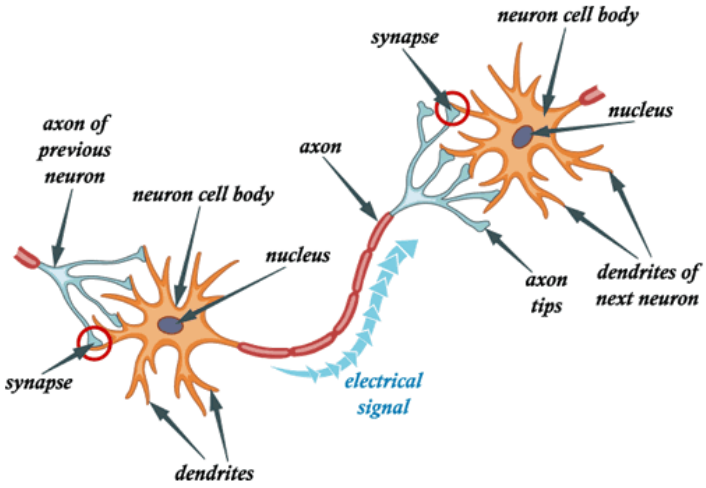

Figure 3.7. *Structure of human neurons*

Artificial neural networks (ANNs) simulate this biological mechanism, where they contain computational unites called neurons. Artificial neural networks (ANNs) are mathematical and computational abstractions of biological processes that take place in the brain, and the name artificial comes from their artificial representation of the working mechanism of human being's nervous system Figure 3.8. Neural network is an arrangement of a set of computational units (neurons) in layers where every neuron is a given layer that is connected to every neuron in the next layer.

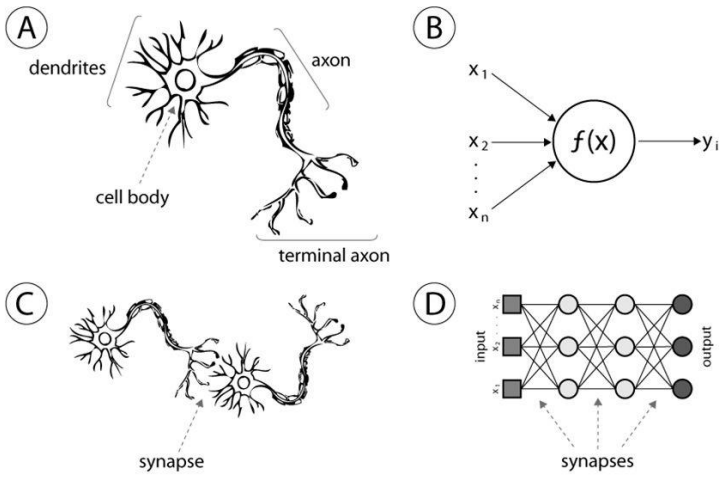

Figure 3.8. *Similarities between ANNs and human nervous system*

A neuron in a given layer takes the data, that is passed into it, and multiplies it by matrix of numbers called weights and then adds a number called bias to produce a single number as an output. The weights and biases of each neuron are adjusted incrementally to decrease the loss which is the average amount the network is wrong by across the training data. Current neural networks contain multiple neurons organized in multiple layers. Figure 3.9 is an example of simple neural network; this network contains input layer, output layer, and two hidden layers. Hidden layer means that it is neither input nor output. These types of layers with all neurons connected to each other are called a fully connected or dense layer.

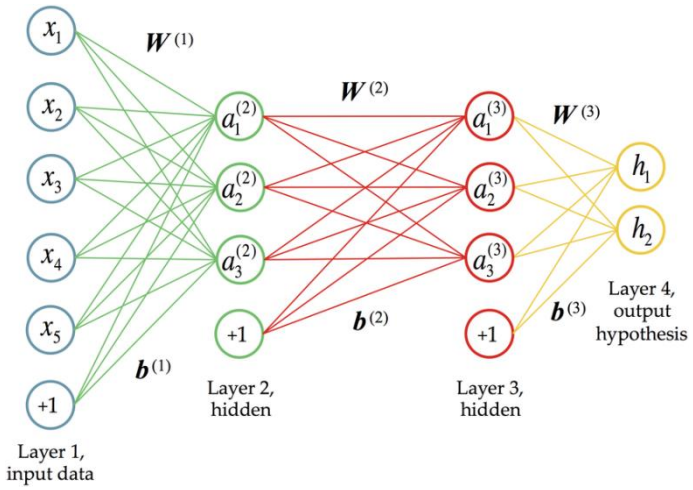

Figure 3.9. *An example of simple neural network with two hidden layers*

Neural networks are the core of Deep Learning (DL). They are scalable, versatile, and powerful, making them a perfect choice to manipulate highly complex machine learning tasks, such as powering natural language processing services (e.g. Amazon's Alexa and Apple's Siri), classifying millions of images (e.g. Google images), and recommending the best web pages, online products, songs, and videos to millions of internet users (e.g. YouTube, Facebook, and Amazon).

Artificial neural networks (ANNs) were first introduced by the mathematician Walter Pitts, and the neurophysiologist Warren McCulloch in 1943 in their paper, "A logical Calculus of Ideas Immanent in Nervous Activity" [6]. There are many reasons behind the growing interest in neural networks and deep learning nowadays. These reasons are listed below:

- The availability of huge quantity of data to train neural networks, and with huge data neural networks outperform machine learning techniques specifically in complex and large problems.
- The tremendous advances in computation power since 1995 that make provide the possibility to train large neural networks in a reasonable time. Thanks to Moor's law and to the growing industry, that has produced powerful GPU cards such as Nvidia.
- The improvement in training and optimizing algorithms such as stochastic gradient descent SGD and adaptive momentum estimation Adam.
- The powerful software frameworks provided by the giant technical manufacturers such as Tensorflow by Google, Keras, Sklearn, OpenAI, etc.

3.4. Deep Learning

Deep learning has emerged as a new area of machine learning research since 2006, many loosely related definitions or high-level descriptions of deep learning are existed, we provide the most general two definitions.

Deep Learning can be defined as a class of machine learning techniques that exploit many layers of non-linear information processing for supervised and

unsupervised feature extraction and transformation, and for pattern analysis and classification [7].

Deep Learning can be also defined as a set of algorithms in machine learning that attempt to learn in multiple levels, corresponding to different levels of abstraction. It typically uses artificial neural networks (ANNs) [7].

The work in deep learning can be broadly classified into three major classes, deep neural networks for unsupervised or generative learning, deep neural networks for supervised learning, and hybrid deep neural networks. These three categories are explained below.

Deep neural networks for unsupervised or generative learning: these techniques are used when there is no information about ground truth labels, and they are intended to detect high-order correlation of the observed data for analyzing patterns or synthesis purposes.

Deep neural networks for supervised learning: they are also called discriminative deep networks where target label data is always available in direct or indirect form, this type of neural networks are intended for pattern classification purposes by characterizing the posterior distributions of classes conditioned on the visible data.

Hybrid deep neural networks: the deep architecture of this type of neural networks either comprises or makes use of both discriminative and generative model components for this reason they are called hybrid. In the literature of hybrid architecture where the discrimination is the final goal, the generative component is mostly used to help with discrimination [7].

3.5. Deep Learning Applications

Tremendous traditional and extended signal processing areas have exploited deep learning techniques. The applications of deep learning are categorized into many classes: speech and audio, images and multidimensionality, language modeling, information retrieval.

4. PROPOSED METHODOLOGY

4.1. Introduction

Investigating how well drones' sounds can be classified using deep convolutional neural networks CNNs that are specially designed for image classification (object recognition) is the main purpose of this thesis. In order to prove the superior efficiency of CNN (the proposed method), two learning algorithms for classification are going to be experimented and compared recurrent neural networks RNNs, and convolutional neural networks CNNs. While the first algorithm the first algorithm is appropriate for sequential data and it has proved its efficiency in NLP applications, the second algorithm is widely used in computer vision applications such as image classification and object recognition inside images. Recently, using convolutional neural networks for audio-related tasks is becoming increasingly widespread.

In this chapter, we evaluate the proposed CNN acoustic-based drones' sounds classification model by gathering real data. The acquired audio data is preprocessed and classified by implementing feature extraction then feature classification using Python and deep learning frameworks.

4.2. Related Works

The acoustic scene classification (ASC) refers to the capability of a human or an artificial system to recognize an audio context from a recording or from a live stream. In recent years, the problem of ASC has received increasing interest from research community. The ASC has many applications like mobile robot navigation [30], context-aware computation [31] and intelligent wearable interfaces [32]. The applications of CNNs exceeded computer vision area to audio-related tasks, and they are becoming more and more widespread, for example speech recognition [33], and environmental sounds classification [34], [35].

Drones acoustic signature recognition or drones' sound classification falls under ASC. Most commercial drones have a typical acoustic signature that can be used to detect the presence of a malicious drone in flight-restricted areas. Many studies presented methods to detect drones based on their acoustic signature.

Recent study presented a real-time drone detection and monitoring system which uses two machine learning algorithms: Plotted Image Machine Learning (PLL), and K-

Nearest Neighbor (KNN) in order to classify the FFT of real-time sampled data by comparing it with a reference FFT template associated with a target of interest [16]. J. Mezei and A. Molnar presented sound detection scheme based on correlation for limited database, but in real-time environment the accuracy was quite low [17]. Linear Predictive Coding (LPC) can be used in sound-based drones' detection systems, since it is used in speech recognition applications. LPC detects spikes in audio signal's frequency spectrum. The principle of LPS is to estimate the signal at a point of time from past samples, LPC coefficients refer to the estimated coefficients. These coefficients can be trained then stored in a database. An LPC-based detection system can be falsified with similar sounds like drones. In order to solve this problem, L. Hauzenberger and E. Holmberg Ohlsson decreased the number of false alarms by considering the slope of frequency spectrum, and they proved that it is beneficial in terms of false alarm detection [36].

Although many studies suggested methods for drones' sound recognition and classification, previous studies were trained and tested on datasets (audio signals) collected and recorded in an ideal environment (in terms of noise) rather than real-life environment. On top of that, there isn't a good dataset that contains a good number of training examples (drones' audio signals) recorded in real environment. Our goal is to use deep learning algorithms in order to build a classifier (model) which is able to recognize a drone's acoustic signature in real environment, and this classifier can be later utilized in a sound-based real-time drones' detection system. Motivated by our goal, our contributions are summarized as follow:

- Collecting a dataset from real environment in order to train and test the effectiveness of the proposed model in terms of accuracy, precision, recall, f1 score, Receiver Operating Characteristics (ROC) curve, Area under the ROC Curve (AUC), and Precision-Recall (PR) curve.
- Showing that the integration of ML framework and the advanced acoustic processing techniques can effectively and timely detect drones based on their acoustic signatures.
- Comparing the performance of two deep learning algorithms (classifiers) applied to our classification task: Recurrent Neural Networks (RNN), and Convolutional Neural Network (CNN), in order to prove that the

proposed CNN model yields state-of-the-art performance for drone sound classification.

- Performing hyper-parameters tuning by changing the values of the model parameter and see their influences on the classification performance. The parameters which are tuned: acoustic features (Mel Frequency Cepstral Coefficients (MFCC), or Log-Filter Banks), sampling rate (16 KHz, 32 KHz, or 44.1 KHz), frame length (100 ms, 200 ms, or 300 ms). The classification performance is evaluated using the abovementioned metrics accuracy, precision, etc.
- Introducing a deep learning classification algorithm that is applicable for all sound classification tasks such as accent classification, music genre classification, urban sound classification, and etc.

The rest of this chapter is organized as follow: section (2) describes in details the proposed drone's acoustic signature classification methodology. Section (3) discusses implementation tools needed to achieve our classification task. Finally, experimental results are discussed in Section (4).

4.3. Proposed Methodology

In this section we provide our methods and architectures chosen for achieving our target task. Drones have salient acoustic signature that can be used to differentiate them from other sounds in the environment. Sound features play a key role in achieving high accuracy and efficiency in sound classification and recognition. Hence, in order to get accurate classification results, it is really important to select meaningful features from sound samples. Raw audio signals contain noise and silent regions. Hence, it is not appropriate to pass these signals directly to a sound classifier, and it will lead to inaccurate results. By extracting acoustic features, we transform the raw audio signal into feature vectors that represent the signal in non-redundant and compact way, and have the prominent frequency components.

The adopted sound-based drone detection scheme is illustrated in Figure 4.1. We used an array of microphones (exactly 5) to gather audio records of drones in different scenarios. Acoustic features are extracted from these sounds using audio analysis techniques such as MFCC and Mel-Scaled Filter Banks. After getting feature vectors and feature images, these vectors are passed to a trained deep learning model (classifier)

in order to classify recorded audio signal if it belongs to drone or not. The workflow of the proposed scheme is shown in Figure 4.2.

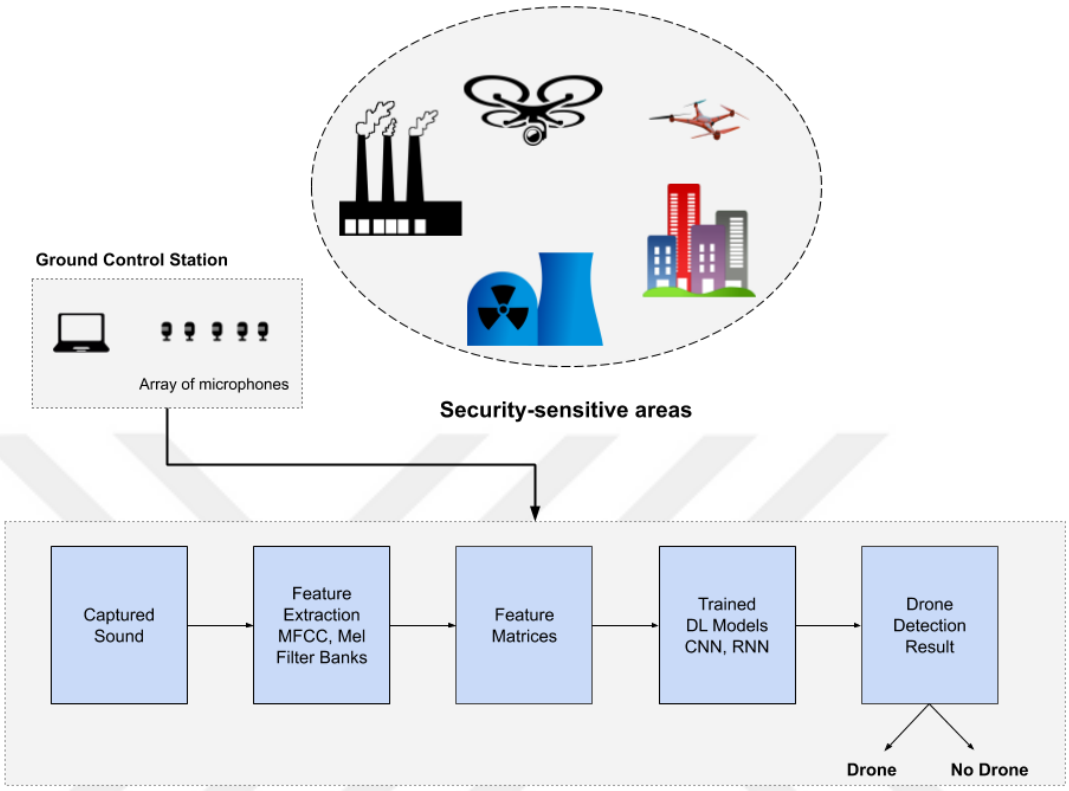

Figure 4.1. *Drone detection system model*

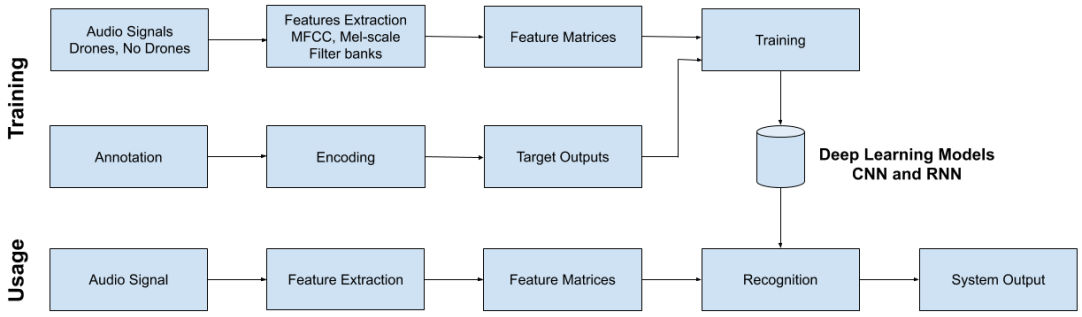

Figure 4.2. *Work flow of drone detection system*

4.3.1. Data acquisition

Data acquisition is the first stage in the development of our proposed system, and it is a critical stage because it affects the performance of the whole proposed system. Data acquisition is an important stage because machine learning algorithms use

this data to learn parameters of the acoustic models, then assess the performance of the trained models. The data should include the raw audio material (set of audio clips) and the associated reference metadata (e.g. class labels); in general, the reference metadata comes in CSV file format. The type of acoustic data, the type or required metadata, and conditions in which recordings are collected are all dictated by the defined target application.

Essentially, the aim of data acquisition stage is to collect as realistic as possible acoustic signals in conditions which are similar to the intended target application. Metadata is usually manually annotated during the data collection, and it includes ground truth information. All sound classes required for the target application should be represented in a sufficient amount in the dataset to enable the acoustic models to generalize well [37]. For the supervised classification methods the availability of datasets is limited. Motivated by this, we collected our dataset for our target task. This dataset is a combination of three different datasets. The first one is collected at our university using TASCAM US 1800 sound card with an array of five microphones Figure 4.3. We set the sampling frequency to 44.1 KHz and 24 bit is the sample depth. The second dataset is DREGON which stands for DRone EGo noise and localizatiON. It is a publicly available dataset of annotated sounds recorded with 8-channel microphone array embedded into a quadcoptor [10]. The third one is UrbanSound8k [9]. The size of all these datasets is 11.77 GB, and they include 10420 audio files divided into two main sets: training set (9380 audio files 9.79 GB), and test set (1040 audio files 1.98 GB). It is highly recommended to divide the data into train and test sets. The test set includes audio files that are not seen during model training, so it is used for model's performance assessment. The audio files in both train and test sets have different length. The underlying reason behind this is to make the system able to manipulate and process audio files having different length, instead of being designed for a specific length. Figure 4.3 and Figure 4.4 show the distribution (mean) of audio files' length in both train and test sets respectively.

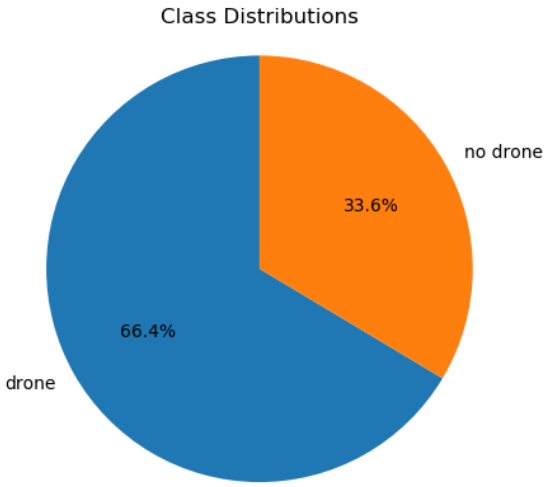

Figure 4.3. *Audio files' length distribution in train data*

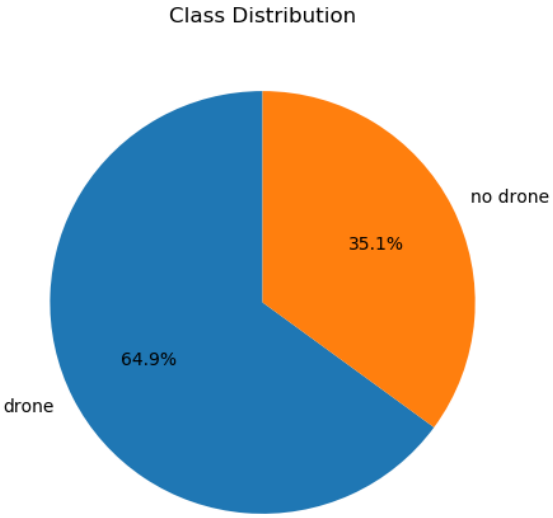

Figure 4.4. *Audio files' length distribution in test data*

4.3.2. Audio processing

In this stage of the overall system, audio files are processed and prepared for machine learning algorithms. This stage includes two phases: preprocessing in which the audio signals are down sampled and processed to emphasize the target sounds, then extract acoustic features from the audio signals in order to represent them in a compact form.

4.3.2.1. Preprocessing

Since we collected the audio data from different sources, the recording settings differ with variations in used sampling rate. Addressing these variations can be done by converting the audio signals into uniform format by resampling them into fixed sampling rate. We use three unified sampling rates (16 KHz, 32 KHz, and 44.1 KHz) to analysis the influence of sampling rate on the models performance. We apply preprocessing to the audio signals before acoustic feature extraction. This step is necessary for enhancing specific characteristics of the incoming signal in order to maximize the performance of audio analysis in upcoming stages. This can be achieved by enhancing the target sounds in the signal. Figure 4.5 and Figure 4.6 show the drone's audio signal before and after emphasizing [38].

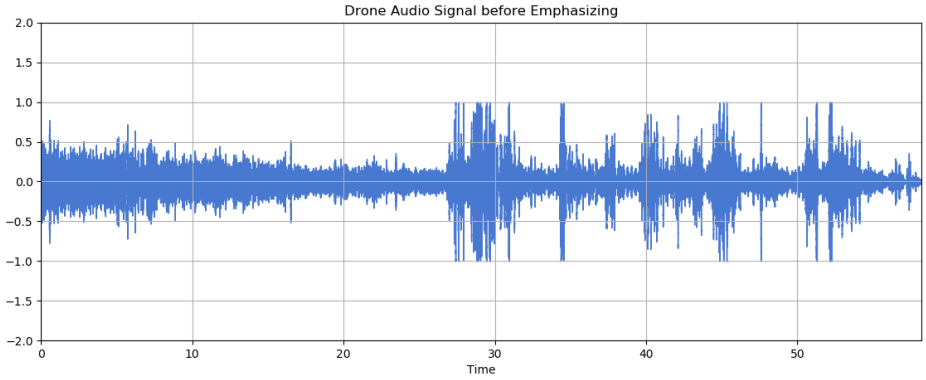

Figure 4.5. *Drone's audio signal before emphasizing*

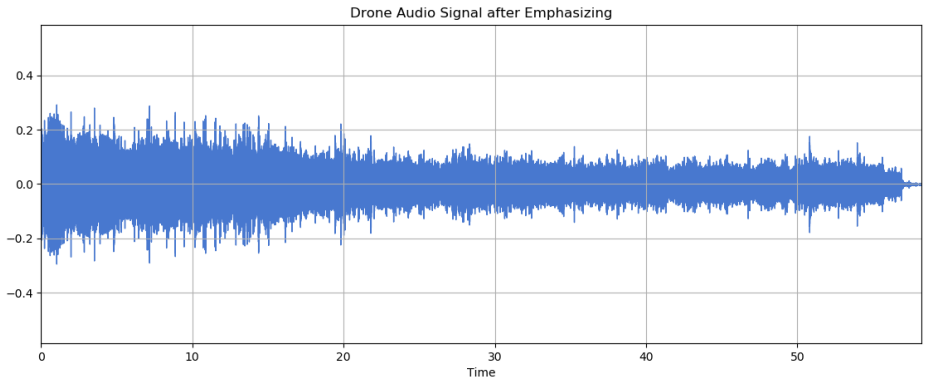

Figure 4.6. *Drone's audio signal after emphasizing*

4.3.2.2. Acoustic feature extraction

Acoustic feature extraction is the critical stage in any sound classification system, because it represents the audio signal in a compact and non-redundant form. In sound recognition systems, the extracted acoustic features should have low variability

allowing distinction among features belong to examples from different classes [39]. The feature representations satisfying the previous property make the learning process easier. In terms of memory efficiency and computation power, using a compact feature representation is better than the direst use of the audio signals in the sound detection systems.

The main purpose of acoustic features extraction stage is to transform the raw audio signal into a numerical representation which reflects its physical characteristics, and which is suitable for machine learning algorithms. We have several types of acoustic features extraction techniques sharing the same processing pipeline. As shown in Figure 4.7 the processing pipeline includes frame blocking, windowing, spectrum calculation, and subsequent analysis [38].

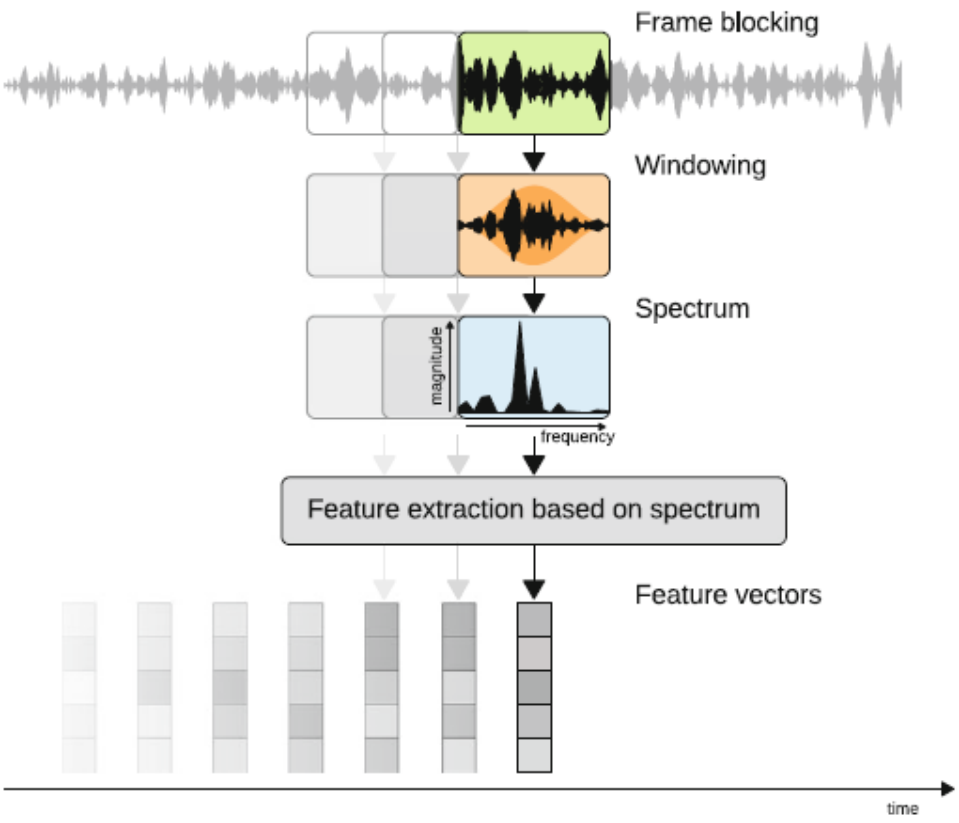

Figure 4.7. *The processing pipeline of feature extraction techniques*

For audio analysis, it usually suggested to use of frequency domain features or time-frequency representations because they include a significant amount of information in relative distribution of energy in frequency [38]. Discrete Fourier Transform (DFT) is the most common transformation used for audio signals, in which

the signal is represented by a superposition of sinusoidal base functions, each base characterized by a magnitude and phase [40]. Discrete Wavelet Transform (DWT) and Constant-Q Transform (CQT) are examples of other transformations for audio signals.

The audio signals are generally classified as non-stationary signals because their statistics such as, mean and magnitudes of frequency components, change quickly over time. Due to this fact, acoustic feature extraction uses the short-time processing approach which means the analysis is performed periodically in short-time segments called analysis frames in order to capture the signal in quasi-stationary state [38]. Acoustic features are classified into time domain features (i.e., energy, zero crossing rate, entropy of energy) and frequency domain features (i.e., spectral centroid and spread, spectral entropy, spectral flux, spectral roll-off, MFCCs, Mel-scaled filter banks).

4.3.2.3. MFCCs and Mel-scale filter banks computation

MFCCs and Mel-scaled filter banks are two widely used acoustic features extraction techniques for representing the audio signals in terms of their spectral content. Their design is inspired by the human auditory system which focuses on low frequency components' magnitudes more that the high frequency components. These magnitudes are non-linearly received (perception) [38]. Using MFCCs and Mel-scaled filter banks, acoustic signals can be represented in the form of visual images. In frequency domain, MFCCs are the most popular used 'acoustic' feature extraction technique due to its high accuracy compared to time-domain features; but more recently, Mel-scaled filter banks are becoming increasingly popular. Computing MFCCs and filter banks include somewhat the same steps, where filter banks are computed in both cases with a few more extra steps in MFCCs computation procedure. Substantially, the processing pipeline in both Mel-scaled filter banks and MFCCs include the following steps:

Pre-Emphasis: A pre-emphasis filter is applied to the audio signal to amplify the high frequencies components. A pre-emphasis filter has several benefits: (1) high frequency components usually have smaller magnitudes compared to lower frequency components, so the role of pre-emphasis filter is to balance the frequency spectrum, (2) avoid numerical complexities during the Fourier transform computation, and (3) a pre-emphasis filter may improve the signal-to-noise ratio (SNR).

$$y(t) = x(t) - \alpha x(t-1) \quad (1)$$

A first order pre-emphasis filter is described in equation (1), where α is the filter coefficient and it takes values 0.95 or 0.97. An audio signal before and after pre-emphasis is illustrated previously in Figure 4.5 and Figure 4.6 respectively. Since mean normalization, that will be discussed in later part, can achieve most of the motivations for the pre-emphasis filter, so in modern systems pre-emphasis filter have a modest effect [41].

Framing: After pre-emphasis step, fixed length analysis frames are generated by slicing the audio signal. These audio frames are shifted by a fixed time step. Audio signal frequencies change over time, so calculating the Fourier transform across the entire signal does not make sense because the frequency contours of the signal will be lost over time. To address this issue, the short-time processing is utilized where it can be assumed that frequencies in a signal are stationary over a very short period of time. Therefore, by calculating a Fourier transform over the concatenated adjacent short-time frames we obtain a good approximation of the frequency contours. The sizes of analysis frames in sound recognition applications typically range from 20 ms to 40 ms with 50% (+/- 10 %) overlap between consecutive frames [41].

Window: Once the signal sliced into frames, a window function such as Hamming window is applied to each frame. Hamming window has the following form:

$$w[n] = 0.54 - 0.46 * \cos(\frac{2\pi n}{N-1}) \quad (2)$$

Where N is the length of the window, $0 \leq n \leq N-1$. Applying Hamming window to the analysis frames has many benefits: (1) reduce spectral leakage, and (2) counteract the assumption made by the FFT that the data is infinite. Figure 4.8 represents the graph of Hamming window.

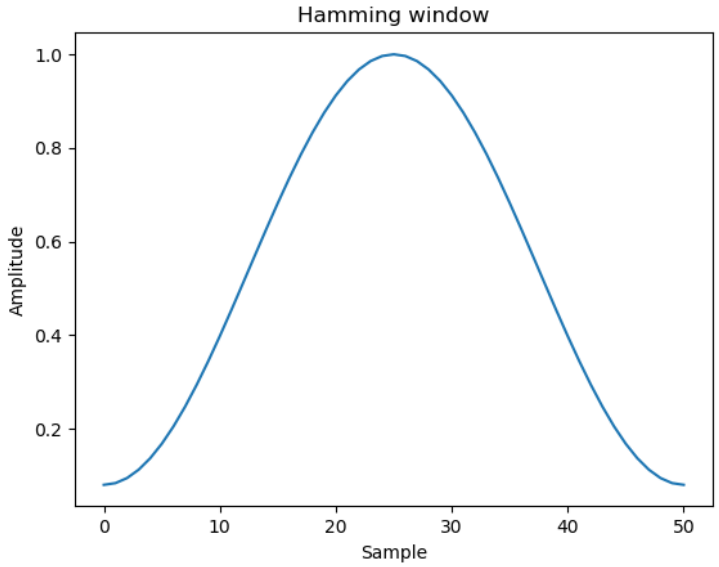

Figure 4.8. *Hamming window*

Fourier transform and power spectrum: N-point FFT can be now applied to each frame in order to calculate the frequency spectrum; in this case it is called Short-Time Fourier Transform (STFT). The power spectrum (periodogram) is computed using the following formula:

$$P = \frac{|FFT(x_i)|^2}{N} \tag{3}$$

Where, $\boldsymbol{x_i}$ is the i^{th} frame of the signal x.

Filter banks: The last step for filter banks calculating is applying triangular filters on a Mel-scale to the power spectrum in order to extract frequency bands. The purpose of applying Mel-scale is to mimic the non-linear human hear perception of sound. Less discriminative at higher frequencies and more discriminative at lower frequencies. The following two equations (4) and (5) convert between Hertz and Mel.

$$m = 2595 \log_{10}\left(1 + \frac{f}{700}\right) \tag{4}$$

$$f = 700(10^{m/2595} - 1) \tag{5}$$

As shown in Figure 4.9 each filter in the filter banks is triangular with unit response at the center frequency and decreased linearly toward 0 till it reaches the center frequencies of the two adjacent filters where the response is 0.

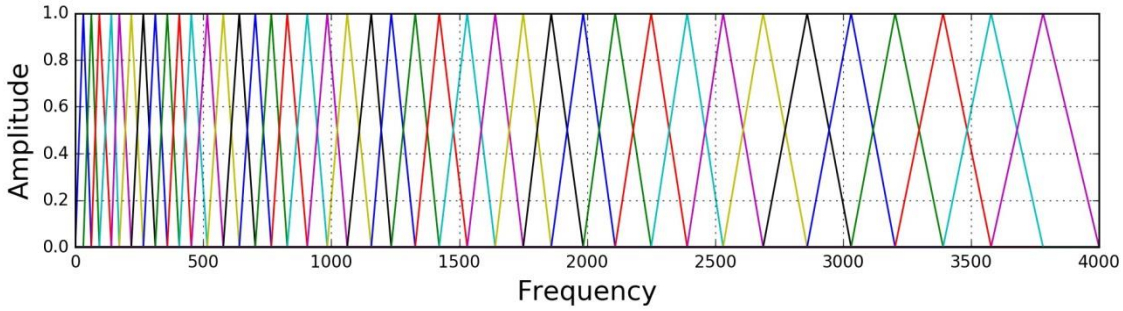

Figure 4.9. *Mel-scale filters*

The following equation (6) models filter banks.

$$H_m(k) = \begin{cases} 0 & k < f(m-1) \\ \dfrac{k - f(m-1)}{f(m) - f(m-1)} & f(m-1) \le k \le f(m) \\ 1 & k = f(m) \\ \dfrac{f(m+1) - k}{f(m+1) - f(m)} & f(m) < k \le f(m+1) \\ 0 & k > f(m+1) \end{cases} \quad (6)$$

Mel-scaled spectrogram can be obtained by applying the above filter bank to the power spectrum. The resulting Mel-scale filter banks (spectrogram of the drone's audio signal) before normalization are shown in Figure 4.10.

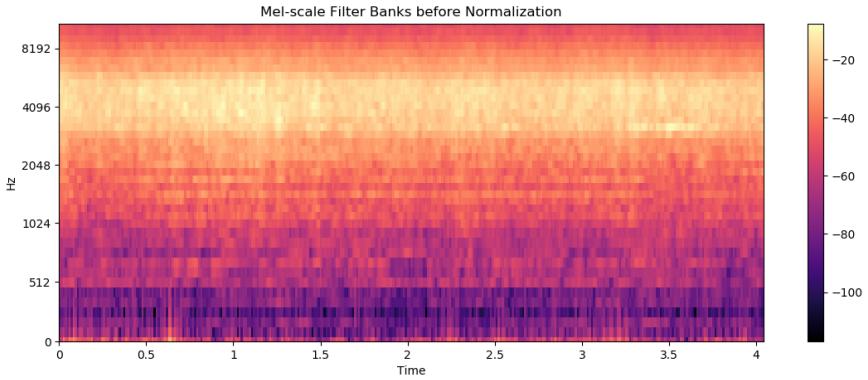

Figure 4.10. *Mel-scale filter banks of drone's audio signal (spectrogram)*

MFCCs: The filter bank coefficients calculated in the previous step are highly correlated, which can lead to some problems in some traditional machine learning algorithms. Therefore, by applying Discrete Cosine Transform (DCT) to the correlated

filter bank coefficients we get a compressed (de-correlated) representation of these coefficients. In most sound recognition systems, the number of retained Cepstral coefficients is between 2 and13 and all other coefficients are discarded. The discarded coefficients represent fast changes in the filter bank coefficients, and these fine details don't contribute to sound recognition performance [41]. The resulting MFCCs before normalization are shown in Figure 4.11.

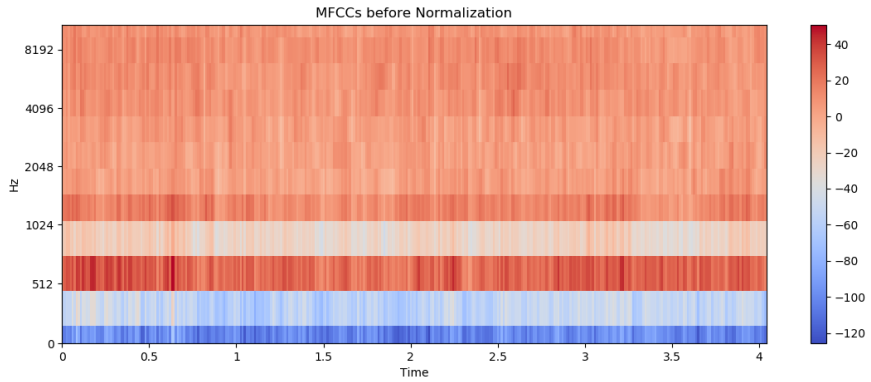

Figure 4.11. *MFCCs of drone's audio signal*

Mean normalization: in order to improve the SNR and balance the spectrum of the audio signal, mean normalization can be applied by subtracting the mean of each coefficient from all frames. The normalized Mel-scale filter banks and MFCCs are shown in Figure 4.12 and Figure 4.13.

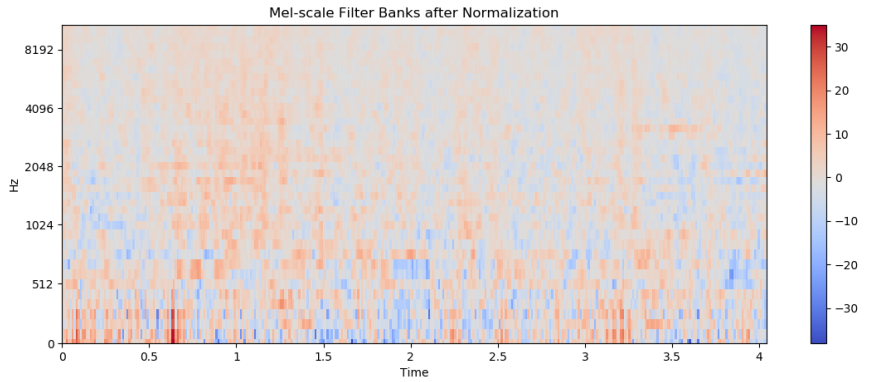

Figure 4.12. *Normalized Mel-scale filter banks*

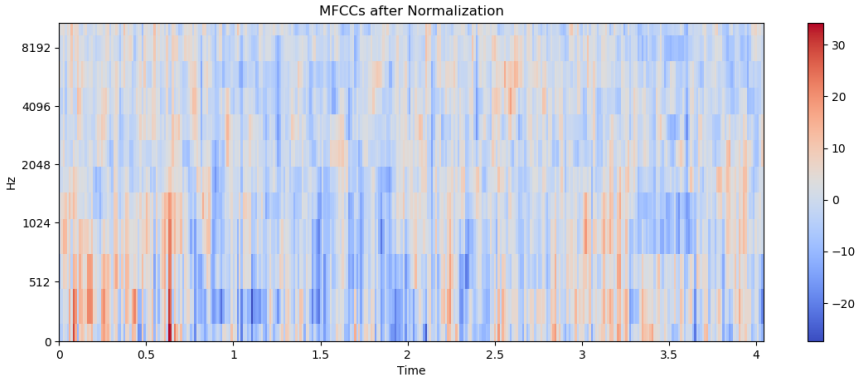

Figure 4.13. *MFCCs of drone's audio signal after normalization*

4.3.2.4. MFCCs and Mel-scale filter banks comparison

To this point, the required steps for calculating the Mel-scale filter banks and MFCCs were discussed in terms of implementation and motivation. It is interesting to note that in order to calculate Mel-scale filter banks we need to implement several steps, these steps are inspired by the human perception of acoustic signals and the nature of these signals. However, the limitation of some machine learning algorithms toward highly correlated data is the underlying reason of implementing some extra steps in order to calculate MFCCs. The DCT was required to de-correlate the filter bank coefficients, this procedure also referred to as whitening. With advent of machine learning algorithms which are less susceptible to highly correlated inputs [42], one might ask question if there is still a need to use MFCCs as a feature extraction technique in deep learning based sound recognition systems; and that what will be investigated by using both techniques (MFCCs and Mel-scale filter banks) with our acoustic models and see which technique yields the best results.

4.3.3. Deep learning models

4.3.3.1. Convolutional neural network model

Using several architectures such as ResNet [43] and VGG nets [44] deep convolutional neural networks (CNNs) have shown outstanding performance in pattern recognition applications [45]. A typical CNN architecture involves three important parts: convolutional layers, pooling layers, and fully-connected layers.

A convolutional layer consists of a set of filters (also called kernels), each filter has a receptive field. Convolutional layers are able to deal with two dimensional data with translation due to the shared filters and local connectivity. This local connectivity

is highly effective in capturing highly correlated values and useful patterns with time-frequency representation of audio signal data [45]. In the Mel-scale filter banks (spectrogram) of drone's sound shown in Figure 4.10, it can be observed that the drone sound has noticeable invariance characteristics in the vicinity of 4.1 KHz. The convolution operation for input s is described according to the following equation (7):

$$a_{ij} = f\left(\sum_{m=0}^{M-1}\sum_{n=0}^{N-1} w_{mn} s_{(i+m)(j+n)} + b\right) \tag{7}$$

Where, a_{ij} represents output after convolution (the feature representation). f denotes the activation function (ReLU, Sigmoid, or Tanh). M and N denote the height and width of the kernel (filter), w is the convolution kernel weights, and b represents the bias offset. Further, (i, j) and (m, n) denote the position indices [45].

After several convolutional layers, a pooling layer is generally applied. The pooling layer aims to reduce the number of parameters in a convolutional neural network, it actually provides a form of non-linear down-sampling. The output of the pooling layer is given as follow:

$$p = \sigma(\acute{a}) \tag{8}$$

Where, p is the output and $\boldsymbol{\acute{a}}$ denotes the input of the pooling layer. The down-sampling operation over the receptive field is denoted by the function $\sigma(.)$, i.e. maximum or average function as illustrated in Figure 4.14 Figure 4.15. The input $\boldsymbol{\acute{a}}$ has dimensions $L \times L$, and the size of the output p is $\frac{L}{K} \times \frac{L}{K}$ where the receptive field size is $k \times k$ [45].

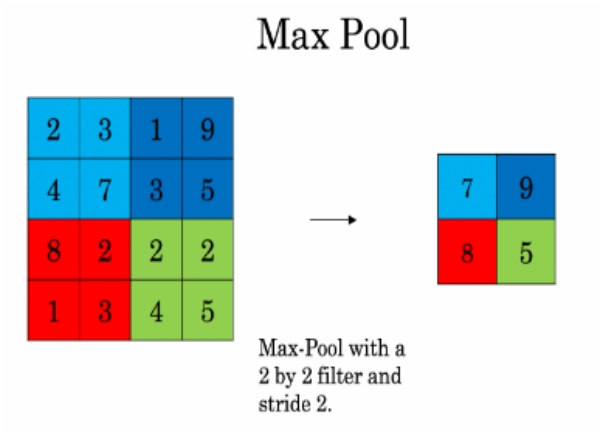

Figure 4.14. *Max pooling function*

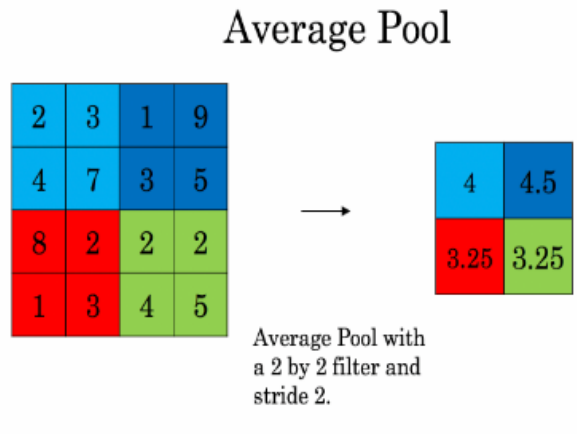

Figure 4.15. *Average pooling function*

After convolution and pooling operations, the multiple feature maps are aggregated and used as the input to the fully-connected layer. The operation in the fully-connected layer is described by the following equation (9):

$$a^l = f(w^l a^{l-1} + b^l) \tag{9}$$

Where, a^l and a^{l-1} are the output and input of layer l respectively. Function f denotes the activation function and l is the index of the l^{th} fully-connected layer.

Layer (type)	Output Shape	Param #
conv2d_1 (Conv2D)	(None, 9, 13, 16)	160
conv2d_2 (Conv2D)	(None, 9, 13, 32)	4640
conv2d_3 (Conv2D)	(None, 9, 13, 64)	18496
conv2d_4 (Conv2D)	(None, 9, 13, 128)	73856
max_pooling2d_1 (MaxPooling2	(None, 4, 6, 128)	0
dropout_1 (Dropout)	(None, 4, 6, 128)	0
flatten_1 (Flatten)	(None, 3072)	0
dense_1 (Dense)	(None, 128)	393344
dense_2 (Dense)	(None, 64)	8256
dense_3 (Dense)	(None, 32)	2080
dense_4 (Dense)	(None, 16)	528
dense_5 (Dense)	(None, 8)	136
dense_6 (Dense)	(None, 2)	18

Total params: 501,514
Trainable params: 501,514
Non-trainable params: 0

Figure 4.16. *CNN model's architecture*

Figure 4.16 shows our proposed CNN architecture which consists of thirteen layers. These layers are organized as four successive convolutional layers, one pooling layer, one dropout layer, one flatten layer, and six successive fully-connected (dense) layers. The first layer in the model is a convolution layer which performs a convolution over the input image (taken from Mel-scale filter banks or from MFCCs of drone's audio signals, the dimensions of the image depend on the feature extraction technique and frame size) with 16 kernels (filters) characterized by 3×3 Receptive Fields (RFs) with unitary depth and stride in both dimensions. The output of the first convolutional layer is passed to the second convolutional layer which has the same properties of the first layer except the number of kernels which is 32. The third and fourth convolutional layers are the same as the first and second with the exception that more kernels (64 and 128 respectively) are used in order to grant higher level representation. The obtained feature maps from the first four convolutional layers are then subsampled with a max-pooling layer that operates over 2×2 non overlapping squares. The output of the max pooling layer is then passed to a dropout layer, the main task of this layer is to prevent overfitting so it works as a regularization technique. After that, the feature maps are flattened from 2D into 1D in order to pass them to the fully-connected layers. We have 6 successive fully-connected layers; these layers have the same properties except the number of neurons in each layer (starting 128, 64, 32, 16, 8, and 2). Since the classification involves two different classes ('Drone', 'No Drone'), the last layer in the model is a softmax layer composed of two fully-connected neurons. The activation function used for kernels in all convolutional layers and fully-connected layers is the rectifier function. Therefore kernels are usually called rectifier linear units (ReLUs) [46].

In order to obtain the classification for an audio segment we split it into sequences (frames), then we average all prediction scores obtained for these sequences. The output of the CNN $y^{(i)}$ is now a vector which contains all class-wise prediction scores for the i^{th} sequence. The predicted class c^* for the whole audio segment is computed according to the following equation (10):

$$c^* = \underset{c}{argmax}\left[\frac{1}{M}\sum_{i=1}^{M} y_c^{(i)}\right] \tag{10}$$

Where, M is the total sequences number into which the audio segment is split and $y_c^{(i)}$ is the c^{th} entry of $y^{(i)}$ [46]. The proposed CNN architecture (model) is implemented using Keras library for Python and it is trained using the CPU unit. The total number of epochs is 10 and the batch size is 64. In order to train the model we use the categorical cross-entropy as a loss function, and we use the adaptive momentum (Adam) as an optimization algorithm to minimize the loss function [47].

4.3.3.2 Recurrent neural network model

Recurrent Neural Networks (RNNs) are among the most popular deep neural networks that are designed to make use of past information to feed forward the network. RNNs work perfectly and flexibly with time-series signals (sequence data) such as audio and video, they repeat the same task with memory so the context of the information is accumulated up to a given moment. The role of memory elements is to prevent the vanishing gradient problem which reduces the influence of past data [48].

In natural language processing and other sequence tasks RNNs are very effective because they have memory. At a given time instance t, RNNs read an input $x^{\langle t \rangle}$ (such as a word or audio segment), and remember previous information (context) that get passed from one time-step to the next. This allows a unidirectional RNN to process later inputs by taking information from the past, while a bidirectional RNN can process later inputs by taking information from the past and the future.

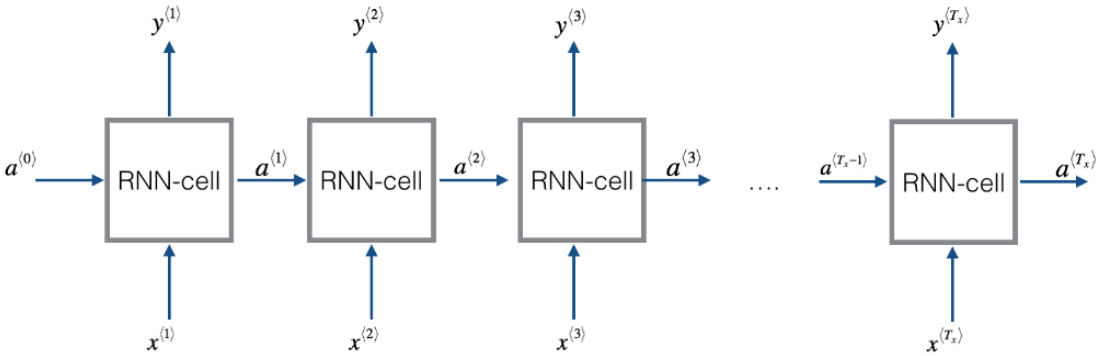

Figure 4.17. *RNN basic model*

Figure 4.17 shows a basic RNN which consists of a set of RNN cells. A recurrent neural network can be seen as a set of repeated cells. Figure 4.18 shows the basic RNN cell.

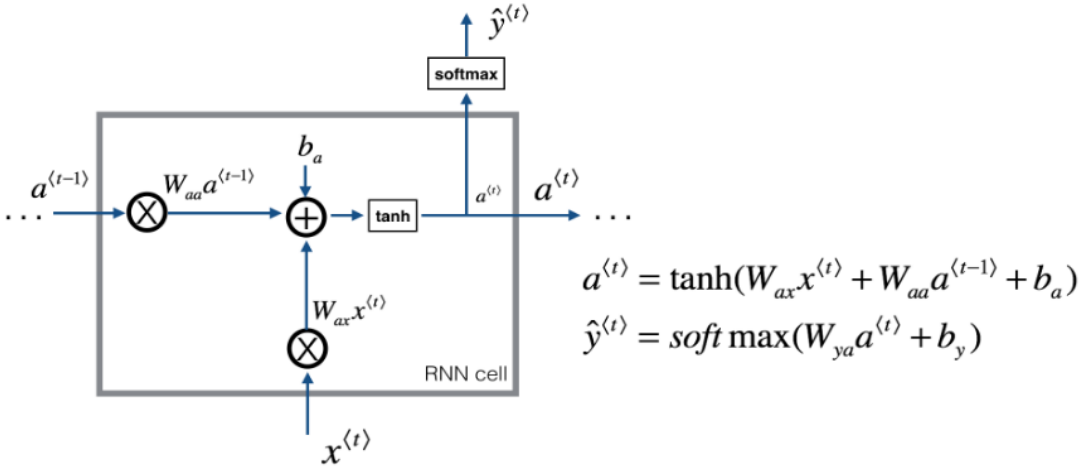

Figure 4.18. *Basic cell in recurrent neural network*

Where, $x^{\langle t\rangle}$ is the current input, $a^{\langle t-1\rangle}$ is the previous hidden state which contains information from the past, and $a^{\langle t\rangle}$ is the output that will be passed to the next RNN cell and it is also used to predict $y^{\langle t\rangle}$. The above mentioned recurrent neural network is called conventional RNN. Another RNN architecture is the Long Short-Term Memory (LSTM) that was specially designed to accurately model temporal sequences and their long-range dependencies [49]. LSTM is more effective than conventional RNN especially for acoustic modeling tasks, because LSTM overcomes some modeling weaknesses of RNNs [49].

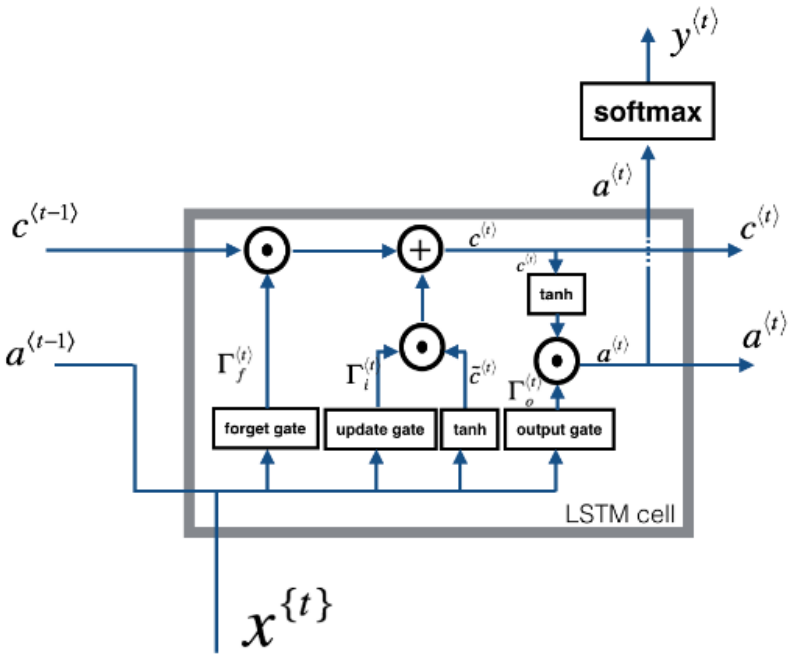

Figure 4.19. *LSTM cell architecture*

Figure 4.19 shows the architecture of LSTM cell where $x^{\langle t \rangle}$ is the input sequence, $a^{\langle t-1 \rangle}$ is the previous hidden state, and $c^{\langle t-1 \rangle}$ is the memory variable of the previous cell. $a^{\langle t \rangle}$ is the output of the cell that will be passed to the next cell and it is also used in $y^{\langle t \rangle}$ computation. $c^{\langle t \rangle}$ is the memory variable that will be passed to the next cell. As can be seen, LSTM contains a set of gates in order to control the flow of information. These gates are: forget gate, update gate, and output gate. In order to understand the working mechanism of LSTM, let's suppose that we want to use LSTM for keeping track of grammatical structures, like whether the subject is plural or singular, while reading words inside a piece of text. The task of each gate is listed below:

Forget gate: let's assume that the subject changes from a plural to a singular word, so we need a way to get rid of the previous memory value which is plural word. The forget gate in LSTM allows us to do this. The operation of forget gate is described by the following equation (11):

$$\Gamma_f^{\langle t \rangle} = \sigma\left(W_f\left[a^{\langle t-1 \rangle}, x^{\langle t \rangle}\right] + b_f\right) \tag{11}$$

Where, W_f are the weights that control the behavior of the forget gate, σ is sigmoid function. In the equation (11) $a^{\langle t-1 \rangle}$ and $x^{\langle t \rangle}$ are concatenated then multiplied by W_f then add a bias b_f, the resulting vector $\Gamma_f^{\langle t \rangle}$ is a vector with values between 0 and 1. The previous cell memory state vector $c^{\langle t-1 \rangle}$ will be multiplied element-wise by the output vector of the forget gate $\Gamma_f^{\langle t \rangle}$. So if a specific value in $\Gamma_f^{\langle t \rangle}$ is 0 (or close to 0), the LSTM will remove (forget) that piece of information (e.g. the plural subject) in the corresponding component of $c^{\langle t-1 \rangle}$. However, if the value in $\Gamma_f^{\langle t \rangle}$ is one, so the previous information corresponding to this value will be kept.

Update gate: once the plural subject has been forgotten, we need a way to update it to the new value that is now singular. The update gate in LSTM achieves this task. The operation of the update gate is described by the following equation:

$$\Gamma_u^{\langle t \rangle} = \sigma\left(W_u\left[a^{\langle t-1 \rangle}, x^{\langle t \rangle}\right] + b_u\right) \tag{12}$$

Where, W_u are weights control the behavior of update gate, the output of the update gate is a vector $\Gamma_{\boldsymbol{u}}^{\langle \boldsymbol{t} \rangle}$ with values between 0 and 1. This vector will be multiplied element-wise by $\tilde{c}^{\langle t \rangle}$ in order to compute $c^{\langle t \rangle}$. In order to update the new subject (e.g.

the singular subject), a new vector of numbers has been created which can be added to the previous cell memory variable $c^{\langle t-1 \rangle}$. This vector is $\tilde{c}^{\langle t \rangle}$ and can be computed according to the following equation:

$$\tilde{c}^{\langle t \rangle} = tanh\left(W_c\left[a^{\langle t-1 \rangle}, x^{\langle t \rangle}\right] + b_c\right) \quad (13)$$

Finally, the new memory variable $c^{\langle t \rangle}$ of current LSTM cell is calculated according to following equation:

$$c^{\langle t \rangle} = \Gamma_f^{\langle t \rangle} * c^{\langle t-1 \rangle} + \Gamma_u^{\langle t \rangle} * \tilde{c}^{\langle t \rangle} \quad (14)$$

Output gate: now it is time to calculate the output of the current LSTM cell according to the following equations:

$$\Gamma_o^{\langle t \rangle} = \sigma\left(W_o\left[a^{\langle t-1 \rangle}, x^{\langle t \rangle}\right] + b_o\right) \quad (15)$$

$$a^{\langle t \rangle} = \Gamma_o^{\langle t \rangle} * tanh\left(c^{\langle t \rangle}\right) \quad (16)$$

Layer (type)	Output Shape	Param #
lstm_1 (LSTM)	(None, 9, 128)	72704
lstm_2 (LSTM)	(None, 9, 128)	131584
dropout_2 (Dropout)	(None, 9, 128)	0
time_distributed_1 (TimeDist	(None, 9, 64)	8256
time_distributed_2 (TimeDist	(None, 9, 32)	2080
time_distributed_3 (TimeDist	(None, 9, 16)	528
time_distributed_4 (TimeDist	(None, 9, 8)	136
flatten_2 (Flatten)	(None, 72)	0
dense_11 (Dense)	(None, 2)	146

Total params: 215,434
Trainable params: 215,434
Non-trainable params: 0

Figure 4.20. *Proposed RNN architecture*

Figure 4.20 shows the proposed LSTM-RNN acoustic model, it involves 10 successive layers. It starts with two successive LSTM layers; these two layers have the same properties in terms of dimensionality of the output space (128) and returning the last output in the output sequence. After two LSTM layers, a dropout layer is needed for

regularization in order to prevent overfitting. The output of dropout layer is then passed to 4 successive (TimeDistributed) layers. This type of layers is a wrapper applies a layer (Dense in our model) to every temporal slice of an input, after that the feature maps are flattened from 2D to 1D in order to pass them to the final fully-connected layer which has 2 neurons. The activation function used in Dense layers is the rectifier ReLU function. The proposed LSTM-RNN model is also implemented using Keras library for python and it is trained using the CPU unit. We used the same settings of the CNN model in training the LSTM-RNN model in terms of the number of epochs 10, batch size 64, and optimization algorithm (Adam). In order to obtain the classification of an audio segment, we follow the same steps previously mentioned in the CNN model.

4.4. Implementation Tools

The implementation tools that we used during implementation and testing are divided into two main groups. The software tools include the required programming frameworks, libraries, and development environment. On the other hand, hardware tools that include the devices and components we used in data acquisition, such as the sound card, several drones, and microphones.

4.4.1. Software tools

4.4.1.1 Python

Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Python comes with a plenty of packages for data analysis and manipulation. It is very attractive for use as a glue language to connect existing components, as well as for rapid application development because it is combined with dynamic typing and dynamic binding, and because its built-in data structures [50]. In the field of artificial intelligence, machine learning, and deep learning it is highly important to choose the appropriate programming language. We have many choices such as MRTLAB, C++, Java, and Python; but we choose Python because it is an intuitive coding language and it comes with a full-features library line called frameworks. These frameworks reduce the time required to get perfect results. The Python version that we used is Python 3.7 and we installed it with Anaconda.

4.4.1.2 Anaconda

Anaconda is a data processing and scientific computing platform based on Python. In other words, Anaconda is a Python distribution integrated with installation

and package management tools; it runs on Linux, Windows, and Mac OS. Anaconda offers everything we need for machine learning, deep learning, and data science [51]. It includes the following components:

- The core Python language.
- More than 100 Python packages.
- Spyder and Jupyter notebook editors.
- Conda which is the Anaconda package manager used for installing and updating Anaconda packages.

Numpy, Scipy, and Pandas are examples of Anaconda's Python packages. Numpy is a library (package) for numerical computation. Scipy is a package for scientific computation, and Pandas is a package for data analysis. Figure 4.21 shows Anaconda navigator environment.

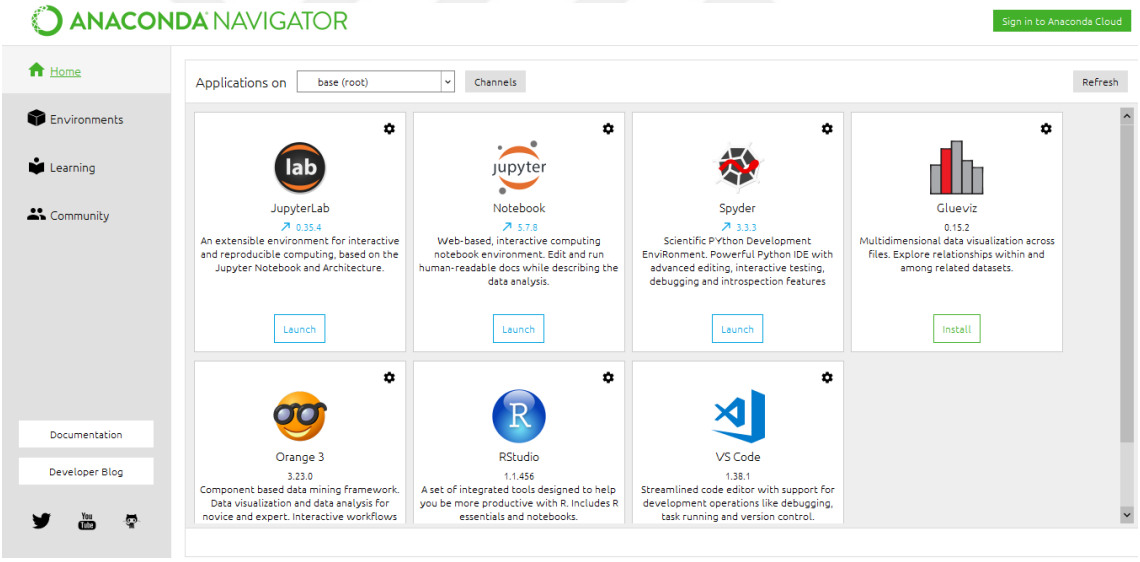

Figure 4.21. *Anaconda navigator environment*

4.4.1.3. Tensorflow

Tensorflow is an end-to-end open source for machine learning. Due to its libraries and community resources, in addition to its comprehensive and flexible ecosystem tools, Tensorflow let researchers push the state-of-the-art in ML and develop ML-powered applications. Tensorflow is created by Google Brain team, and utilizes Python to provide an appropriate API for building applications [52]. Tensorflow provides developers with a set of tools to create dataflow graphs (structures) describing the flow of data through a series of processing unites (nodes), where each node is a mathematical

operation, and connections between nodes are multidimensional data arrays or called tensors [52]. We use Tensorflow 1.13 since it is compatible with Python 3.7. There is a special version of Tensorflow called Tensorflow GPU that has been designed for computers provided with GPU unites to accelerate performance.

4.4.1.4. Keras

Keras is an open-source library for building and developing neural networks, it is written in Python and able to run over Tensorflow. It is especially designed to make researchers and developers able to go from idea to result within the least possible delay enabling fast experimentation [53]. Keras is an appropriate library for deep learning applications for the following reasons:

- User friendliness, modularity, and extensibility of Keras provide developers (users) with easy and fast prototyping.
- Keras supports both CNN and RNN, in addition to any combination of the two.
- Keras runs seamlessly on both CPUs and GPUs.

The version of Keras that we use is 2.2.4 which is compatible with Python 3.7 and Tensorflow 1.13.

4.4.1.5. Scikit-learn

Scikit-learn or Sklearn is a high level machine learning library written in Python which provides many supervised and unsupervised learning algorithms. It is simple and effective tools for data analysis and data mining, and it has been built upon Numpy, Scipy, and Matplotlib [54]. Scikit-learn provides the following functionalities:

- Regression such as linear and logistic regression.
- Classification such as support vector machine and k-nearest neighbor.
- Model selection.
- Preprocessing such as min-max normalization.

4.4.1.6. Cubase LE

Cubase LE is a compact version of Cubase Pro which uses the same core technologies, and which provides the essential tools for recording, editing, and mixing. In the computer-based production world, Cubase is the perfect entry with easy-to-use

software tools [55]. The version that we use in recording our data is Cubase LE 5 that actually comes with the TASCAM US-1800 device.

4.4.2. Hardware tools

4.4.2.1. TASCAM US-1800

It is an audio interface from TASCAM with many inputs more than any device existed in the market in its class. This interface offers up to 16 inputs (14 Analog and 2 Digital) with 96 KHz/24-bit audio resolution, and 4 outputs that are transmitted to Windows or Mac over high-speed USB 2.0 connection. The US-1800 interface provides: 8 XLR microphone inputs with phantom power and 60 dB of clean gain, 6 balanced line inputs, stereo S/PDIF, and digital and MIDI in and out [8]. TASCAM US-1800 is the perfect choice if we want the most inputs with the minimum amount of cash into our computer. The US-1800 inputs and outputs are shown in Figure 4.22.

Figure 4.22. *TASCAM US-1800 audio interface*

4.4.2.2. AT2031 microphone

For professional recording studios, stringed instruments in critical studios, and live applications the AT2031 microphone is the perfect and ideal microphone. Due to its permanently polarized element which offers extended frequency response with a slight rise in the high frequency range, it provides a more detailed sound. This microphone offers a wide dynamic range due to its high SPL handling capability and low self-noise making it perfectly suited for the most demanding applications [56]. The AT2031 microphone has the following features:

- Wide dynamic range and high SPL handling.
- Excellent, smooth, and extended frequency response with a slight rise in the high frequency region.
- Improved isolation of desired sound source due to its centroid polar pattern.

- Low self-noise making it a perfect choice for digital recordings.
- Switchable 150 Hz 6 dB/oct high-pass filter.
- Rugged construction and outstanding performance.

The hardware tools that we use are illustrated in Figure 4.23, and the array installation for collecting drones' records is illustrated in Figure 4.24.

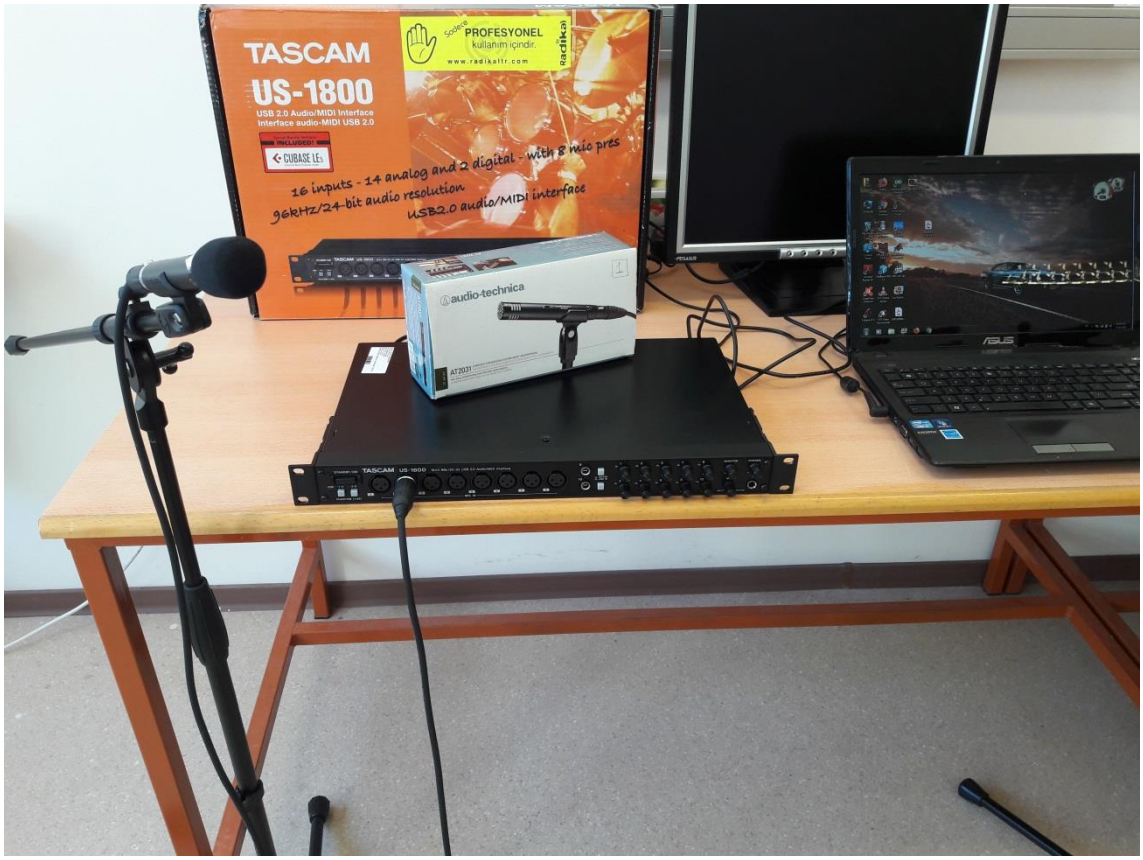

Figure 4.23. *The hardware tools US-1800 and AT2031*

Figure 4.24. *Installing an array of 5 AT2031 microphones with US1800 sound card*

4.5. Implementation Results

4.5.1. Classification assessment

In order to evaluate classification models, we have several measures. The majority of these measures are scalar metrics with some graphical methods. In classification models, we use the training data to build a classifier (classification model) which can estimate (predict) the class label for a new sample. In order to evaluate different learning algorithms, the output of these algorithms must be assessed and analyzed carefully and the interpretation of this analysis must be correct. Any classification process includes three main phases called training phase, validation phase, and test phase. During the training phase the parameters of the model are adjusted in order to find the best values yielding to the minimum error. The goal of learning algorithm is to learn from the training data for predicting classes of unseen data, and this is called the training phase. Classification problems are categorized into binary

classification where there are only two class labels, and multi-label classification where the number of class labels is greater than two.

In the case of imbalanced datasets where the number of samples of one class is greater or smaller than the number of samples of the other class(es), several assessment methods are affected by this imbalance.

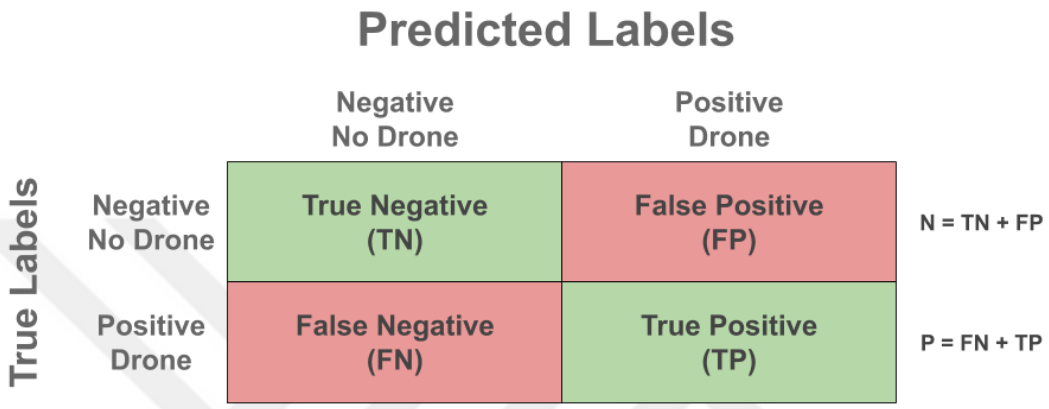

Figure 4.25. *Confusion matrix of our system*

Let's consider the confusion matrix in Figure 4.25 above. The ratio between positive and negative samples (P/N) called class distribution which represents the relationship between the right and left columns. So any classification assessment metric that uses values from both columns of the confusion matrix is sensitive to imbalanced datasets [57]. Some evaluation metrics, such as precision and accuracy, use values from both columns, so any change in the data distribution will change these metrics.

4.5.1.1. Accuracy and error rate

Accuracy is the ratio of the correctly classified samples to the total number of samples, and it is the most commonly used metric for classification assessment. The accuracy is given according o the following equation:

$$Acc = \frac{TP + TN}{TP + TN + FP + FN} \tag{17}$$

Where, N and P denote the number of negative and positive samples respectively. The Error Rate (ERR) is defined as the complement of the accuracy metric, and it represents the number of misclassified samples from both negative and positive classes. ERR is computed according to the following equation:

$$ERR = 1 - Acc = \frac{FP + FN}{TP + TN + FP + FN} \quad (18)$$

The problem of ERR and Accuracy is that they are sensitive to imbalanced datasets.

4.5.1.2. Sensitivity and specificity

Sensitivity represents the correctly classified positive samples to the total number of positive samples, it is also called True Positive Rate (TPR), recall, or hit rate. The sensitivity is given according to the following equation:

$$TPR = \frac{TP}{TP + FN} = \frac{TP}{P} \quad (19)$$

$$TNR = \frac{TN}{TN + FP} = \frac{TN}{N} \quad (20)$$

On the other hand, specificity defined by the ratio of the correctly classified negative samples to the total number of negative samples, specificity can be calculated according to equation (20), and it is also called True Negative Rate (TNR) or inverse recall. In other words, sensitivity represents the portion of the positive samples that were correctly classified, and specificity represents the portion of the negative samples that were correctly classified. Sensitivity depends on TP and FN that are taken from the same column in the confusion matrix, and specificity depends on TN and FP that are taken from the same column in the confusion matrix. The sensitivity (TPR or recall) and specificity (TNR or inverse recall) are suitable for classification assessment when we have imbalanced dataset.

4.5.1.3. False positive rate and false negative rate

False Positive Rate (FPR) represents the ratio of the incorrectly classified negative samples to the total number of negative samples. It is also called fallout or false alarm rate. FPR complements the specificity and represents the portion of negative samples that were incorrectly classified. FPR is calculated according to the following equation:

$$FPR = 1 - TNR = \frac{FP}{TN + FP} = \frac{FP}{N} \quad (21)$$

False Negative Rate (FNR) represents the ratio of incorrectly classified positive samples to the total number of positive samples, it is also called miss rate and it represents the portion of positive samples that were incorrectly classified.

FNR is calculated according to the following equation (22). Both FPR and FNR are suitable for using when we have imbalanced datasets.

$$FNR = 1 - TPR = \frac{FN}{TP + FN} = \frac{FN}{P} \tag{22}$$

4.5.1.4. Predictive values

Positive Prediction Value (PPV) represents the ratio of the correctly classified positive samples TP to the total number of positive predicted samples. It is also called precision and given according to the equation (23):

$$PPV = Precision = \frac{TP}{TP + FP} \tag{23}$$

Negative Prediction Value (NPV) represents the ratio of the correctly classified negative samples to the total number of negative predicted samples. It is also called inverse precision and given according to equation (24):

$$NPV = Inverse\ Precision = \frac{TN}{TN + FN} \tag{24}$$

4.5.1.5. Receiver operating characteristics

Receiver Operating Curve (ROC) is a two dimensional curve in which the FPR is the X-axis and the TPR is the Y-axis. The ROC curve has been used for evaluating many systems such as medical decision-making systems, diagnostic systems, and machine learning systems [58]. The ROC curve makes balance between costs and benefits, costs represented by false positives and benefits represented by true positives. Our classification models and many other classifiers produce only class decisions for each sample of the test data, and hence produce only one confusion matrix which corresponds to one point in the ROC space. However, there are many techniques, such as class portions and using combinations of scoring and voting, are used to generate full ROC curve [58]. An example of ROC curve is illustrated in Figure 4.26, in this curve there are four important points A (lower left), B (upper left), C (upper right), and D (lower right). The point A (0,0) represents a classifier which classifies all negative

samples correctly while there is no positive classification. The point D (1,0) represents a classifier which misclassifies all negative and positive samples. The point B (0,1) represents a classifier which classifies all negative and positive samples correctly, and this point represents the ideal operating point or the perfect classifier. The point C (1,1) represents a classifier which classifies all positive samples correctly while misclassifies all negative samples.

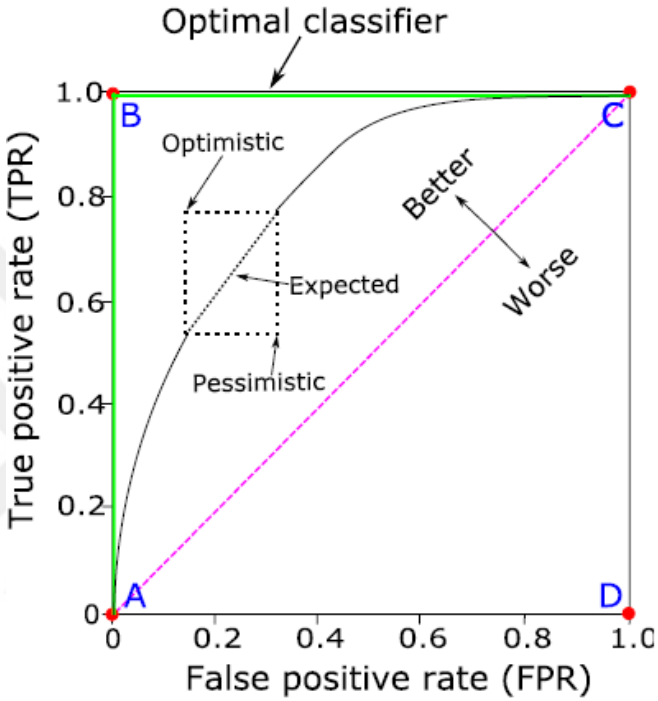

Figure 4.26. *Receiver operating curve (ROC)*

4.5.1.6. Area under the ROC curve

Area Under Curve (AUC) is a scalar value (metric) which determines the area under the ROC curve in order to compare different classifiers. The AUC metric takes any value between 0 and 1, and there is not any realistic classifier with an AUC less than 0.5. Two classifiers may have different ROC curves but they have the same AUC score [57].

4.5.1.7. Precision-recall curve

Precision-Recall (PR) curve is a two dimensional curve in which the X-axis represents the recall (TPR) and the Y-axis represents the precision (PPV), so it shows the relationship between recall and precision that are widely used for classification performance evaluation. The PR curve has the same concept of ROC curve, but in PR curve there is no need for the TN value.

4.5.2. Experimental results

In this section, we conducted a set of experiments in order to evaluate the performance of the two proposed acoustic models with different parameters settings. In these experiments we use our built dataset which is divided into two main sets, a training set that contains 9380 audio clips and a test set containing 1040 audio clips. We group the audio clips into two different sets because it is really important not to train and evaluate the model's performance on audio clips coming from the same location, since this would falsify the generalization score. The lengths of audio clips in both train and test sets are different. Figure 4.27 and Figure 4.28 show the audio clips lengths distribution in train and test sets respectively.

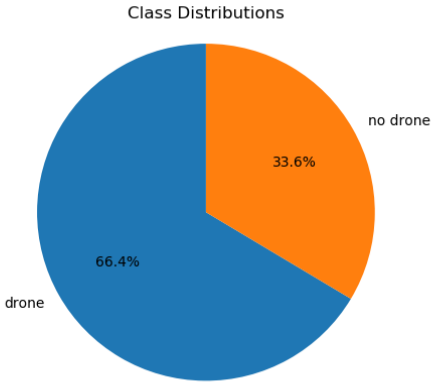

Figure 4.27. *Distribution of audio clips lengths in train set*

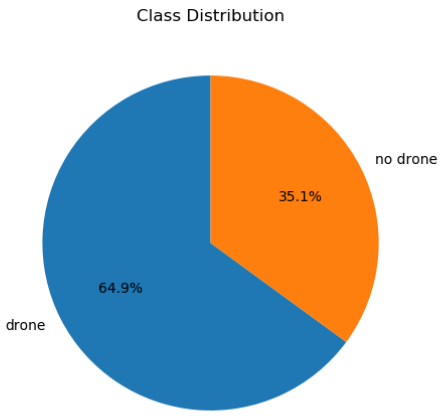

Figure 4.28. *Distribution of audio clips lengths in test set*

The acquired data is classified by implementing MFCCs and Mel-scale filter banks, and deep neural networks. As a baseline approach we initialized the feature extraction parameters and the network hyper-parameters to the following common values:

- Acoustic features: MFCCs and Mel-scale filter banks
- Sampling rate: 16, 32, and 44.1 KHz
- Frame length: 100 ms
- Training period: 10 epochs
- Batch size: 64 samples
- Optimization algorithm: Adaptive Moment Estimation (Adam).

For batch size it is really important to keep this number small for better generalization. The proposed acoustic model (RNN and CNN) are evaluated in terms of classification accuracy, precision, recall, f1-score, average score, area under ROC curve AUC, confusion matrix, ROC curve, and PR curve. The performance results of each model are presented in the following two sections.

4.5.2.1. RNN model's experimental results

We trained and this model 6 times, in each time we used different parameters settings. The results are listed below.

RNN with MFCCs, 16 KHz sampling rate, and 100 ms frame length: the total time that is needed to train this model during 10 epochs is 42.6 minutes. The average time needed in each epoch is 256 seconds. The time needed in each epoch of the ten is [264 sec, 255, 255, 255, 254, 255, 255, 254, 255, and 254] from epoch 1 to 10 respectively. The total number of parameters is 215434. The loss and accuracy in both train and validation sets are shown in Figure 4.29. The criterion that is chosen to save the best model's parameters during training is the validation accuracy. The best recorded validation accuracy in epoch (7) Table 4.1. The validation set consists 10 % of the train set.

Table 4.1. *Best recorded loss and accuracy during training*

	Loss	Accuracy
Train set	0.1083	0.9612
Validation set	0.1040	0.9650

Let's see the performance of this model on test set (unseen data). The accuracy on test set is (0.9201), the average precision is (0.8570), and the AUC is (0.9365). The precision, recall, and f1-score for positive and negative classes are listed in Table 4.2.

Table 4.2. *Model's classification report*

Class	Precision	Recall	F1-score	Support
No drone	0.94	0.97	0.95	851
Drone	0.82	0.71	0.76	189

The confusion matrix of this model is illustrated in Figure 4.30, the ROC of this model is illustrated in Figure 4.31, and the PR curve is illustrated in Figure 4.32.

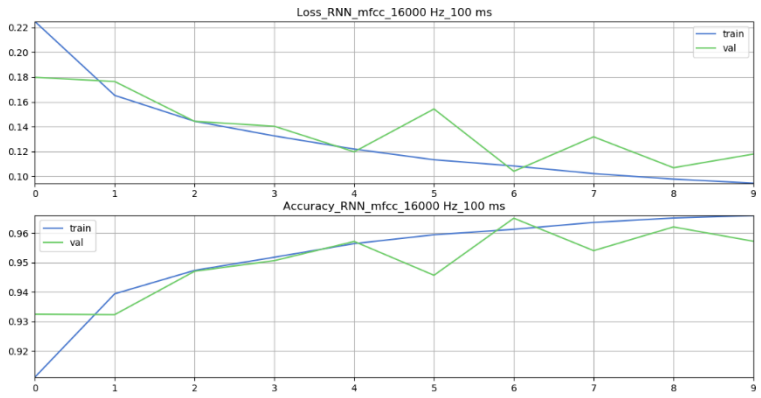

Figure 4.29. *Accuracy and loss during training*

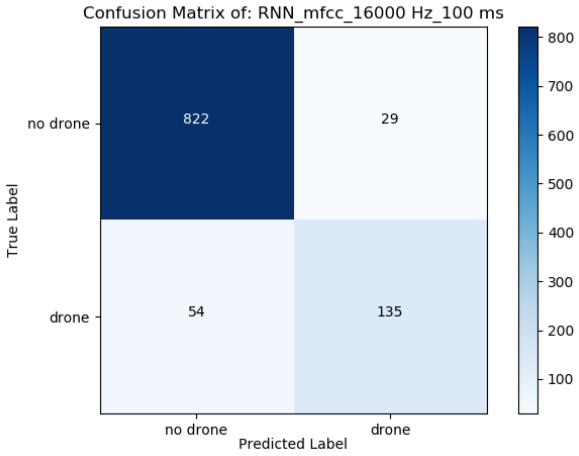

Figure 4.30. *Model's confusion matrix*

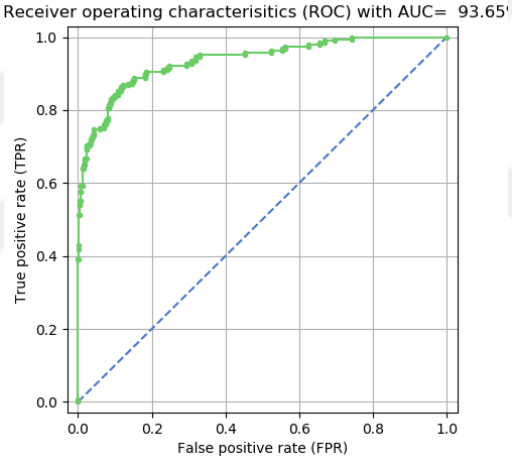

Figure 4.31. *Model's ROC curve with AUC= 93.65%*

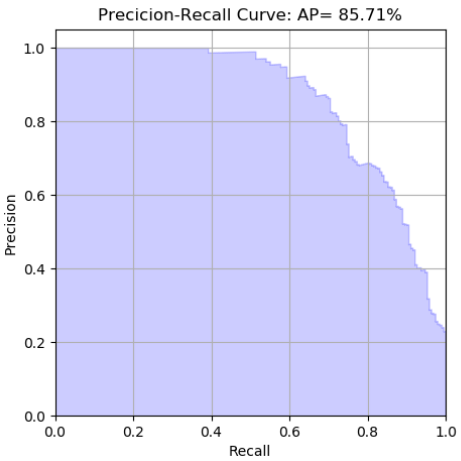

Figure 4.32. *Model's PR curve with average precision 85.71%*

RNN with filter banks, 16 KHz, and 100 ms frame length: the total time that is needed to train this model during 10 epochs is 43.2 minutes. The average time needed in each epoch is 260 seconds. The time needed in each epoch of the ten is [277 sec, 257, 257, 258, 257, 257, 257, 257, 257, and 262] from epoch 1 to 10 respectively. The total number of parameters is 222090. The loss and accuracy in both train and validation sets are shown in Figure 4.33. The criterion that is chosen to save the best model's parameters during training is the validation accuracy. The best recorded validation accuracy in epoch (9) Table 4.3. The validation set consists 10 % of the train set. Let's see the performance of this model on test set (unseen data). The accuracy on test set is (0.9557), the average precision is (0.9471), and the AUC is (0.9854). The precision, recall, and f1-score for positive and negative classes are listed in Table 4.4.

Table 4.3. *Best recorded loss and accuracy during training*

	Loss	Accuracy
Train set	0.1826	0.9295
Validation set	0.1560	0.9464

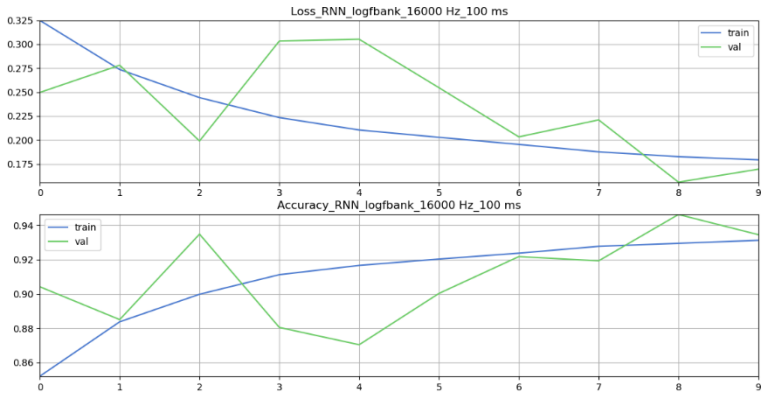

Figure 4.33. *Accuracy and loss during training*

Table 4.4. *Model's classification report*

Class	Precision	Recall	F1-score	Support
No drone	0.96	0.98	0.97	851
Drone	0.91	0.84	0.87	189

The confusion matrix of this model is illustrated in Figure 4.34, the ROC of this model is illustrated in Figure 4.35, and the PR curve is illustrated in Figure 4.36.

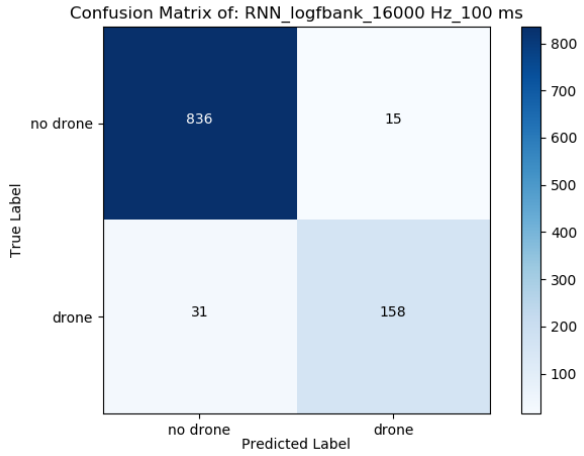

Figure 4.34. *Model's confusion matrix*

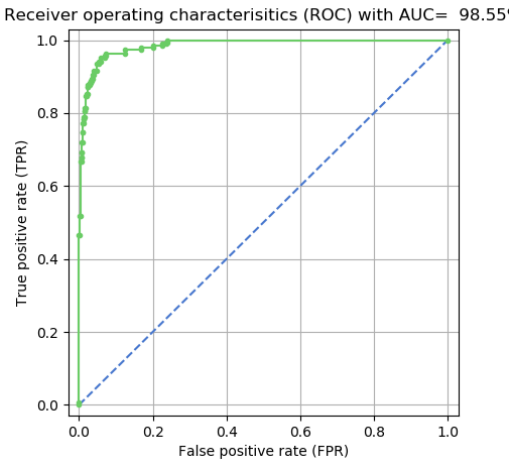

Figure 4.35. *Model's ROC curve with AUC= 98.55%*

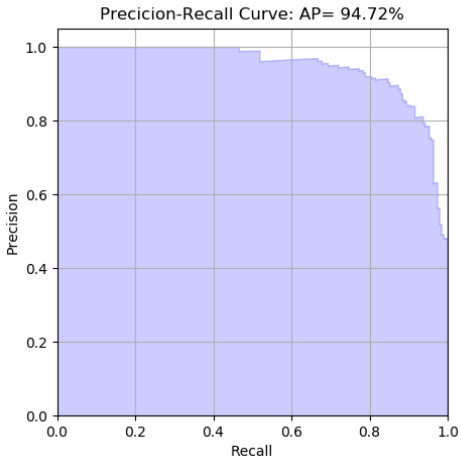

Figure 4.36. *Model's PR curve with average precision 94.72%*

RNN with MFCCs, 32 KHz sampling rate, and 100 ms frame length: the total time that is needed to train this model during 10 epochs is 42.65 minutes. The average time needed in each epoch is 260 seconds. The time needed in each epoch of the ten is [263 sec, 254, 255, 256, 254, 255, 255, 255, 255, and 257] from epoch 1 to 10 respectively. The total number of parameters is 215434. The loss and accuracy in both train and validation sets are shown in Figure 4.37. The criterion that is chosen to save the best model's parameters during training is the validation accuracy. The best recorded validation accuracy in epoch (7) Table 4.5. The validation set consists 10 % of the train set. Let's see the performance of this model on test set (unseen data). The accuracy on test set is (0.9730), the average precision is (0.9896), and the AUC is (0.9976). The precision, recall, and f1-score for positive and negative classes are listed in Table 4.6.

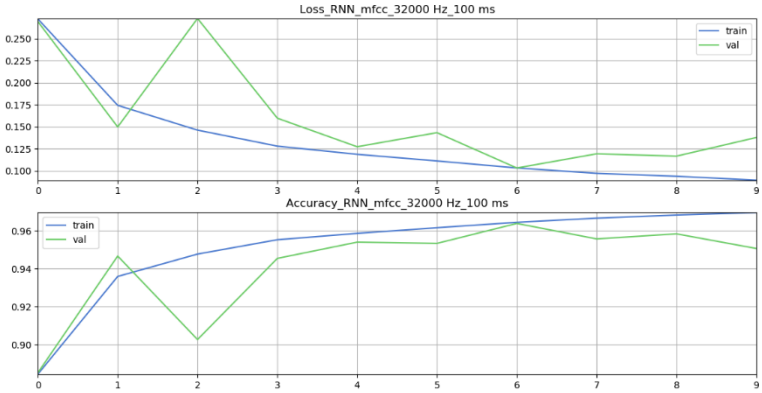

Figure 4.37. *Accuracy and loss during training*

Table 4.5. *Best recorded loss and accuracy during training*

	Loss	Accuracy
Train set	0.1031	0.9645
Validation set	0.1030	0.9639

The confusion matrix of this model is illustrated in Figure 4.38, the ROC of this model is illustrated in Figure 4.39, and the PR curve is illustrated in Figure 4.40.

Table 4.6. *Model's classification report*

Class	Precision	Recall	F1-score	Support
No drone	0.99	0.98	0.98	851
Drone	0.90	0.96	0.93	189

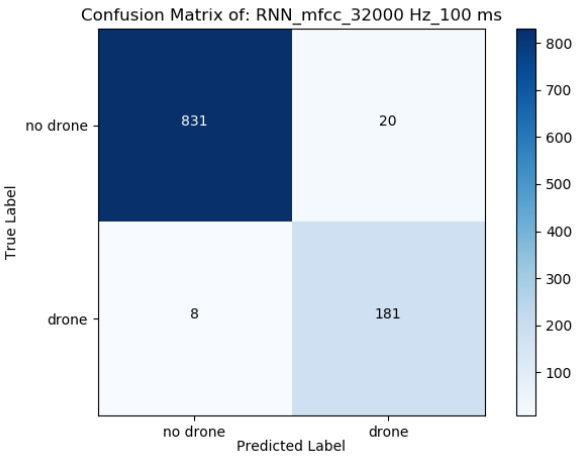

Figure 4.38. *Model's confusion matrix*

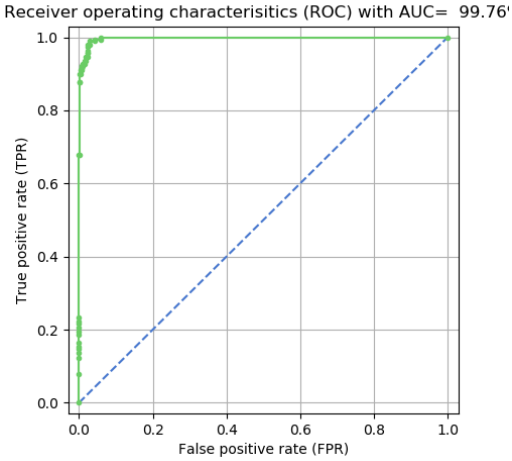

Figure 4.39. *Model's ROC curve with AUC= 99.76%*

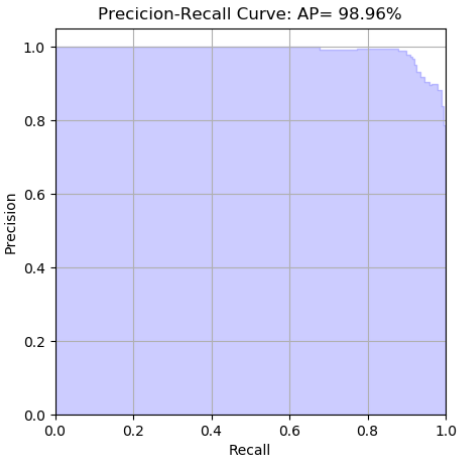

Figure 4.40. *Model's PR curve with average precision 98.96%*

RNN with filter banks, 32 KHz sampling rate, and 100 ms frame length: the total time that is needed to train this model during 10 epochs is 49.5 minutes. The average time needed in each epoch is 297 seconds. The time needed in each epoch of the ten is [344 sec, 291, 292, 292, 290, 290, 292, 292, 294, and 293] from epoch 1 to 10 respectively. The total number of parameters is 222090. The loss and accuracy in both train and validation sets are shown in Figure 4.41. The criterion that is chosen to save the best model's parameters during training is the validation accuracy. The best recorded validation accuracy in epoch (10) Table 4.7. The validation set consists 10 % of the train set. Let's see the performance of this model on test set (unseen data). The accuracy on test set is (0.9875), the average precision is (0.9948), and the AUC is (0.9988). The precision, recall, and f1-score for positive and negative classes are listed in Table 4.8.

Table 4.7. *Best recorded loss and accuracy during training*

	Loss	Accuracy
Train set	0.1250	0.9552
Validation set	0.1329	0.9525

The confusion matrix of this model is illustrated in Figure 4.42, the ROC of this model is illustrated in Figure 4.43, and the PR curve is illustrated in Figure 4.44.

Table 4.8. *Model's classification report*

Class	Precision	Recall	F1-score	Support
No drone	1	0.99	0.99	851
Drone	0.94	0.99	0.97	189

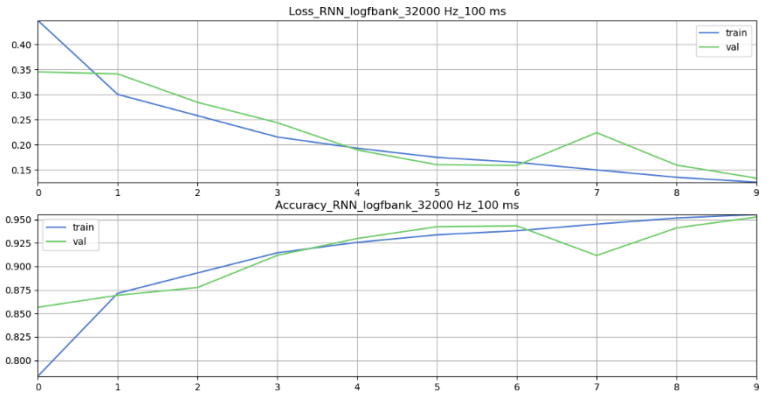

Figure 4.41. *Accuracy and loss during training*

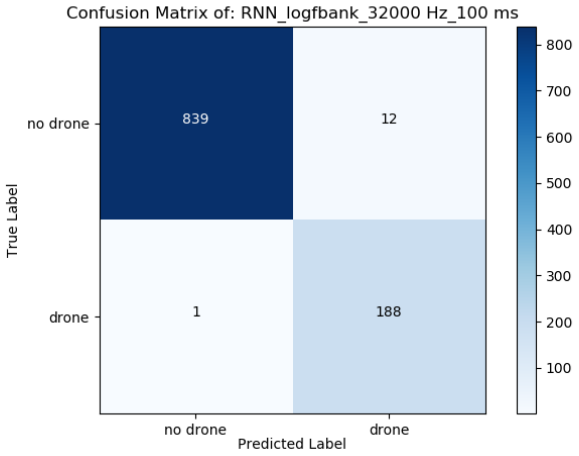

Figure 4.42. *Model's confusion matrix*

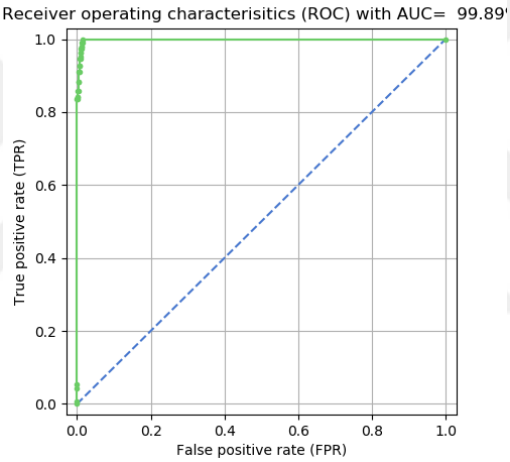

Figure 4.43. *Model's ROC curve with AUC= 99.89%*

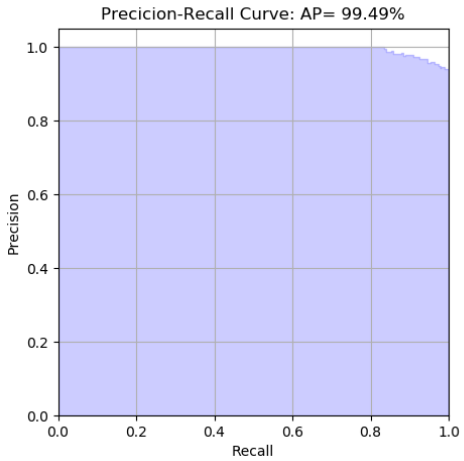

Figure 4.44. *Model's PR curve with average precision 99.49%*

RNN with MFCCs, 44.1 KHz sampling rate, 100 ms frame length: the total time that is needed to train this model during 10 epochs is 47.28 minutes. The average time needed in each epoch is 284 seconds. The time needed in each epoch of the ten is [310 sec, 280, 281, 281, 281, 280, 281, 281, 281, and 281] from epoch 1 to 10 respectively. The total number of parameters is 215434. The loss and accuracy in both train and validation sets are shown in Figure 4.45. The criterion that is chosen to save the best model's parameters during training is the validation accuracy. The best recorded validation accuracy in epoch (6) Table 4.9. The validation set consists 10 % of the train set. Let's see the performance of this model on test set (unseen data). The accuracy on test set is (0.9721), the average precision is (0.9865), and the AUC is (0.9956). The precision, recall, and f1-score for positive and negative classes are listed in Table 4.10.

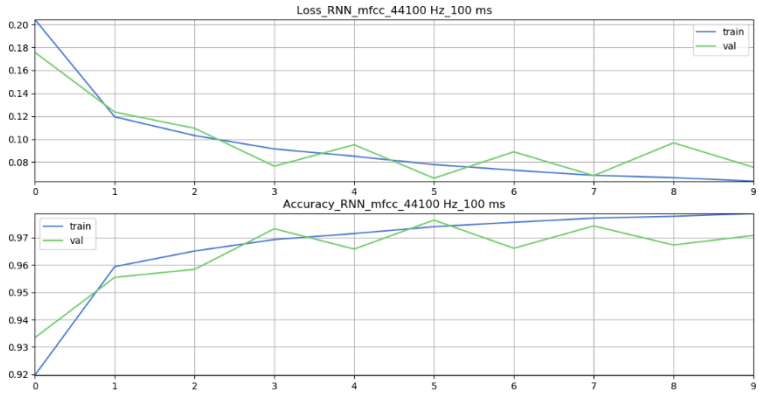

Figure 4.45. *Accuracy and loss during model training*

Table 4.9. *Best recorded loss and accuracy during training*

	Loss	Accuracy
Train set	0.0778	0.9740
Validation set	0.0659	0.9764

Table 4.10. *Model's classification report*

Class	Precision	Recall	F1-score	Support
No drone	0.99	0.98	0.98	851
Drone	0.90	0.95	0.93	189

The confusion matrix of this model is illustrated in Figure 4.46, the ROC of this model is illustrated in Figure 4.47, and the PR curve is illustrated in Figure 4.48.

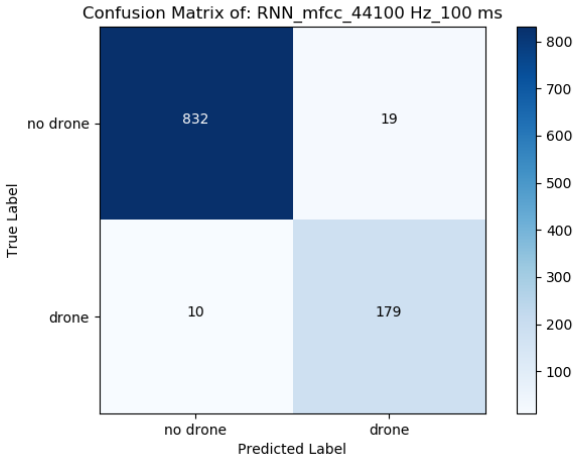

Figure 4.46. *Model's confusion matrix*

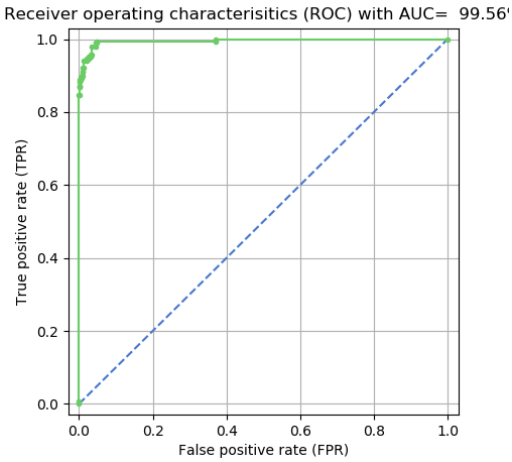

Figure 4.47. *Model's ROC curve with AUC= 99.56%*

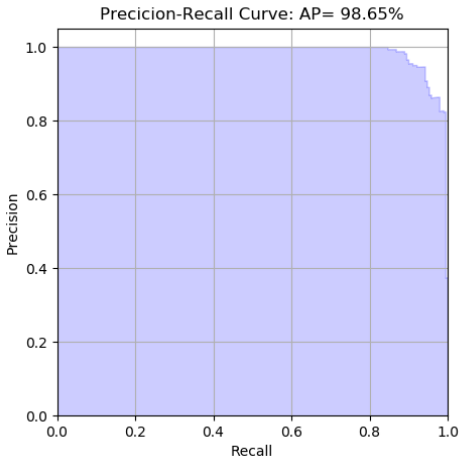

Figure 4.48. *Model's PR curve with average precision 98.65%*

RNN with filter banks, 44.1 KHz sampling rate, 100 ms frame length: the total time that is needed to train this model during 10 epochs is 48.95 minutes. The average time needed in each epoch is 294 seconds. The time needed in each epoch of the ten is [337 sec, 288, 290, 290, 290, 289, 289, 288, 288, and 288] from epoch 1 to 10 respectively. The total number of parameters is 222090. The loss and accuracy in both train and validation sets are shown in Figure 4.49. The criterion that is chosen to save the best model's parameters during training is the validation accuracy. The best recorded validation accuracy in epoch (7) Table 4.11. The validation set consists 10 % of the train set. Let's see the performance of this model on test set (unseen data). The accuracy on test set is (0.9644), the average precision is (0.9834), and the AUC is (0.9962). The precision, recall, and f1-score for positive and negative classes are listed in Table 4.12.

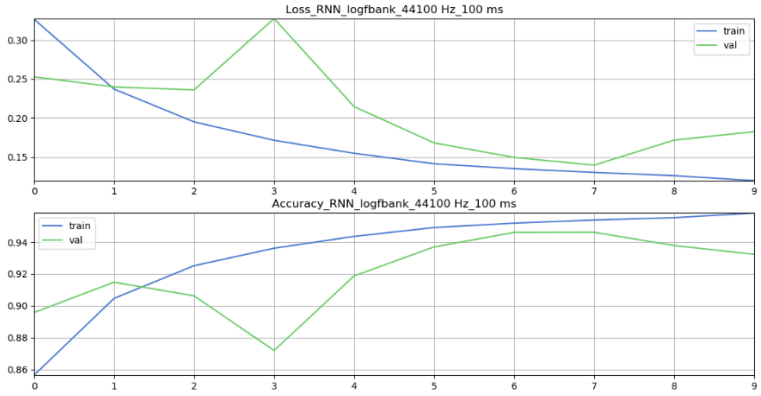

Figure 4.49. *Accuracy and loss during model training*

Table 4.11. *Best recorded loss and accuracy during training*

	Loss	Accuracy
Train set	0.1349	0.9519
Validation set	0.1493	0.9461

The confusion matrix of this model is illustrated in Figure 4.50, the ROC of this model is illustrated in Figure 4.51, and the PR curve is illustrated in Figure 4.52.

Table 4.12. *Model's classification report*

Class	Precision	Recall	F1-score	Support
No drone	0.99	0.96	0.98	851
Drone	0.86	0.97	0.91	189

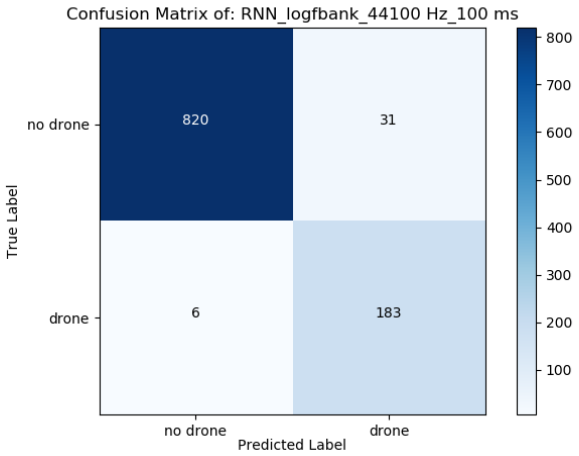

Figure 4.50. *Model's confusion matrix*

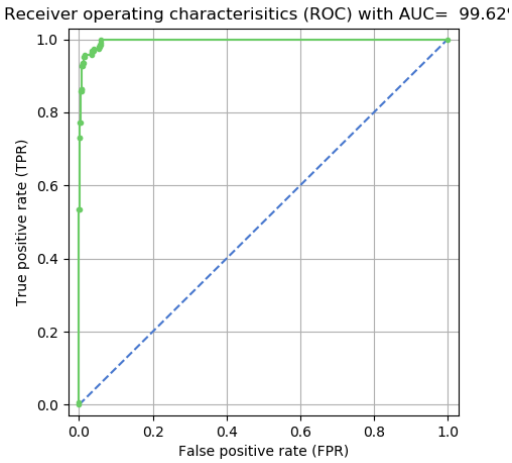

Figure 4.51. *Model's ROC curve with AUC= 99.62%*

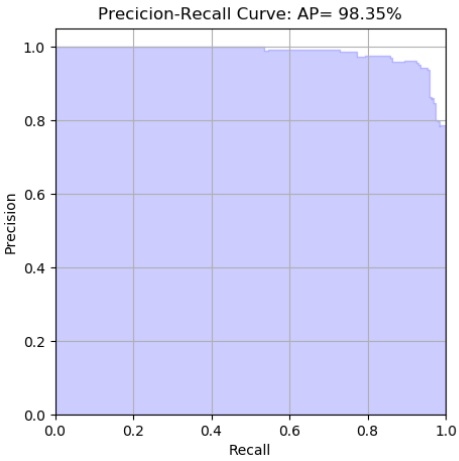

Figure 4.52. *Model's PR curve with average precision 98.35%*

4.5.2.2. CNN model's experimental results

We trained and this model 6 times, in each time we used different parameters settings. The results are listed below.

CNN with MFCCs, 16 KHz sampling rate, 100 ms frame length: the total time that is needed to train this model during 10 epochs is 174.5 minutes. The average time needed in each epoch is 1047 seconds. The time needed in each epoch of the ten is [1053 sec, 1043, 1049, 1048, 1049, 1043, 1042, 1042, 1052, and 1054] from epoch 1 to 10 respectively. The total number of parameters is 501514. The loss and accuracy in both train and validation sets are shown in Figure 4.53. The criterion that is chosen to save the best model's parameters during training is the validation accuracy. The best recorded validation accuracy in epoch (10) Table 4.13. The validation set consists 10 %

of the train set. Let's see the performance of this model on test set (unseen data). The accuracy on test set is (0.9409), the average precision is (0.9193), and the AUC is (0.9567). The precision, recall, and f1-score for positive and negative classes are listed in Table 4.14.

Table 4.13. *Best recorded loss and accuracy during training*

	Loss	Accuracy
Train set	0.0623	0.9777
Validation set	0.0732	0.9735

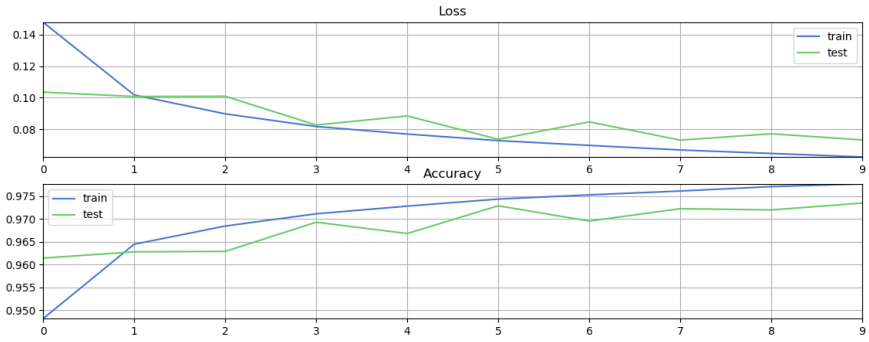

Figure 4.53. *Accuracy and loss during model's training*

Table 4.14. *Model's classification report*

Class	Precision	Recall	F1-score	Support
No drone	0.94	1	0.97	851
Drone	1	0.72	0.84	189

The confusion matrix of this model is illustrated in Figure 4.54, the ROC of this model is illustrated in Figure 4.55, and the PR curve is illustrated in Figure 4.56.

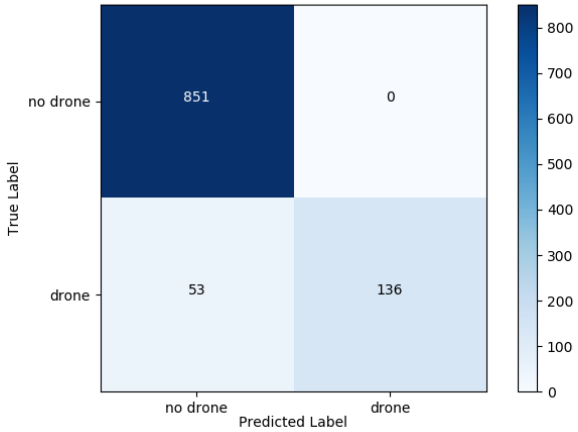

Figure 4.54. *Model's confusion matrix*

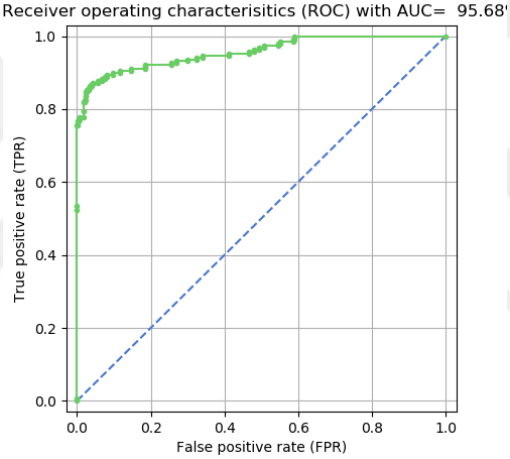

Figure 4.55. *Model's ROC curve with AUC= 95.68%*

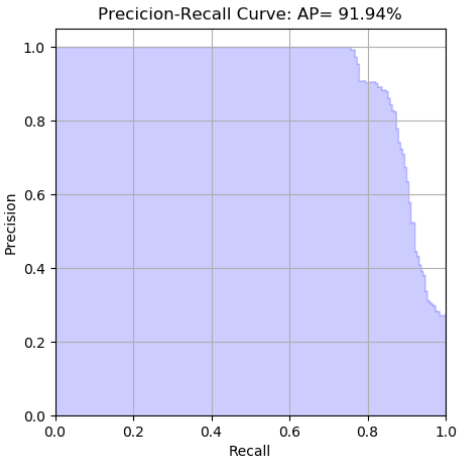

Figure 4.56. *Model's PR curve with average precision 91.94%*

CNN with filter banks, 16 KHz sampling rate, 100 ms frame length: the total time that is needed to train this model during 10 epochs is 363.61 minutes. The average time needed in each epoch is 2182 seconds. The time needed in each epoch of the ten is [2210 sec, 2199, 2179, 2175, 2174, 2176, 2179, 2175, 2176, and 2180] from epoch 1 to 10 respectively. The total number of parameters is 960266. The loss and accuracy in both train and validation sets are shown in Figure 4.57. The criterion that is chosen to save the best model's parameters during training is the validation accuracy. The best recorded validation accuracy in epoch (7) Table 4.15. The validation set consists 10 % of the train set. Let's see the performance of this model on test set (unseen data). The accuracy on test set is (0.9528), the average precision is (0.9144), and the AUC is (0.9351). The precision, recall, and f1-score for positive and negative classes are listed in Table 4.16.

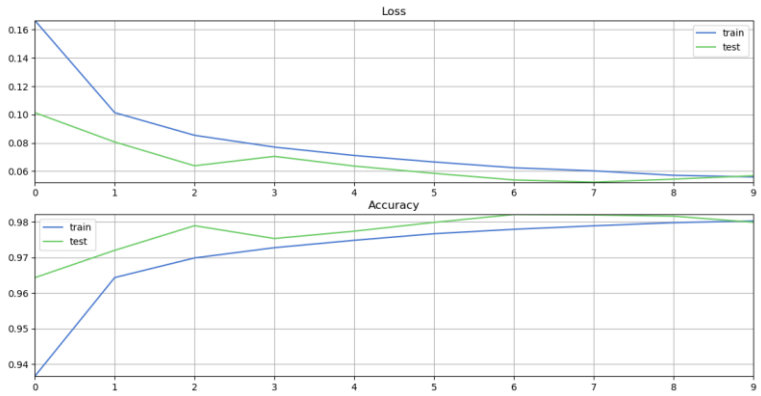

Figure 4.57. *Accuracy and loss during model's training*

Table 4.15. *Best recorded loss and accuracy during model's training*

	Loss	Accuracy
Train set	0.0625	0.9779
Validation set	0.0539	0.9820

Table 4.16. *Model's classification report*

Class	Precision	Recall	F1-score	Support
No drone	0.95	1	0.97	851
Drone	1	0.74	0.85	189

The confusion matrix of this model is illustrated in Figure 4.58, the ROC of this model is illustrated in Figure 4.59, and the PR curve is illustrated in Figure 4.60.

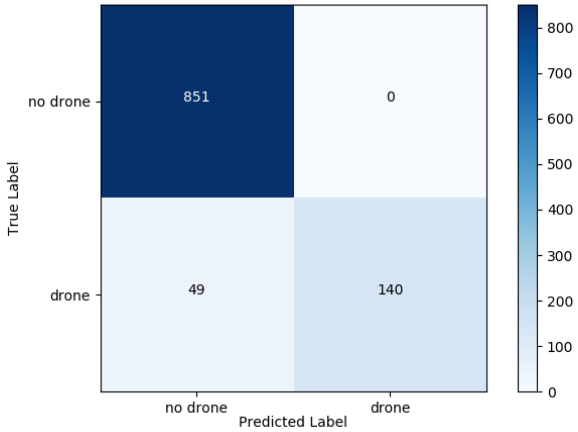

Figure 4.58. *Model's confusion matrix*

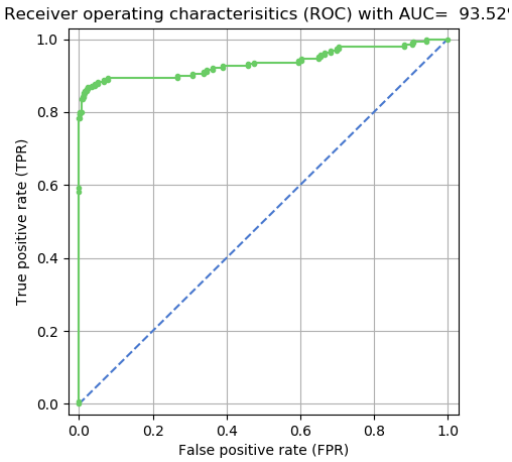

Figure 4.59. *Model's ROC curve with AUC= 93.52%*

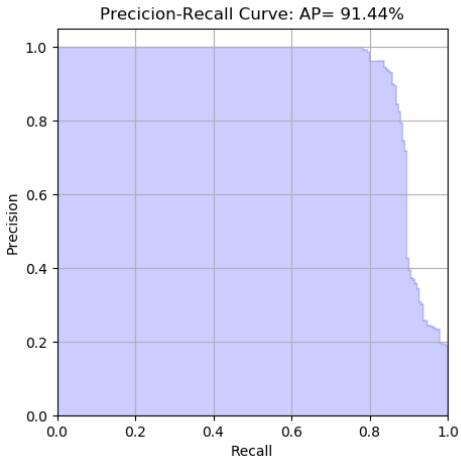

Figure 4.60. *Model's PR curve with average precision 91.44%*

CNN with MFCCs, 32 KHz sampling rate, 100 ms frame length: the total time that is needed to train this model during 10 epochs is 177.08 minutes. The average time needed in each epoch is 1062 seconds. The time needed in each epoch of the ten is [1061 sec, 1064, 1067, 1059, 1059, 1064, 1059, 1062, 1065, and 1065] from epoch 1 to 10 respectively. The total number of parameters is 501514. The loss and accuracy in both train and validation sets are shown in Figure 4.61. The criterion that is chosen to save the best model's parameters during training is the validation accuracy. The best recorded validation accuracy in epoch (9) Table 4.17. The validation set consists 10 % of the train set. Let's see the performance of this model on test set (unseen data). The accuracy on test set is (0.9846), the average precision is (0.9903), and the AUC is (0.9969). The precision, recall, and f1-score for positive and negative classes are listed in Table 4.18.

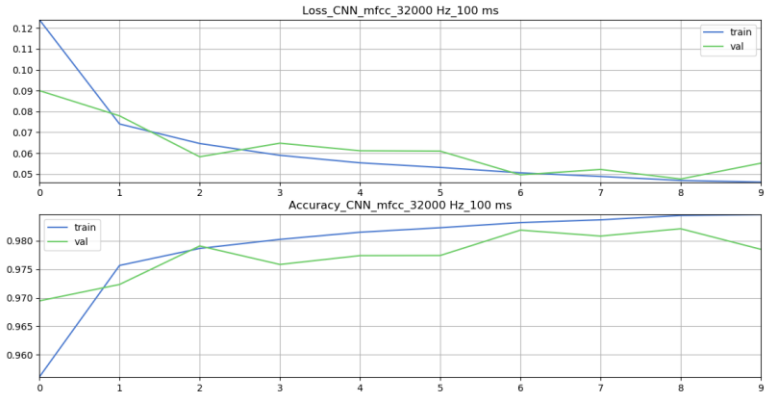

Figure 4.61. *Accuracy and loss during model's training*

Table 4.17. *Best recorded loss and accuracy during model's training*

	Loss	Accuracy
Train set	0.0469	0.9844
Validation set	0.0476	0.9821

The confusion matrix of this model is illustrated in Figure 4.62, the ROC of this model is illustrated in Figure 4.63, and the PR curve is illustrated in Figure 4.64.

Table 4.18. *Model's classification report*

Class	Precision	Recall	F1-score	Support
No drone	0.99	0.99	0.99	851
Drone	0.97	0.94	0.96	189

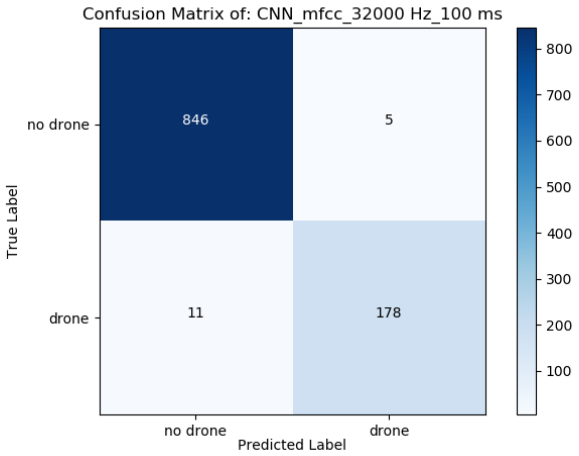

Figure 4.62. *Model's confusion matrix*

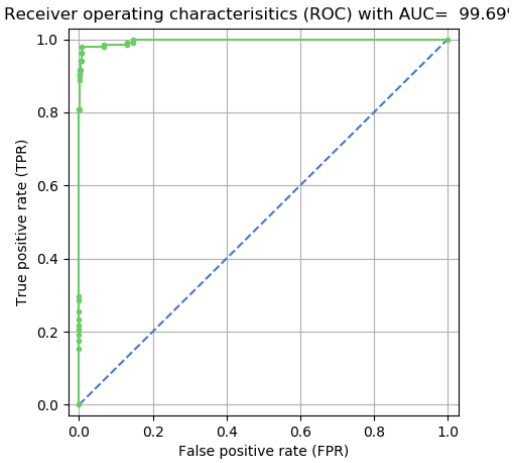

Figure 4.63. *Model's ROC curve with AUC= 99.69%*

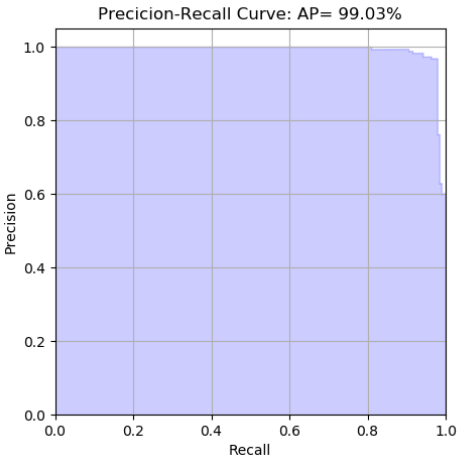

Figure 4.64. *Model's PR curve with average precision 99.03%*

CNN with filter banks, 32 KHz sampling rate, 100 ms frame length: the total time that is needed to train this model during 10 epochs is 366.85 minutes. The average time needed in each epoch is 2201 seconds. The time needed in each epoch of the ten is [2223 sec, 2199, 2197, 2197, 2196, 2196, 2211, 2198, 2197, and 2197] from epoch 1 to 10 respectively. The total number of parameters is 960266. The loss and accuracy in both train and validation sets are shown in Figure 4.65. The criterion that is chosen to save the best model's parameters during training is the validation accuracy. The best recorded validation accuracy in epoch (10) Table 4.19. The validation set consists 10 % of the train set. Let's see the performance of this model on test set (unseen data). The accuracy on test set is (0.9884), the average precision is (0.9853), and the AUC is

(0.9902). The precision, recall, and f1-score for positive and negative classes are listed in Table 4.20.

Table 4.19. *Best recorded accuracy and loss during model's training*

	Loss	Accuracy
Train set	0.0355	0.9884
Validation set	0.0363	0.9865

The confusion matrix of this model is illustrated in Figure 4.66, the ROC of this model is illustrated in Figure 4.67, and the PR curve is illustrated in Figure 4.68.

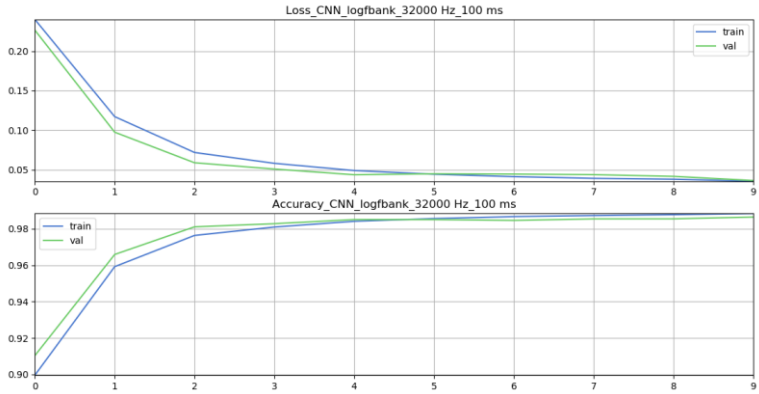

Figure 4.65. *Accuracy and loss during model's training*

Table 4.20. *Model's classification report*

Class	Precision	Recall	F1-score	Support
No drone	0.99	0.99	0.99	851
Drone	0.97	0.97	0.97	189

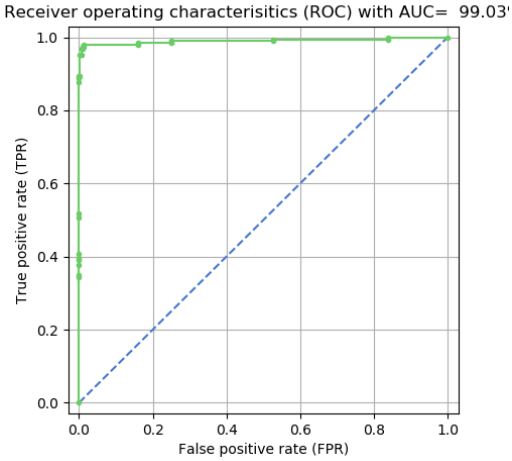

Figure 4.66. *Model's ROC curve with AUC= 99.3%*

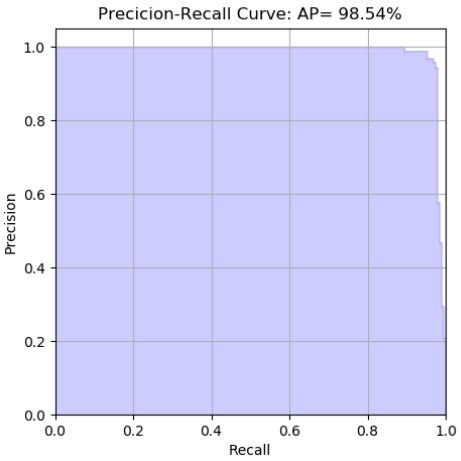

Figure 4.67. *Model's PR curve with average precision 98.54%*

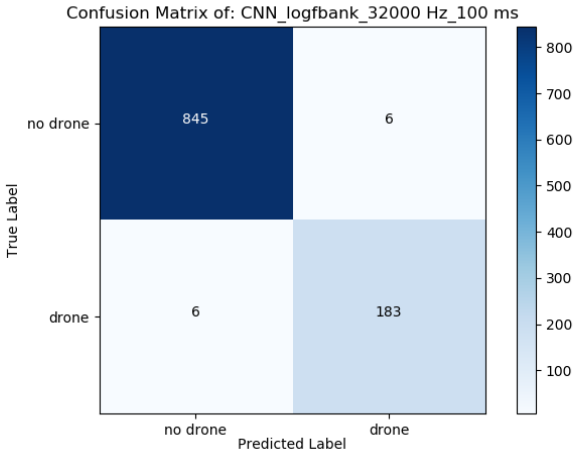

Figure 4.68. *Model's confusion matrix*

CNN with MFCCs, 44.1 KHz sampling rate, 100 ms frame length: the total time that is needed to train this model during 10 epochs is 177.46 minutes. The average time needed in each epoch is 1065 seconds. The time needed in each epoch of the ten is [1072 sec, 1060, 1064, 1068, 1063, 1083, 1061, 1061, 1058, and 1058] from epoch 1 to 10 respectively. The total number of parameters is 501514. The loss and accuracy in both train and validation sets are shown in Figure 4.69. The criterion that is chosen to save the best model's parameters during training is the validation accuracy. The best recorded validation accuracy in epoch (8) Table 4.21. The validation set consists 10 % of the train set. Let's see the performance of this model on test set (unseen data). The accuracy on test set is (0.9855), the average precision is (0.9832), and the AUC is (0.9921). The precision, recall, and f1-score for positive and negative classes are listed in Table 4.22.

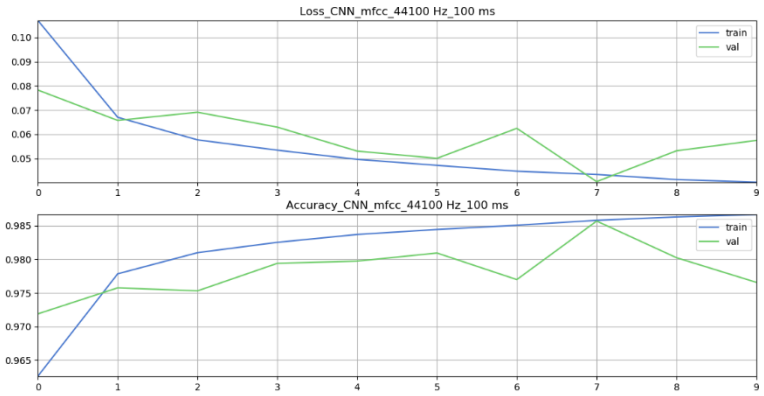

Figure 4.69. *Accuracy and loss during model's training*

Table 4.21. *Best recorded loss and accuracy during model's training*

	Loss	Accuracy
Train set	0.0435	0.9858
Validation set	0.0405	0.9857

Table 4.22. *Model's classification report*

Class	Precision	Recall	F1-score	Support
No drone	0.99	0.99	0.99	851
Drone	0.96	0.96	0.96	189

The confusion matrix of this model is illustrated in Figure 4.70, the ROC of this model is illustrated in Figure 4.71, and the PR curve is illustrated in Figure 4.72.

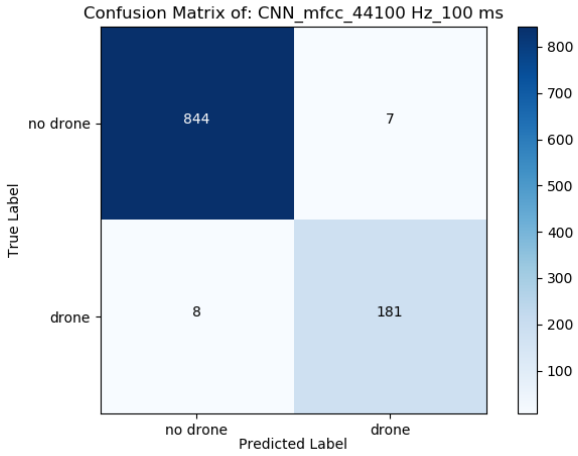

Figure 4.70. *Model's confusion matrix*

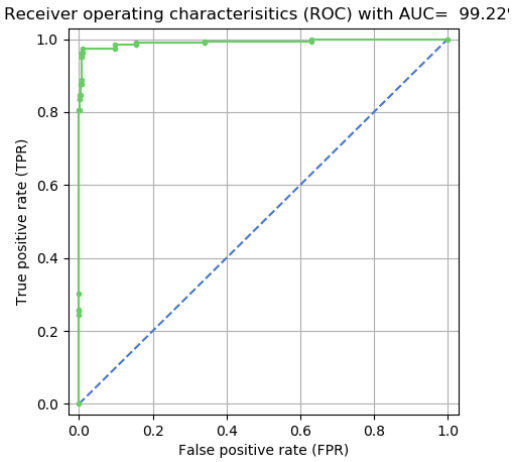

Figure 4.71. *Model's ROC curve with AUC= 99.22%*

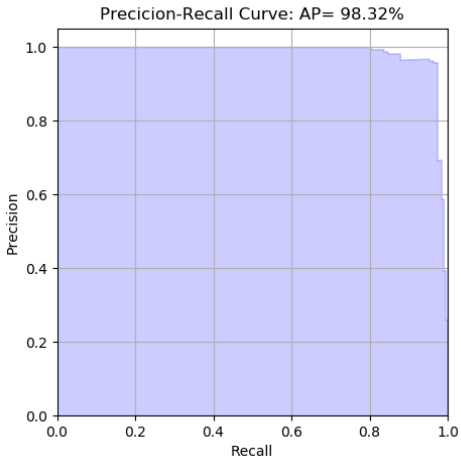

Figure 4.72. *Model's PR curve with average precision 98.32%*

CNN with filter banks, 44.1 KHz sampling rate, 100 ms frame length: the total time that is needed to train this model during 10 epochs is 366.6 minutes. The average time needed in each epoch is 2200 seconds. The time needed in each epoch of the ten is [2214 sec, 2198, 2198, 2207, 2201, 2207, 2206, 2187, 2189, and 2189] from epoch 1 to 10 respectively. The total number of parameters is 960266. The loss and accuracy in both train and validation sets are shown in Figure 4.73. The criterion that is chosen to save the best model's parameters during training is the validation accuracy. The best recorded validation accuracy in epoch (10) Table 4.23. The validation set consists 10 % of the train set. Let's see the performance of this model on test set (unseen data). The accuracy on test set is (0.9836), the average precision is (0.9855), and the AUC is (0.9935). The precision, recall, and f1-score for positive and negative classes are listed in Table 4.24.

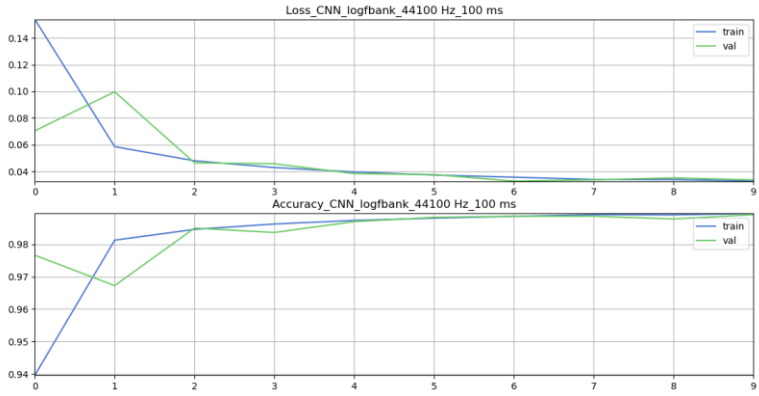

Figure 4.73. *Accuracy and loss during model's training*

Table 4.23. *Best recorded loss and accuracy during model's training*

	Loss	Accuracy
Train set	0.0330	0.9895
Validation set	0.0338	0.9891

The confusion matrix of this model is illustrated in Figure 4.74, the ROC of this model is illustrated in Figure 4.75, and the PR curve is illustrated in Figure 4.76.

Table 4.24. *Model's classification report*

Class	Precision	Recall	F1-score	Support
No drone	0.99	0.99	0.99	851
Drone	0.97	0.94	0.95	189

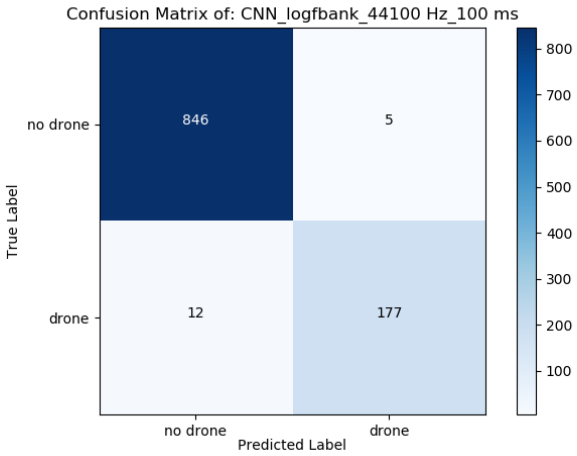

Figure 4.74. *Model's confusion matrix*

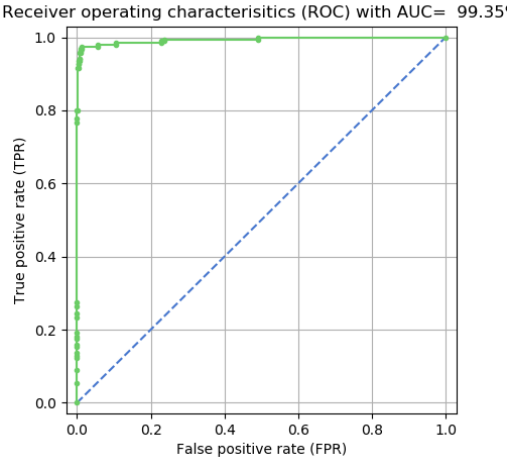

Figure 4.75. *Model's ROC curve with AUC= 99.35%*

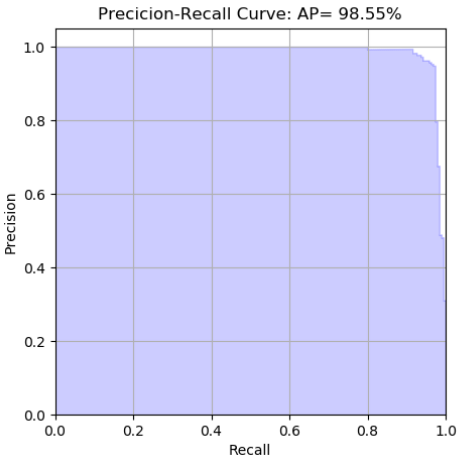

Figure 4.76. *Model's PR curve with average precision 99.55%*

5. CONCLUSION

In this thesis, the efficiency of deep learning algorithms in addressing the problem of acoustic drone detection was investigated. In this context, an empirical analysis of drones and environmental sounds, recorded by a set of audio sensors (microphones) using a multichannel sound card (TASCAM US1800), is performed. MFCCs and Mel-scaled filter banks are used for acoustic features and these features are adopted by the detection system to classify drones' sounds. In this thesis, the advanced deep learning algorithms, which are especially designed for object recognition, can be adapted to classify acoustic drones' sounds using visual frames taken from the running MFCCs and spectrograms of sounds.

In our experiments, we evaluated different sampling rates, acoustic features, and two deep neural networks algorithms (CNN and RNN) using our recorded and collected dataset. We assessed the classification performance for two different acoustic features extraction techniques MFCCs and Mel-scale filter banks, and the best classification assessment metrics were obtained using Mel-scale filter banks. Deep convolutional neural network models (CNN) showed better performance than the recurrent neural network (RNN) models, although RNN models are especially designed for speech recognition applications.

Expanding the training data with different types of drones, represents one of the main critical future works. In our setup, we did not take the records of drones from long distances that exceed 50 m. So using multiple microphones with beam forming and a signal processing technique for filtering in order to achieve directional signal transmission can be a future research works which can improve the detection range and performance of the proposed model.

REFERENCES

[1] B. Nassi, A. Shabati, R. Masuka and e. a. (2019), "SoK - Security and Privacy in the Age of Drones: Threats, Challenges, Solution Mechanisms, and Scientific Gaps," *arXiv preprint arXiv:1903.05155.*

[2] M. Z. Anwar, Z. Kaleem and A. J. (2019), "Machine learning inspired sound-based amateur drone detection for public safety applications," *IEEE Transactions on Vehicular Technology,* vol. 68, no. 3, pp. 2526-2534.

[3] B. Hudson, August 2018. [Online]. Available: https://www.washingtonpost.com/opinions/drone-attacks-are-essentially-terrorism-by-joystick/2018/08/05/f93ec18a-98d5-11e8-843b-36e177f3081c_story.html.

[4] A. V. Herrero and N. Casey, Augest 2018. [Online]. Available: https://www.nytimes.com/2018/08/04/world/americas/venezuelan-president-targeted-in-attack-attempt-minister-says.html.

[5] J. Desjardins, March 2018. [Online]. Available: https://www.businessinsider.com/amazon-and-ups-are-betting-big-on-drone-delivery-2018-3.

[6] A. G. (2017), Hands-On Machine Learning with Scikit-Learn and TensorFlow, O'Reilly Media, pp. 4-14.

[7] L. Deng and D. Y. (2014), "Deep learning: methods and applications," *Foundations and Trends in Signal Processing,* vol. 7, no. 3-4, pp. 197-387.

[8] Tascam. [Online]. Available: https://tascam.com/us/product/us-1800/feature.

[9] J. Salamon, C. Jacoby and J. B. (2014), "A Dataset and Taxonomy for Urban Sound Research," in *22nd ACM International Conference on Multimedia,* Orlando USA.

[10] M. Strauss, P. Mordel, V. Miguet and e. a. (2018), "DREGON: Dataset and Methods for UAV-Embedded Sound Source Localization," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS),* Madrid, Spain.

[11] BBC, February 2016. [Online]. Available: https://www.bbc.com/news/uk-35641453.

[12] D. Barrett, May 2015. [Online]. Available: https://www.telegraph.co.uk/news/uknews/crime/11613568/Burglars-use-drone-helicopters-to-identify-targe-homes.html.

[13] B. Nassi, A. Shamir and Y. E. (2019), "Xerox Day Vulnerability," *IEEE Transactions on Information Forensics and Security*, vol. 14, no. 2, pp. 415--430.

[14] M. Guri, B. Zadov and Y. E. (2017), "LED-it-GO: Leaking (a lot of) data from air-gapped computers via the (small) hard drive LED," in *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*, Cham.

[15] S. Birnbach, R. Baker and I. M. (2017), "Wi-Fly?: Detecting Privacy Invasion Attacks by Consumer Drones," *NDSS*.

[16] J. Kim, C. Park, Y. Ko and e. a. (2017), "Real-time UAV sound detection and analysis system," in *2017 IEEE Sensors Applications Symposium (SAS)*, Glassboro, NJ, USA.

[17] J. Mezei and A. M. (2016), "Drone sound detection by correlation," in *2016 IEEE 11th International Symposium on Applied Computational Intelligence and Informatics (SACI)*, Timisoara, Romania.

[18] V. Mirelli, S. Tenney, Y. Bengio and e. a. (2009), "Statistical machine learning algorithms for target classification from acoustic signature," in *Proc. MSS Battlespace Acoust. Magn. Sensors*.

[19] E. E. Case, A. M. Zelnio and B. D. R. (2008), "Low-cost acoustic array for small UAV detection and tracking," in *2008 IEEE National Aerospace and Electronics Conference*, Dayton, OH, USA.

[20] L. V. Santana, A. S. Brando, M. Sarcinelli-Filho and e. a. (2014), "A trajectory tracking and 3d positioning controller for the ar. drone quadrotor," in *2014 international conference on unmanned aircraft systems (ICUAS)*, Orlando, FL,

USA.

[21] E. Unlu, E. Zenou and N. R. (2018), "Using shape descriptors for UAV detection," *Electronic Imaging*, vol. 2018, no. 9, pp. 1-5.

[22] M. Saqib, S. D. Khan, N. Sharma and e. a. (2017), "A study on detecting drones using deep convolutional neural networks," in *2017 14th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, Lecce, Italy.

[23] C. Aker and S. K. (2017), "Using deep networks for drone detection," in *2017 14th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, Lecce, Italy.

[24] T. M. (2017), "Robust drone detection for day/night counter-UAV with static VIS and SWIR cameras," in *Ground/Air Multisensor Interoperability, Integration, and Networking for Persistent ISR VIII*.

[25] H. Liu, Z. Wei, Y. Chen and e. a. (2017), "Drone detection based on an audio-assisted camera array," in *2017 IEEE Third International Conference on Multimedia Big Data (BigMM)*, Laguna Hills, CA, USA.

[26] J. Busset, F. Perrodin, P. Wellig and e. a. (2015), "Detection and tracking of drones using advanced acoustic cameras," in *Unmanned/Unattended Sensors and Sensor Networks XI; and Advanced Free-Space Optical Communication Techniques and Applications*.

[27] M. U. d. Haag, C. G. Bartone and M. S. B. (2016), "Flight-test evaluation of small form-factor LiDAR and radar sensors for sUAS detect-and-avoid applications," in *2016 IEEE/AIAA 35th Digital Avionics Systems Conference (DASC)*, Sacramento, CA, USA.

[28] W. Shi, G. Arabadjis, B. Bishop and e. a. (2011), "Detecting, tracking, and identifying airborne threats with netted sensor fence}," in *Sensor Fusion-Foundation and Applications*.

[29] A. Hommes, A. Shoykhetbrod, D. Noetel and e. a. (2016), "Detection of acoustic, electro-optical and RADAR signatures of small unmanned aerial vehicles," in *Target and Background Signatures II, International Society for Optics and Photonics*.

[30] S. Chu, S. Narayanan, C.-c. J. Kuo and e. a. (2006), "Where am I? Scene recognition for mobile robots using audio features," in *2006 IEEE International Conference on Multimedia and Expo*, Toronto, Ont., Canada.

[31] B. Schilit, N. Adams and R. W. (1994), "Context-aware computing applications," in *1994 First Workshop on Mobile Computing Systems and Applications*, Santa Cruz, California, USA.

[32] Y. Xu, W. J. Li and K. K. L. (2008), in *Intelligent Wearable Interfaces*, WIlEY.

[33] O. Abdel-Hamid, L. Deng and D. Y. (2013), "Exploring Convolutional Neural Network Structures and Optimization Techniques for Speech Recognition," in *Interspeech*.

[34] K. J. P. (2015), "Environmental sound classification with convolutional neural networks," in *2015 IEEE 25th International Workshop on Machine Learning for Signal Processing (MLSP)*, Boston, MA, USA.

[35] V. Boddapat, A. Petefi, J. Rasmusson and e. a. (2017), "Classifying environmental sounds using image recognition networks," *Procedia computer science*, vol. 112, pp. 2048-2056.

[36] L. Hauzenberger and E. H. O. (2015), "Drone detection using audio analysis".

[37] I. Goodfellow, Y. Bengio and A. C. (2016), Deep Learning, MIT press.

[38] T. Heittola, E. Çakır and T. V. (2018), "The machine learning approach for analysis of sound scenes and events," in *Computational Analysis of Sound Scenes and Events*, Cham.

[39] B. Gold, N. Morgan and D. E. (2011), Speech and audio signal processing: processing and perception of speech and music, 2nd ed., John Wiley & Sons.

[40] A. V. Oppenheim and R. W. S. (2014), Discrete-time signal processing, Pearson Education.

[41] H. F. (2016), "Speech Processing for Machine Learning: Filter banks, Mel-Frequency Cepstral Coefficients (MFCCs) and What's In-Between," [Online]. Available: https://haythamfayek.com/2016/04/21/speech-processing-for-machine-learning.html. [Accessed 08 October 2019].

[42] A. R. M. (2014), "Deep neural network acoustic models for ASR".

[43] K. He, X. Zhang, S. Ren and e. a. (2016), "Deep Residual Learning for Image Recognition," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[44] K. Simonyan and A. Z. (2014), "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556.*

[45] Y. Wu, H. Mao and Z. Y. (2018), "Audio classification using attention-augmented convolutional neural network," *Knowledge-Based Systems*, vol. 161, pp. 90-100.

[46] M. Valenti, A. Diment, G. Parascandolo and e. a. (2016), "DCASE 2016 acoustic scene classification using convolutional neural networks," in *Detection and Classification of Acoustic Scenes and Events*, Budapest, Hungary.

[47] D. P. Kingma and J. L. B. (2014), "Adam: A Method for Stochastic Optimization," *arXiv preprint arXiv:1412.6980.*

[48] S. Jeon, J. W. Shin, Y. J. Lee and e. a. (2017), "Empirical study of drone sound detection in real-life environment with deep neural networks," in *2017 25th European Signal Processing Conference (EUSIPCO)*, Kos, Greece.

[49] H. Sak, A. Senior and F. B. (2014), "Long short-term memory recurrent neural network architectures for large scale acoustic modeling," in *Fifteenth annual conference of the international speech communication association*, Singapore.

[50] P. S. Foundation. [Online]. Available: https://www.python.org/about/.

[51] A. Inc, 2019. [Online]. Available: https://www.anaconda.com/what-is-anaconda/.

[52] TensorFlow. [Online]. Available: https://www.tensorflow.org.

[53] Keras. [Online]. Available: https://keras.io/.

[54] Scikit-Learn. [Online]. Available: https://scikit-learn.org/stable/#.

[55] S. M. T. GmbH, 2019. [Online]. Available: https://new.steinberg.net/cubase/le/.

[56] Audio-Technica. [Online]. Available: https://eu.audio-technica.com/AT2031.

[57] A. T. (2018), "Classification assessment methods," *Applied Computing and Informatics.*

[58] K. H. Zou, A. J. O. and L. M. (2007), "Receiver-operating characteristic analysis for evaluating diagnostic tests and predictive models," *Circulation*, vol. 115, no. 5, pp. 654-657.