UAV DETECTION SYSTEM WITH MULTIPLE ACOUSTIC NODES USING MACHINE LEARNING MODELS

by

Bowon Yang

A Thesis

Submitted to the Faculty of Purdue University

In Partial Fulfillment of the Requirements for the Degree of

Master of Science

Department of Computer and Information Technology

West Lafayette, Indiana

May 2019

THE PURDUE UNIVERSITY GRADUATE SCHOOL
STATEMENT OF COMMITTEE APPROVAL

Prof. Eric T. Matson, Chair

Department of Computer and Information Technology

Prof. J. Eric Dietz

Department of Computer and Information Technology

Prof. Anthony H. Smith

Department of Computer and Information Technology

Prof. John C. Gallagher

Department of Computer Science, Wright State University

Approved by:

Dr. Eric T. Matson

Head of the Graduate Program

ACKNOWLEDGMENTS

I wish to gratefully acknowledge every individual who made this thesis possible with warm support and amazing inspiration. First of all, I owe my deepest gratitude to my advisor Prof. Eric T. Matson for his support and belief in me. He did not only support my research but also let me see the bigger world and showed me what a hard work is himself. I express my deepest respect for him. I would like to say thank you to Prof. Anthony H. Smith, Prof. J. Eric Dietz, Prof. John C. Gallagher for valuable advice, allowing me to complete my thesis. Their insight and knowledge on the topic had a significant impact on developing this thesis.

In addition to my committee members, I wish to show gratitude to M2M lab members for helping me with field experiments under the sizzling sun in August. I sincerely appreciate Eunsuh Lee for sparing her time for every one of my experiment. I would never have finished my research without her. I would like to acknowledge Dr. Yongho Kim and Dr. Seongha Park for the continuous advice and encouragement. Also, I want to say thank you to Hyeonae Jang for staying next to me as a great friend and roommate through the hard times. Finally, I give my deepest love to my family for always being there and for trusting me.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

UAV	Unmanned Aerial Vehicle
CUAS	Counter Unmanned Aerial Vehicle
CNN	Convolutional Neural Network
SVM	Support Vector Machine
ZCR	Zero Crossing Rate
MFCC	Mel-frequency Cepstrum Coefficient
STFT	Short-time Fourier Transform

ABSTRACT

Author: Yang, Bowon. M.S.
Institution: Purdue University
Degree Received: May 2019
Title: UAV Detection System with Multiple Acoustic Nodes Using Machine Learning Models
Major Professor: Eric T. Matson

This paper introduced a near real-time acoustic unmanned aerial vehicle detection system with multiple listening nodes using machine learning models. An audio dataset was collected in person by recording the sound of an unmanned aerial vehicle flying around as well as the sound of background noises. After the data collection phase, support vector machines and convolutional neural networks were built with two features, Mel-frequency cepstral coefficients and short-time Fourier transform. Considering the near real-time environment, the features were calculated after cutting the audio stream into chunks of two, one or half seconds. There are four combinations of features and models as well as three versions per combination based on the chunk size, returning twelve models in total. To train support vector machines, the exhaustive search method was used to find the best parameter while convolutional neural networks were built by selecting the parameters manually. Four node configurations were devised to find the best way to place six listening nodes. Twelve models were run for each configuration, generating color maps to show the paths of the unmanned aerial vehicle flying along the nodes. The model of short-time Fourier transform and support vector machines showed the path most clearly with the least false negatives with 2-second chunk size. Among the four configurations, the configuration for experiment 3 showed the best results in terms of the distance of detection results on the color maps. Web-based monitoring dashboards were provided to enable users to monitor detection results.

CHAPTER 1. INTRODUCTION

Unmanned Aerial Vehicles (UAVs), known as drones, have become increasingly popular for the last two years. Around 1.3 million UAVs were sold in 2015 for recreational purposes and 3 million in 2017, more than doubling the number [1]. Among multiple models of different manufacturers, DJI Phantom, made by DJI in China, accounts for 70% of the market share [2].

Although most of the UAVs are used for recreation, they pose a threat to security because some people can use them for malicious purposes. One of the incidents that raised the alarm to the world about its potential danger was in January 2015 when a UAV crashed in the White House lawn overnight, and nobody noticed until the owner reported it himself. It happened by accident, as he lost control of the UAV and it fell by the White House, but this could have been a serious case otherwise.

This can have bad results when people use UAVs for malicious purposes. For example, drug dealers have started to fly UAVs to smuggle drugs into jail. In the United Kingdom, a gang was sentenced for smuggling drugs and prohibited items into prisons at least 49 times by using UAVs [3]. They were caught by accident by cameras filming nature. Besides, terrorists can use UAVs to attack. Islamic State fighters are already using UAVs as weapons. They modified consumer UAVs to carry bombs and dropped them in Iraqi troops [4].

1.1 Significance

It is significant that the UAV was not detected even with the most advanced security technologies in the White House. The surveillance systems at the White House could not detect the UAV because of its small size. Majority of the air surveillance systems that detect flying objects use radars that are optimized for fast and massive objects like missiles, but consumer UAVs are small and slow to be recognized by those radars [5].

UAVs are accessible in the market, and some hobbyists build their own UAVs as they are open source hardware. They are easy to tune according to users' needs. Along

with the fact that UAVs are small and not easily noticeable, the chance is higher that this could lead to serious incidents more frequently in the near future. Given that, an affordable UAV detection system that can be installed at different places is in high demand.

1.2 Problem

The problem being addressed in this research is the lack of realistic, usable and accessible detection systems to detect class 1 UAVs and the necessity of these systems with limited computing resources in the public space.

1.3 Research Question

Can an acoustic UAV detection system that consists of multiple listening nodes detect the audio signal of a class 1 UAV in near real-time using a machine learning model and determine the path of the UAV?

1.4 Scope

The scope of this research includes the development of audio-based UAV detection system in near real time using multiple computing nodes. There is one control center that the nodes keep reporting the probability of whether there is a UAV around to. Each node continually listens to the surroundings with a microphone and runs detection module comprised of feature extraction and a pre-trained machine learning model. Features of audio signals recorded from the microphones are calculated and put into the detection module as an input as the node is listening.

To achieve this, there are three deliverables to be proposed, first of which is a machine learning model that can recognize audio signals of a UAV, along with feature extraction methods and a dataset. Secondly, the design of a computing node that is low-cost and capable of detecting UAVs with acoustic signals is proposed. Lastly, an optimal configuration of nodes in the field is proposed.

1.5 Definitions

- A *UAV* is an Unmanned Aerial Vehicle. In this paper, a class 1 UAV, AR Drone 2.0, is used.
- A *UAV detection system* is a system that detects UAVs using sensors and algorithms
- A *node* is a computing unit deployed in the UAV detection system. In this paper, it is refined to a Raspberry Pi.
- A *target* is the goal of the UAV in the context of this research.
- A *near real-time* detection system indicates a system that can send detection results to the control center in a second.

1.6 Assumptions

The assumptions for this study include:

- There is only one UAV in flight at one time.
- The UAV does not have any payloads.

1.7 Limitations

The limitations of this research include:

- The hardware limitations of computing nodes can put a limit on detection results.
- The distance between the control center and computing nodes can be limited by the network hardware used in the experiments.
- Detection results can be affected by the ambient environments and background noise from birds, cars and so on.

- The hardware performance of the machine used for training the machine learning model can limit the quality of the model.

1.8 Delimitations

The delimitations for this study include:

- The study aims at developing a UAV detection system, and tracking part of the counter UAV system is not considered.
- Only one type of UAVs, AR Drone 2.0, is used for training the model and testing it.
- UAV audio signals are recorded by flying a UAV from every angle, and the Doppler effect is not considered.

CHAPTER 2. REVIEW OF RELEVANT LITERATURE

This chapter provides a review of relevant literature in three sections. The first section introduces counter UAV technologies. Next, different strategies to detect UAVs are explored. The last section reviews general audio detection methodologies including feature extraction and machine learning algorithms, not limited to the sound of UAVs.

2.1 Counter Unmanned Aerial Systems

As concerns of threats by UAVs arise, anti-UAV technologies have been rapidly growing. Academia and industry have started to build systems to detect UAVs and take measures, which are called Counter Unmanned Aerial Systems (CUAS). The missions of a CUAS can be illustrated as a sequence of four tasks as shown in figure 2.1. The sequence starts with initial detection of a UAV and then leads to the analysis step, where the system categorizes the type of the UAV and analyzes the threat using visual analysis or sound signatures. If the UAV is identified as a threat, it is mitigated using countermeasures, depending on what kind of strategies the system is using. At the last forensics step, the UAV caught by the countermeasures is analyzed, and further investigation is performed.

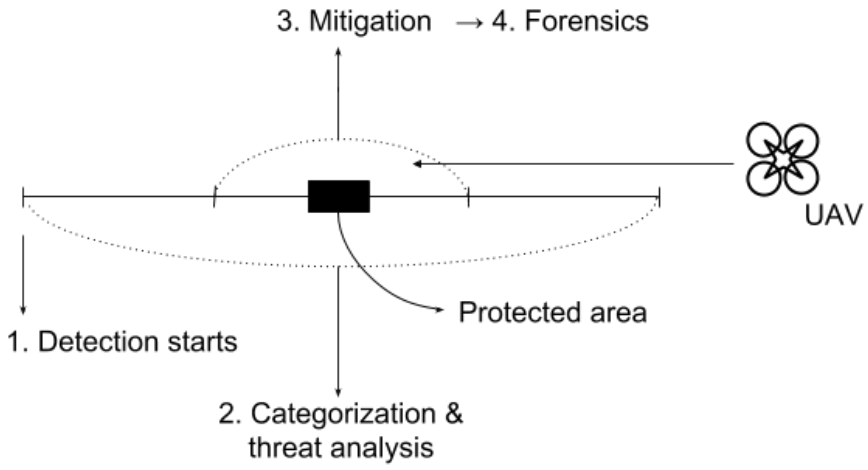


Figure 2.1. Sequences of CUAS

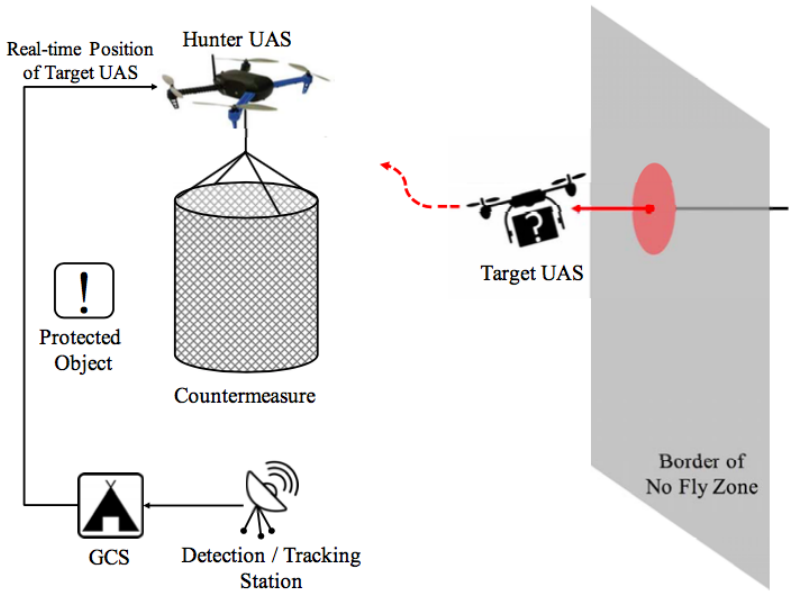


Figure 2.2. The overview of the CUAS by Purdue University [6]

Researchers at Purdue University realized an autonomous CUAS that intercepts an unauthorized UAV, as a result of a challenge put forward for researchers after the White House incident [6]. The system detects, tracks, and takes down class 1 UAVs using a hunter UAV attached with a net. As in figure 2.2, the radar-based station starts detecting and tracking the target UAV when it gets into the restricted zone. The hunter UAV keeps updated with the position of the target UAV from the station and filters noises to make it more precise. Then the hunter approaches the target UAV with the attached net to catch it. The net is cylindrical to maximize the chance of interception. The system is well-designed and notable in the sense that it is fully autonomous and minimizes possible threats that come from the payload of the target UAV using the net.

Inspired by the implementation above, the same research group developed a vision-based CUAS, instead of using radar [7]. The hunter UAV autonomously detects and tracks the target UAV using a monocular camera attached to it. The system is on surveillance mode by default, where the hunter UAV keeps monitoring and reports the detected UAV to the authorities. If the authorities decide to engage, they activate interception mode, and then the human pilot operates the hunter UAV to cause an air-to-air

(a) DroneGun by DroneShield [10] (b) SkyWall projectile by OpenWorks [11]

Figure 2.3. Rifle-shaped countermeasures by startups

collision with the target UAV. The hunter UAV can carry a net or some explosive material, adding counter measures. According to the researchers [7], the success rate of tracking the target UAV is 87.23%, and that of interceptions with a net attached is 81.3% on average. This system requires a human operator, but it becomes a good indicator for further research, showing how a fully autonomous CUAS can be developed into.

From industry, traditional aerospace companies have started to build CUAS solutions. Existing air defense systems from these companies mainly focus on huge and fast flying objects such as missiles and aircraft. Now they are making air defense systems optimized for smaller UAVs. Airbus Defense and Space released a CUAS against small UAVs in 2015 to secure military barracks, airports or nuclear plants using radars, infrared cameras and direction finder [8]. It can detect UAVs that are up to 10 kilometers away, identify threats and jam the connection between the UAV and the operator. Also, in 2017, Airbus added portable countermeasures to their CUAS product [9].

Also, drone security startups have emerged, releasing various solutions to the UAV threats. DroneShield [10] is a startup that specializes in UAV monitoring systems and counter measure devices. They recently announced a portable UAV countermeasure called DroneGun as in figure 2.3 (a), which jams the connection with the operator of the unauthorized UAV, ceasing the video transmission immediately. It can have the target UAV land safely without any damage to enable further investigation. DroneShield also provides detection solutions integrating acoustic, optical, radar, and radio frequency (RF)

sensors. SkyWall by an English company OpenWorks is a type of gun that shoots a netted parachute to capture the target UAV without electronic countermeasure [11].

Commercial solutions by startups including DroneShield and OpenWorks are through the use of portable devices in a rifle shape that does not require installation. They are easy to use, and no technical training is necessary. However, the fact that the operator should be out in the field near the UAV is a limitation compared to the CUAS by Purdue University that can perform counter measures remotely. Solutions by big companies are highly accurate and used by the military, but the cost is too high for government facilities with a lower budget.

2.2 Detection of UAVs

Even though counter measures are the ultimate goal of a CUAS, an effective detection phase is crucial to perform the measures successfully. In this section, the literature on UAV detection methodologies is covered, divided by different sensors including radar, lidar, vision, and acoustics.

2.2.1 Radar detection

Radar is an object detection system that sends radio waves and detects the echo returned from the target [12]. The range, angle, or speed of objects can be determined using radar sensors. Radar sensors are reliable in the sense that they are not affected by weather conditions and have a long range of detection [12]. Traditionally, radar has been mainly used for air defense systems to track aircraft or missiles. These radar-based detection systems are optimized for bigger objects and highly expensive [13]. Detection of class 1 UAVs is challenging because they have a very small radar cross section (RCS) as well as lower speed and lower altitude [14].

However, research shows the possibility of a low-cost UAV detection and tracking system by using a synthetic aperture radar (SAR) on top of an unmanned ground vehicle [15]. The system uses frequency modulated continuous wave (FMCW) and

operates at a center frequency of 2.4274 GHz with a bandwidth of 260 MHz. The range of detection is up to 50 meters.

2.2.2 Laser Sensor

Lidar (Light Detection And Ranging) sensors transmit laser beams and analyze the beams reflected from the target [16]. Lidar is frequently used in underwater detection [16] and forestry monitoring [17]. Although it has higher resolution and accuracy, the cost is high, and the results are vulnerable to weather conditions, which makes it unpractical for UAV detection [7]. However, a group of researchers implemented a UAV detection system using an affordable lidar sensor mounted on a UAV [18]. The system was deployed indoors and could detect a UAV as well as the velocity and speed.

2.2.3 Computer Vision

Object detection is one of the major tasks in computer vision. Numerous researchers have published papers on object detection using optical sensors [19]. Approaches to this problem include appearance-based methods and motion-based methods [20]. Appearance-based methods rely on analyzing individual frames while motion-based methods use motion information that comes from differences between frames. Machine learning techniques are involved in appearance-based methods, where deep learning led breakthroughs improving the accuracy significantly even in a noisy environment [21]. Motion-based methods are through techniques such as background subtraction or frame differencing that calculates differences between two frames.

A. Rozantsev [20] presents an integration of the two methods to detect aircraft and UAVs using a camera on board a UAV. Figure 2.4 shows the flow of detection in [20]. The motion information is extracted to detect flying objects by motion compensation. Each frame is processed to decide if the object is a target object using convolutional neural networks. The authors developed a UAV image dataset combining real and synthetic data

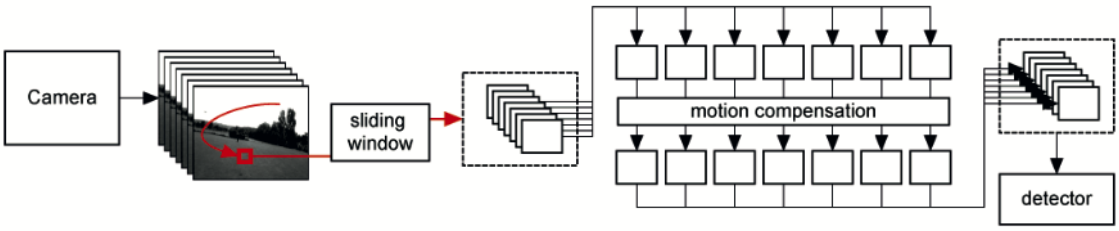


Figure 2.4. Detecting flying objects integrating appearance-based and motion-based methods [20]

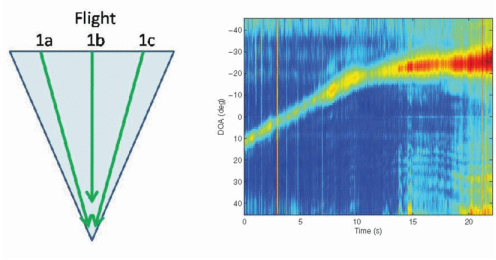
as well as the adaptation technique that prevents overfitting. The average precision of their system is 0.732 according to the author.

Using computer vision to detect UAVs is available with cheap optical sensors compared to radar or lidar, and it inherently enables accurate tracking. It can also be beneficial when detecting UAVs that do not have RF transmission [14]. However, it can be challenging to detect UAVs when the weather condition is bad, or there are many noises. Also, it can consume a lot of resources depending on the algorithm, which makes it hard to get detection results immediately with limited resources.

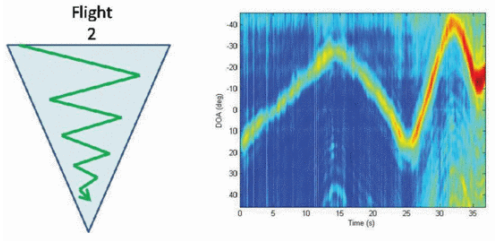
2.2.4 Acoustic Sensors

Using acoustic sensors is another cheap and easily deployable way to detect UAVs. Audio signal processing can be more economical than vision-based methods because it requires less computing resources than image processing in terms of data size, and the sensors are cheaper than cameras. Also, a wider range of detection is possible, meaning that UAVs can be detected from greater distances and the sound signals can be recognized from any angle.

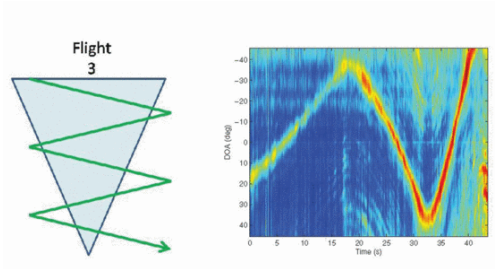
A paper published in 2008 [22] demonstrates the possibility of a cheap UAV detection system using acoustic nodes. The system makes use of a microphone array and beamforming, which allows for a determination of position and direction of the target UAV. The array consists of 24 microphones aligned in a straight line with the sensors 10



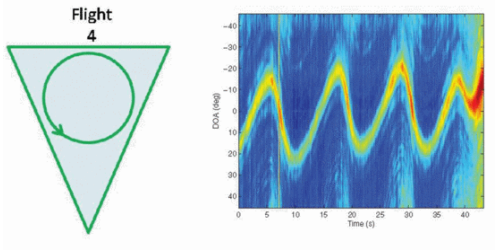
(a) Path 1: straight path towards the array.



(b) Path 2: Zig-zag in the field of view.



(c) Path 3: Zig-zag that passed out of the field of view.



(d) Path 4: three circles in front of the array.

Figure 2.5. Four trajectories of the UAV and the waterfall plots from [22]

inches apart from each sensor. They flew a consumer UAV following four trajectories as in the left figures from figure 2.5. Although this implementation still requires initial calibration to get the position of the array, this shows an inexpensive way to detect UAVs using well-known and straightforward techniques. Also, a report by army research professionals was published on the topic of class 1 UAV detection using tetrahedral microphone arrays with beamforming.

Recently, as machine learning has been demonstrated effective in the audio signals domain, researchers used different learning algorithms to develop detection modules. An implementation from [23] detects UAVs in real-time using deep learning models, including recurrent neural networks and convolutional neural networks, as well as a Gaussian mixture model as a baseline algorithm. The study focuses on event sound detection in a noisy urban area using binary classification. For features, Mel-frequency cepstral coefficients (MFCC) which are wildly used in sound detection field are chosen [24]. Also, sound augmentation is presented to augment UAV sound into environmental sound to overcome the shortage of datasets in prohibited areas. The authors recorded the sound produced by a few popular commercial hobby UAVs, and then augmented this data with varius environmental sound data to remedy the scarcity of UAV sound data in diverse environments.

2.3 UAV Sound Detection

While the problem of UAV audio detection is novel, a lot of research has been done on classifying audio signals in different domains, such as speech recognition [25, 26], music classification [27], sound event detection [28, 29], environment sound classification [30, 31] and so on. Among them, UAV sound classification is a partial task of sound event detection or environmental sound classification where there is only one class to classify.

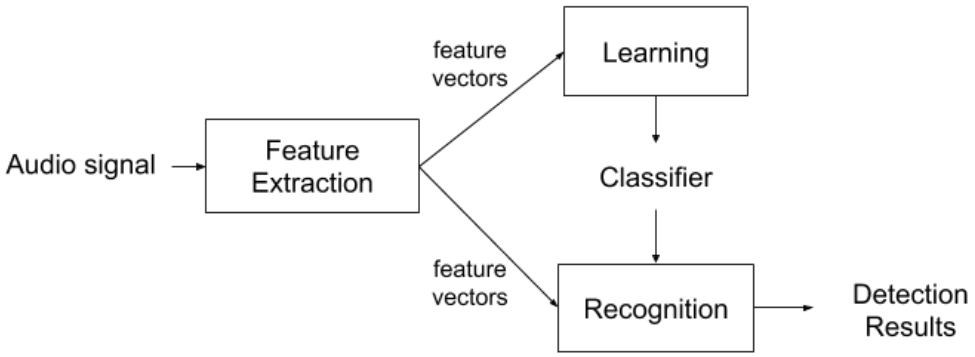


Figure 2.6. Audio event classification framework

Table 2.1. Comparison of public datasets for environmental sound classification and sound event detection

Name	Year	Number of classes	Baseline accuracy
UrbanSound8K	2014	10 (18 hours)	MFCC-SVM: 69%
ESC-50	2015	50 (2.78 hours)	MFCC-SVM: 39.6%
ESC-10	2015	10 (0.56 hours)	MFCC-SVM: 67.5%
TUT Acoustic Scenes	2016	15	MFCC-GMM: 72.5%

2.3.1 Dataset

Classification of audio signals using machine learning is a series of tasks that include feature extraction, learning and recognition as illustrated in figure 2.6. To recognize audio effectively, a proper dataset for the task is the requirement. There are several public datasets for environmental sound classification or sound event detection as in 2.1. The authors of these datasets chose classes of sounds and collected audio either by recording themselves or curating open data. Mel-frequency cepstral coefficients (MFCC) are commonly used for baseline models, and the types of models vary from support vector machine (SVM) to Gaussian mixture models (GMM).

UrbanSound8K [32] is a dataset released in 2014. Focusing on urban sounds, the authors selected 10 classes based on frequency and downloaded audio files from a free online sound storage after filtering and annotating the dataset instead of recording in person. The authors showed the results of baseline models with MFCC, and SVM showed the highest accuracy among them. ESC-50 [33] is another public dataset with 50 classes of animal sounds, natural sounds, human sounds, interior sounds, and exterior urban sounds. Each file lasts 5 seconds, and each class has 40 clips. The authors also published ESC-10 after selecting 10 classes from ESC-50. Expectedly, 10 classes show better accuracy than 50 classes.

For sound event detection, an annotated dataset is released every year for the Detection and Classification of Acoustic Scenes and Events (DCASE) competition. TUT Acoustic Scenes [34] is a dataset published for DCASE 2016 challenge. Researchers brought a recording device in the public places and record for 30 seconds. Unlike

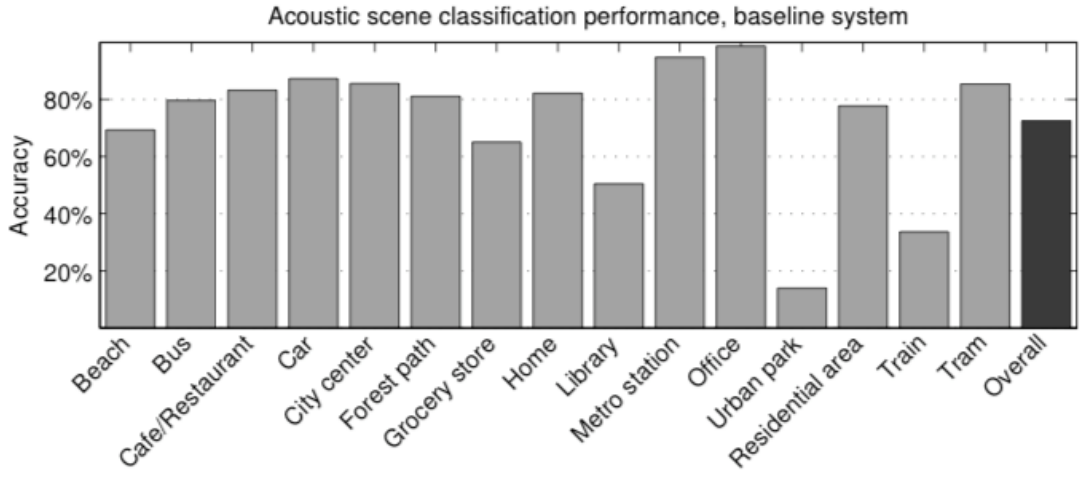


Figure 2.7. Baseline accuracy of TUT Acoustic Scenes 2016

UrbanSound8K and ESC datasets, TUT dataset has 15 locations and multiple classes within each location. For example, the data for residential areas has the classes of a car passing by, children shouting, people speaking and so on. Each audio file is around one minute long. Multiple classes exist in each file with overlaps, helping to build a polyphonic sound event detection system that can detect multiple events occurring at the same time. Figure 2.7 shows the performance of GMM on each location. The overall accuracy is 72.5%.

2.3.2 Feature Extraction

Figure 2.8 summarizes the process of feature extraction in audio signal processing. The first step is preprocessing, where raw audio values are filtered, resampled, or normalized. Next, the preprocessed signals are segmented into analysis frames on a short period like 20 milliseconds. Then the analysis frames are shifted, generating overlaps with the frames next to them. After that, using a windowing function, each frame is smoothed with abrupt changes removed [35].

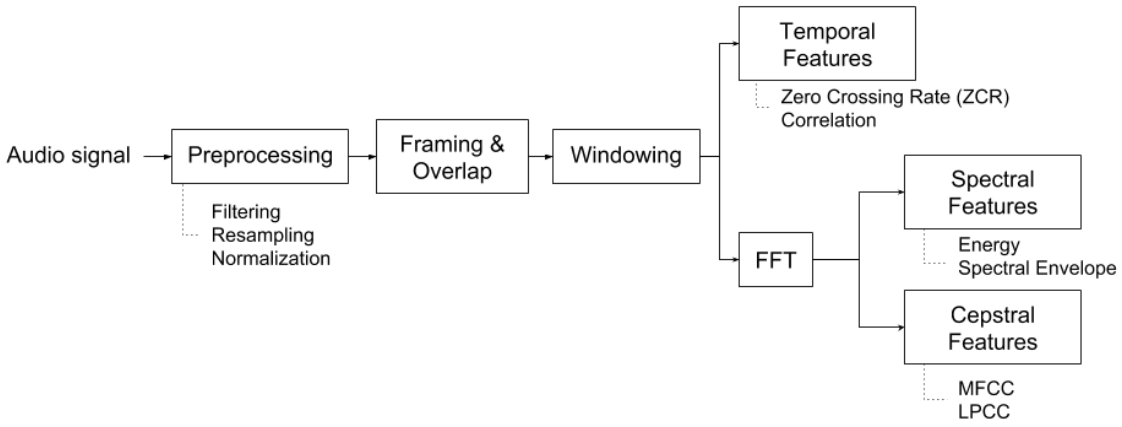


Figure 2.8. Feature extraction flow

Depending on the choice for features, a feature set is computed from the sequence of frames. Features can be classified into three groups [31], first of which is a temporal feature, where the values are still in the time domain. The other two features, spectral and cepstral, require a transformation that converts data from the time domain into the frequency domain using fast Fourier transform (FFT). Transformation makes the data less redundant, and signals represented in the frequency domain are also used as features [36]. Spectral and cepstral features can be extracted by applying further transformations. Multiple features can be combined into one vector. In [28], the feature set is a 13-dimensional vector, combining four different kinds of features. Also, principal component analysis (PCA) can reduce the dimensionality of feature vectors [37].

2.3.3 Classification and Detection

According to [35], machine learning tasks in sound recognition have two types based on whether the output contains temporal information. Figure 2.9 describes input and output characteristics for three types of analysis systems. Sound event detection systems return the classes of target items and the time of the detection as in the right diagram in figure 2.9. On the other hand, classification systems only output a class of a whole chunk. The left diagram in figure 2.9 shows single class classification, and the

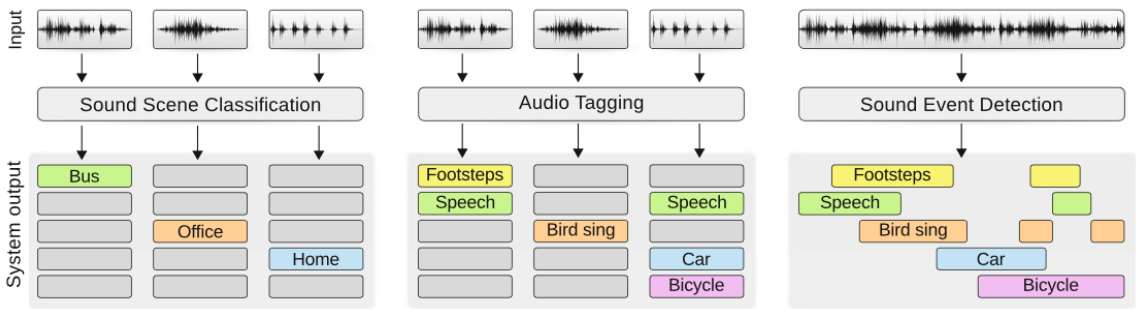


Figure 2.9. System input and output characteristics for three analysis systems: sound scene classification, audio tagging, and sound event detection [35]

diagram in the middle describes multi-label classification problems. Classifiers can be trained by supervised learning algorithms when annotations are available. However, it is more challenging to make annotations for a sound event detection system because it is hard to find a clear boundary of the start or the end of an event. For that reason, unsupervised learning methods are being actively studied recently [38].

Classification based on supervised learning is the main focus of this research as there are two classes of whether or not the current chunk is UAV sound with annotations. Also, this research can be seen as the combination of classification and detection because the classes do not overlap and the results are in time series. The term, detection, is interchangeably used as classification in further context.

2.4 Summary

This chapter provided a review of the literature relevant to UAV detection. Starting from the concept of the CUAS, it reviewed the previous works on the UAV detection using different kinds of sensors as well as general audio detection methodologies.

CHAPTER 3. FRAMEWORK AND METHODOLOGY

The goal of this research is to build a UAV detection system using multiple listening nodes and machine learning models. In this chapter, the overview of the proposed system is introduced along with the hardware and software components, followed by the research flow. In this chapter, the research flow is introduced, followed by the system overview along with hardware and software components. Then, the next sections describe methodologies on classification including data collection, features, and algorithms. Finally, the rest of this chapter covers the evaluation metrics and the field testing setups.

3.1 System Overview

Figure 3.1 describes the overall system. Nodes are placed on a circular formation surrounding the protected area. A control center within the network reach interacts with nodes receiving detection results. Each node runs a process to keep listening using a

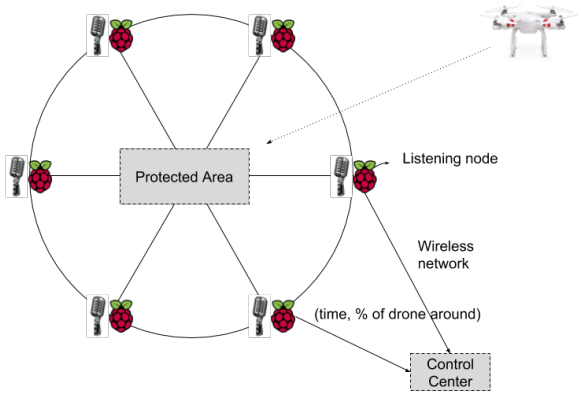
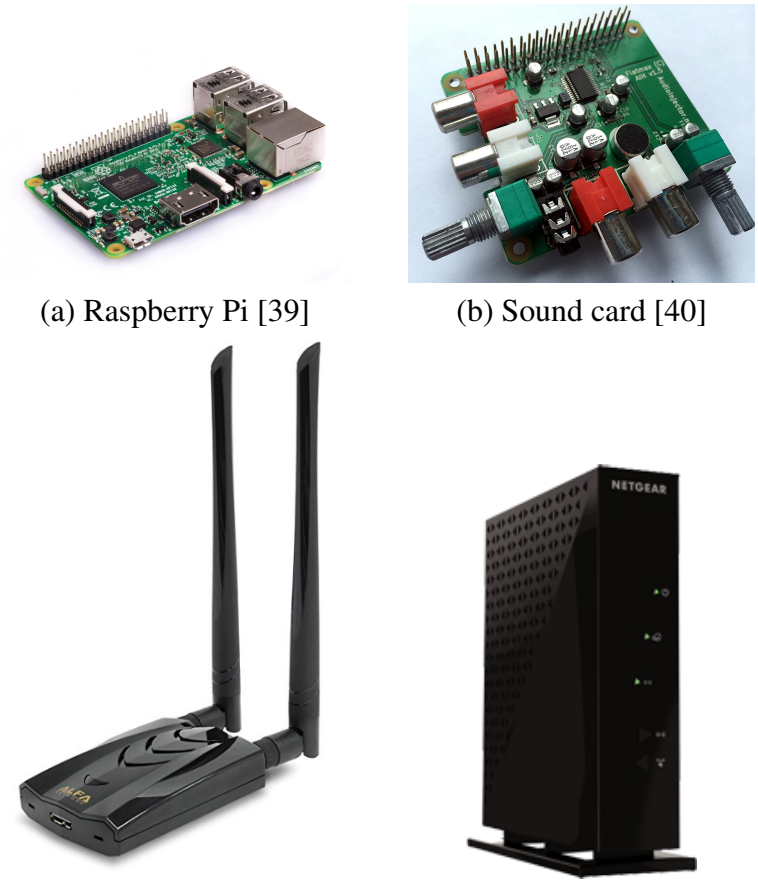


Figure 3.1. Overview of the system hardware



(a) Raspberry Pi [39] (b) Sound card [40]

(c) Wi-Fi adapter [41] (d) Netgear wireless router [42]

Figure 3.2. Core hardware needed for the proposed system

microphone and a sound card installed on a node. At the same time, another process runs the detection module per frame using a machine learning model implanted in the node.

3.1.1 Hardware

Figure 3.3 shows how one node is set up. Each node consists of a Raspberry Pi with a sound card with a built-in microphone, a battery, an inverter, and a power cable for the Raspberry Pi as in figure 3.3. It is put on a chair so that the node can be 0.7m above the ground. Details for the major components for the node are below:

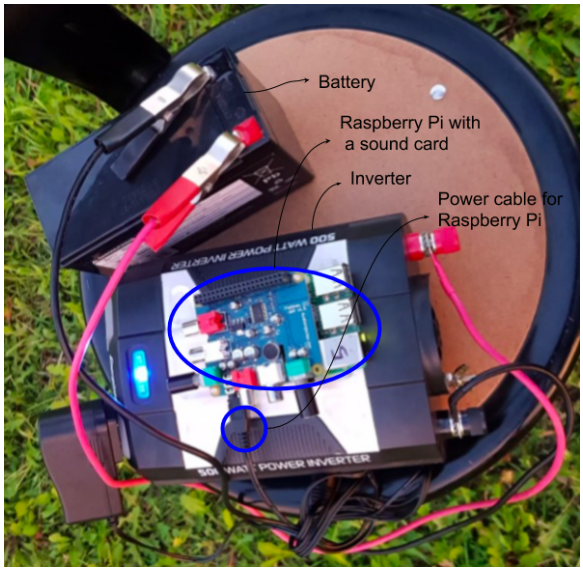


Figure 3.3. Node setup

- *Node*: Raspberry Pi 3 Model B is used as a computing node (figure 3.2 (a)) [39]. Six of them will be deployed, making a cluster of listening nodes.
- *Sound card*: A stereo sound card from Audio Injector (figure 3.2 (b)) [40] is used, which can be stacked on a Raspberry Pi. The sample rate is 96 kHz, and the sample size is 32 bits. The sound card is based on ALSA driver [43].
- *Wireless network*: Each Raspberry Pi with a wireless USB Wi-Fi adapter (figure 3.2 (c)) is connected to a Netgear router (figure 3.2 (d)) [41], allowing long-range wireless network between listening nodes and the control center.
- *Hardware for learning and server*: MacBook Air 2015 edition with 1.6 GHz Intel Core i5 for CPU and 8 GB 1600 MHz DDR3 memory, is used for building machine learning models. It is also used as a server to which each node sends data.

3.1.2 Software

Figure 3.4 shows the flow of the detection software. Below are the pieces of software utilized to implement the system:

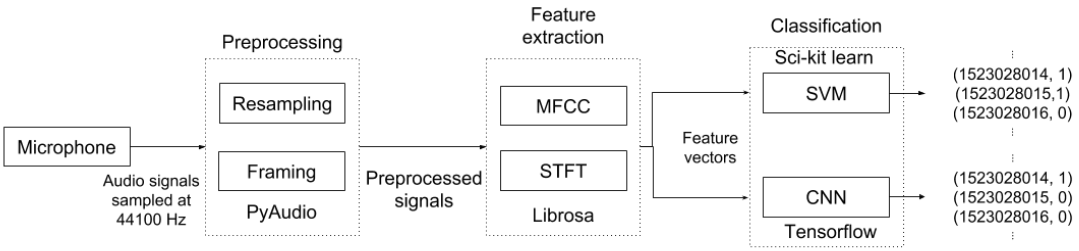
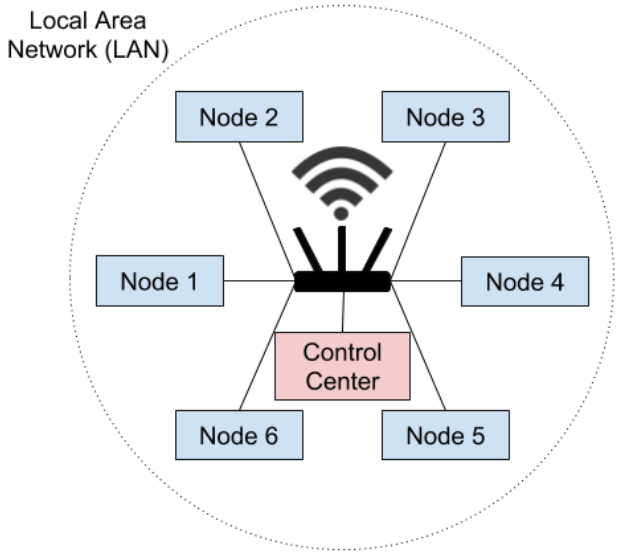


Figure 3.4. Flow of the detection system

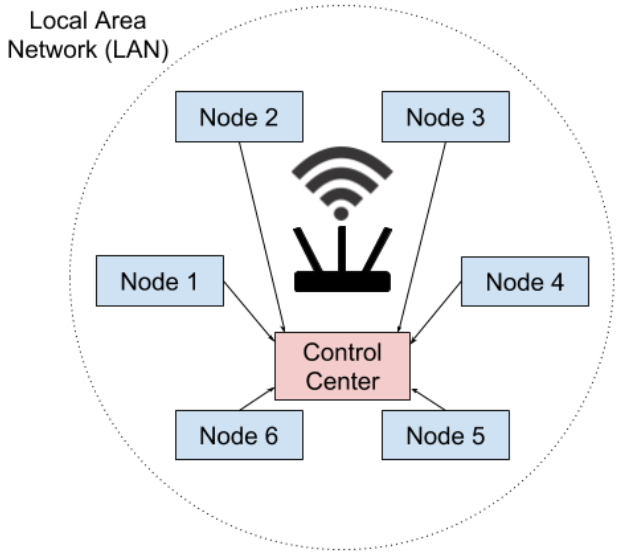
- *Python 3* is an interpreter language that is commonly used for data processing and scientific computing. In this research, Python 3.4 will be used, and the installed packages are managed using virtualenv [44].
- *PyAudio* [45], Python bindings for PortAudio [46], is going to be used as an audio I/O library for the front-end. The first module that streams audio from microphones will be written in Python 3 using this library.
- *Librosa* [47] is a Python library for audio analysis. It offers various feature extraction methods, decomposition and reconstruction, and visualization.
- *Scikit-learn* [48], a Python library for machine learning, is used to build SVM to classify signals and perform dimensionality reduction with PCA.
- *TensorFlow* [49], a library for data flow graphs for scalable machine learning, is used for deep learning model construction.

3.1.3 Network Configuration

In the proposed system, a local area network (LAN) is formed using a router as an access point. The control center and the nodes are connected to the access point via Wi-Fi. Raspberry Pi 3 Model B has a built-in wireless network adapter that supports single-band 2.4 GHz IEEE 802.11b/g/n [50]. For a wider range of communication, a wireless network adapter can be installed on each node. The control center communicates with nodes using



(a) Hardware access point connected to the nodes and the control center



(b) Data transmission from the nodes to the control center

Figure 3.5. Network configuration

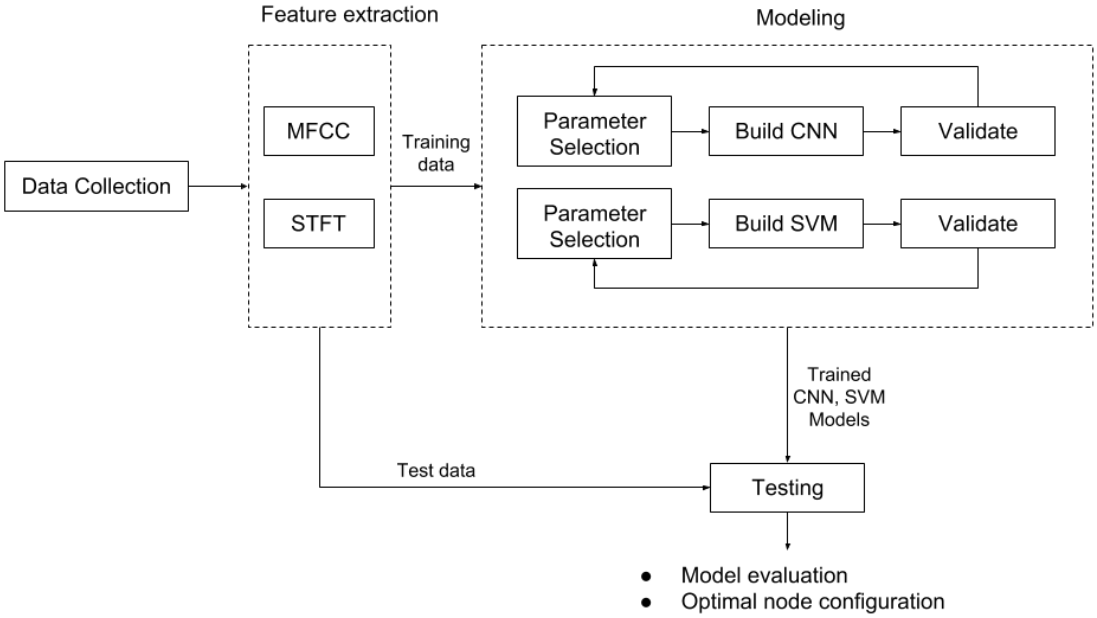


Figure 3.6. Research flow

HTTP APIs and WebSocket to keep sending detection data. The control center acts as a server which takes HTTP POST requests from nodes to alert current detection results. Figure 3.5 (a) shows that the nodes are located within the range of wireless network, and that the hardware access point connects the nodes and the server. Figure 3.5 (b) shows the flow of data from the nodes to the control center. The nodes generate detection results periodically and send them to the control center.

3.2 Research Flow

Figure 3.6 introduces the research methodology step by step. The first procedure is data collection. In this research, the training dataset and the test dataset are collected separately. Different from the standard machine learning methodologies that split the whole data into training, validation, and test set by a certain ratio, the test set is collected in different settings to evaluate models and find the optimal node configurations.

Figure 3.7. AR Drone 2.0 by Parrot

For feature extraction, Mel-frequency cepstral coefficients (MFCC) and short-time Fourier transform (STFT) are obtained from the training data. The next step is modeling, an iterative process of training classification models with combinations of features, algorithms, and parameters. Convolutional neural networks (CNN) and support vector machines (SVM) are trained by repeating the modeling process with different parameters. For each combination, the best performing parameters can be selected.

After obtaining a set of models, tests are done by running the models with the test data collected for the four experiments, which returns a series of binary classification results by six nodes. The purpose of the test is to evaluate the model and find the optimal configuration of nodes based on the evaluation. The test results can be plotted using two different colors, each for the positive and the negative. If the detection is successful, the paths of UAVs should be noticeable on the plot.

3.3 Data Collection

The first thing required for training a machine learning model is to prepare good a training dataset. In this research, the dataset is collected in person and is used to train and test the models. For training data, the audio is recorded using the six listening nodes placed near each other to consider differences between microphones as the pilot flies the

UAV. For testing, the six nodes record audio data, placed on a circle or a half circle in certain formations which will be mentioned in section 3.8.

Only one type of UAVs is used in this research, which is AR Drone 2.0 [51] (figure 3.7) by Parrot. This model is 800g including both the body and the external frame. The UAV was tethered with a long string for safety reasons, but it did not carry any payloads. Below are the three factors considered for data collection.

- *Weather*: The recorded sound has noises according to weather conditions. When it is windy, there should be big wind noises. When it is rainy, it is more noisy with the raining sound, and the range of detection can be smaller. In this research, while the sounds of winds are recorded, rain is not considered. It is also hard for attackers to fly UAVs in a bad weather condition.

- *Distance*: Recordings can be hardly distinguishable when the distance is far. The UAV was flown 0 to 10 meters above the node, and the distance between the UAV and each node was up to 20m. Recordings that the UAV is not audible due to distance were cut off for the training dataset.

- *Variance of hardware*: The dataset should include data from as much hardware as possible because recordings by different devices can show variant ranges of amplitude, which should be reflected in the model.

- *UAV model and payloads*: Each UAV model has its own audio features, and the signals are different when it is carrying payloads. Only one type of UAVs is used with no payloads in this research.

- *Other noises*: Noises include the sounds of planes, human voice, birds, etc. Recordings should be trimmed and selected so that data can represent the labels effectively.

3.4 Preprocessing

In this research, the raw signals are used to extract features. As previously mentioned in figure 2.8, it is common to apply filtering, resampling, and normalization to raw signals. However, the system should run in a real-time setting where the raw stream of audio data should be used because new chunks of audio are coming in periodically. Maximum values of audio files are commonly used as the base value of normalization, but they are undefined in the real-time setting. For that reason, features are obtained straight from raw integer amplitude values.

3.5 Feature Extraction

MFCC and spectrogram using STFT are the two types of features used in this research as in the following subsections.

3.5.1 Mel-Frequency Cepstral Coefficients (MFCC)

MFCC is a feature set that reflects human perception of sounds [52]. It is commonly used in audio classification areas in general, and it is successfully used with machine learning approaches [31]. To get MFCC, first, STFT is computed on each analysis window, yielding a spectrogram. Then, the spectrum is passed through Mel-filters that concentrate on certain frequencies by having more filters at lower frequencies and less at higher frequencies as in figure 3.8. After obtaining the Mel-frequency spectrum from the previous step, a discrete cosine transform (DCT) is applied to log-magnitude of the filter outputs to obtain MFCC.

3.5.2 Spectrogram

A spectrogram is an expression of signals in the time domain with spectral information. Chunks of magnitudes in frequency-domain obtained by conversion using

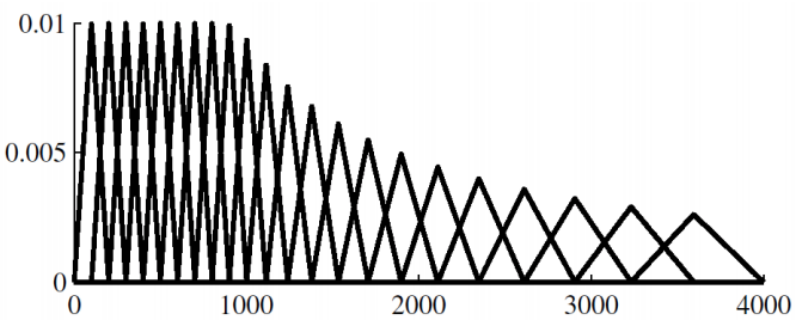


Figure 3.8. Filterbank in Mel-scale with frequency in the X-axis and amplitude in Y-axis [53]

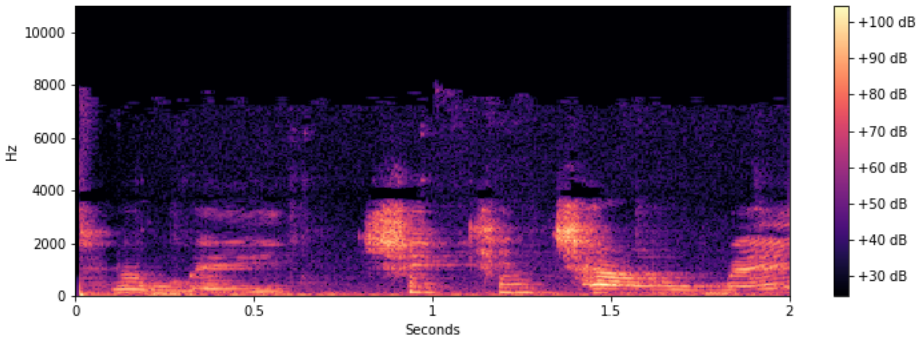


Figure 3.9. Sample STFT spectrogram for female voice

fast Fourier transform (FFT) for a short time are stacked and plotted in the time domain as figure 3.9. Practically, stacking short-time Fourier transform (STFT) generates the spectrogram. When $x(n)$ is an input signal at time n, and the STFT at n with frequency ω_k is defined as [54]:

$$X_n(e^{j\omega_k}) = \sum_{m=-\infty}^{\infty} w(n-m)x(m)e^{-j\omega_k m}. \tag{3.1}$$

where $w(n)$ is the window function, which is Hamming in this research.

STFT is an intermediate feature compared to MFCC in the sense that MFCC compresses signals as it represents them with a set of coefficients. STFT contains more information as well as noises, so MFCC has been used more commonly. Recently, with the advent of deep neural networks, spectrograms are revisited as deep learning models can handle more complicated data than traditional models like Gaussian mixture models [55].

3.6 Detection Models

3.6.1 Support Vector Machine

Support vector machine (SVM) is supervised classifier that learns the boundaries, called optimal separating hyperplane, among classes as in figure 3.10 [56]. SVM is based

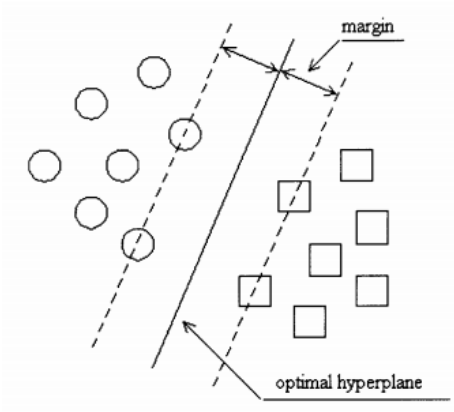


Figure 3.10. SVM classifier after learning optimal hyperplane that maximizes the margin [56]

on kernel functions that "map the input vectors into a very high-dimensional feature space through some nonlinear mapping chosen a priori" [57]. Figure 3.11 shows how the kernel function enables discrimination between two classes that are not linearly separable. There are multiple kinds of kernel functions that can be used with SVM. In this research, the radial basis function (RBF) kernel is used for nonlinear classification. The input space is mapped into the feature space by nonlinear mapping Φ. When $K(\mathbf{x},\mathbf{y})$ is the RBF kernel on the input space (x,y),

$$K(\mathbf{x},\mathbf{y}) = exp(-\gamma||\mathbf{x}-\mathbf{y}||^2)), \tag{3.2}$$

where parameter gamma γ is the free parameter, meaning more variance of the kernel function when the value is smaller. Another parameter `C` is taken into consideration, which is a parameter for the soft margin function. If `C` is big, there is a trade-off with the correctness of classification.

3.6.2 Convolutional Neural Network

Convolutional neural networks (CNN) has brought significant success in pattern recognition or classification in computer vision. Also, researchers approach audio

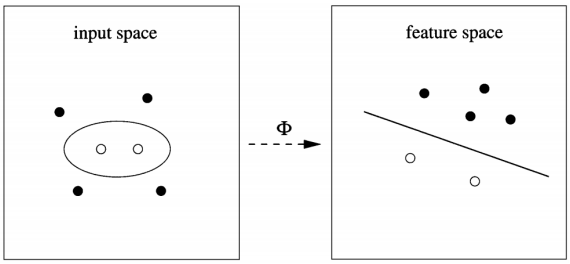


Figure 3.11. Kernel function Φ that maps input vectors into the feature space, enabling linear discrimination in the feature space [56]

classification problem with CNN as audio signals can be represented in two dimensions like an image [24, 30]. CNN is composed of different layers including an input layer, convolutional layers, pooling layers, fully connected layers, and an output layer [21]. On convolutional layers, a small piece of the input vector is passed through the filters, resulting in feature maps. Pooling layers take the output of the convolutional layer as the input and reduce the dimensionality by taking the maximum value from a small piece of the input vector moving by the stride size. After repeating convolution and pooling, the outputs can be seen as features and fed into fully-connected layers, where the learning happens. A sample architecture of a CNN can be illustrated as in figure 3.12, which has two convolutional and max-pooling layers and two fully connected layers. The parameters that should be configured to build a CNN include the number of filters and their size for convolutional layers. For pooling layers, pooling operation, pool size, and stride size is necessary, as well as output size for fully-connected layers. Also, an activation function should be selected as well as learning rate and dropout rate, and rectified linear units (ReLU) is used as the activation function throughout the network.

3.7 Evaluation

F1-score is used as metrics for evaluation as it is commonly used for many machine learning techniques to evaluate classification models. While accuracy measures the ratio of correct predictions among all predictions, it cannot reflect the skewness of the dataset. For F1-score, precision and recall are calculated as equation 3.4 and 3.5 using the counts of true-positives, false-positives, and false-negatives. True-positives represent the cases when the predictions are correct. False-positive cases are when the system detects UAVs even there are no UAVs around. When the system cannot detect UAVs when they are around, it is a false-negative.

$$F_1 = 2 \cdot \frac{precision \cdot recall}{precision + recall}, \tag{3.3}$$

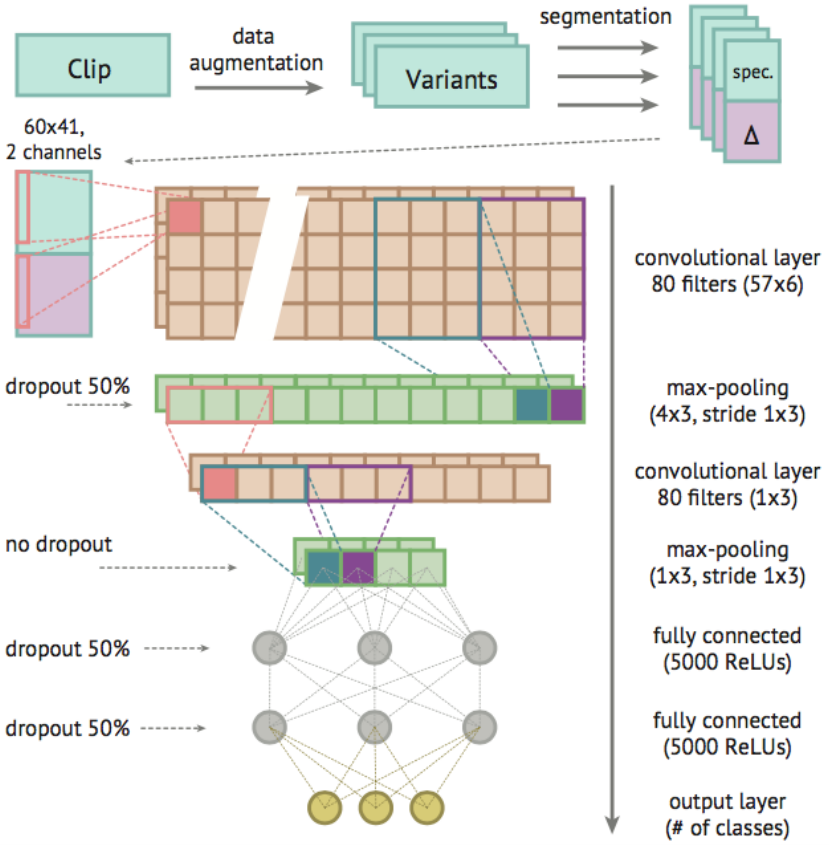


Figure 3.12. A model CNN architecture for short-clip audio classification [30]

where

$$Precision = \frac{TP}{TP+FP}, \tag{3.4}$$

$$Recall = \frac{TP}{TP+FN}. \tag{3.5}$$

TP is true-positives, FP is false-positives, and FN is false-negatives.

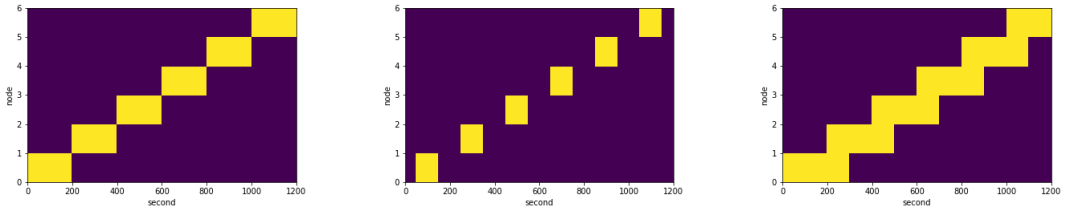
3.8 Testing

Two sub-goals are essential for essential for the development of a UAV detection system. The other is to determine the optimal configuration of hardware components. After the training phase, tests are done to evaluate the model and to determine the configuration of the nodes. During the testing, test data is collected by placing nodes on the formations and flying the UAV and then feed the data into the models. If the nodes are placed on a proper formation, the path of the UAV can be noticeable when the results are plotted as a color map.

For example, figure 3.13 shows virtual test results assuming that the quality of the model stays the same for every device with a clear boundary for detection. The X-axis is time, and the Y-axis is the node number, and the bright area means positive results and the dark area means negative results. When the UAV flies following the nodes from node 1 to node 6, the bright areas should make a diagonal form as in all three of the figures. From these color maps, optimal configurations can be found by the distance between each bright area. Figure 3.13 (a) is the case when the nodes are as apart as the detection range, (b) is when the nodes are farther than detection range, and (c) is when the nodes are placed closer than the detection range.

The formation of the nodes is circular surrounding the protected area as in figure 3.14 (a). The formation has three parameters to consider for the experiments:

- *Radius* is the distance between a node and the protected area that can affect the range of detection.

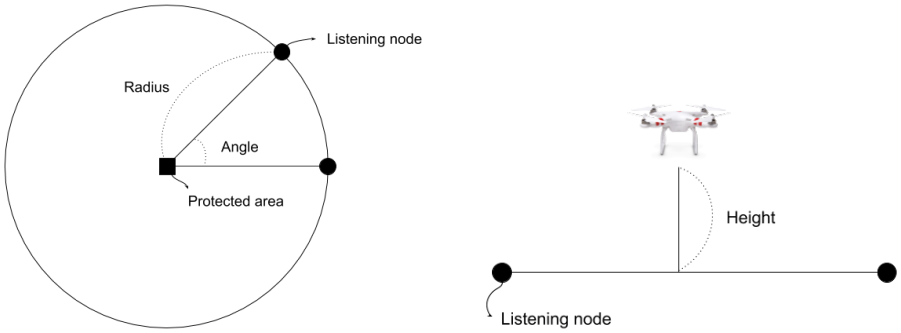


(a) (b) (c)

Figure 3.13. Virtual results of six nodes that are plotted with dark and bright color each representing negative and positive results

Table 3.1. Set of Experiments

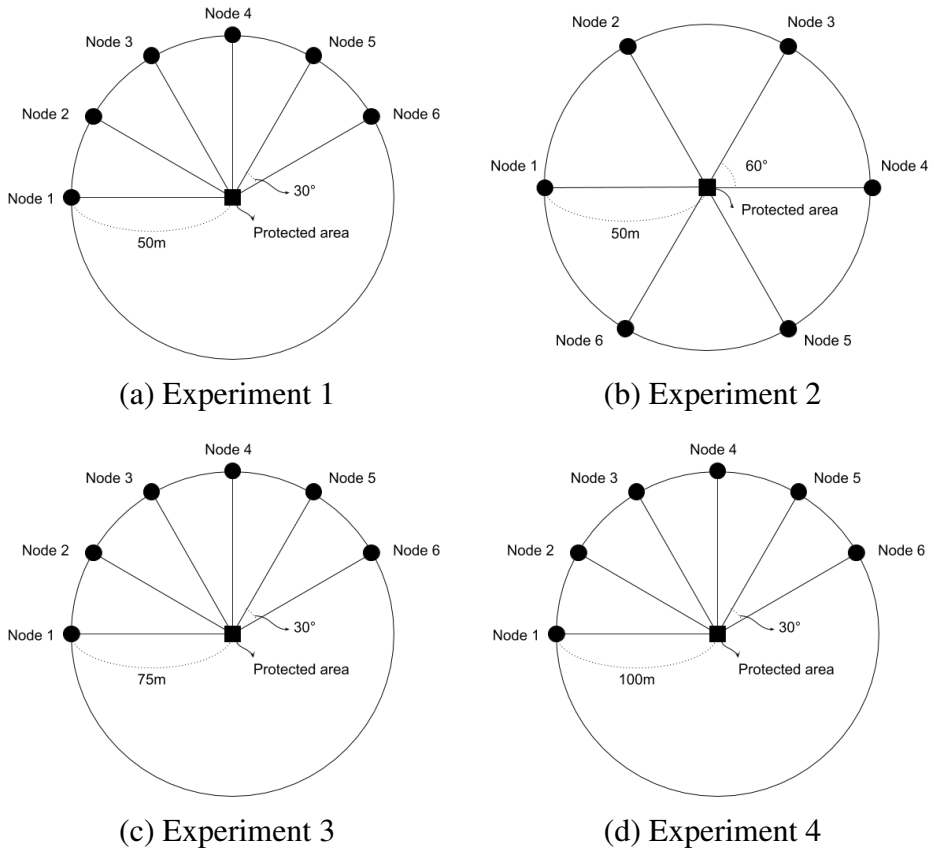
Exp #	Radius(m)	Angle(°)	# Nodes
1	50	30	6
2	50	60	6
3	75	30	6
4	100	30	6



(a) View from the top (b) View from the side

Figure 3.14. Experimental parameters from the top and the side

- *Angle* measures the angle between two lines connecting the protected area and two adjacent nodes.
- *Height* is the vertical distance between the UAV and the ground where nodes are set up.

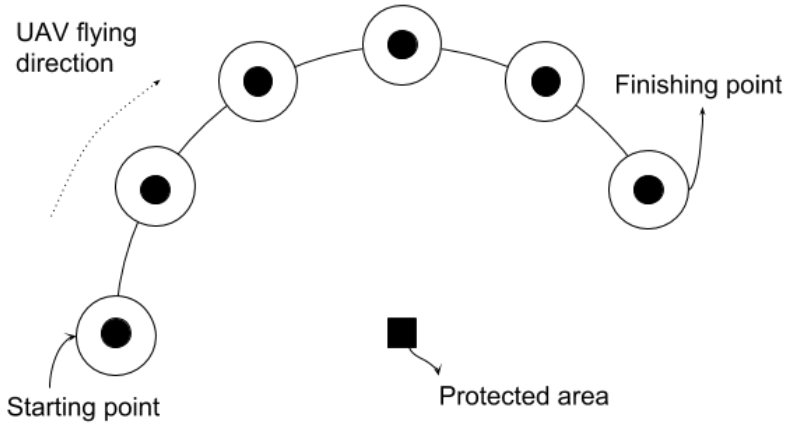


(a) Experiment 1 (b) Experiment 2

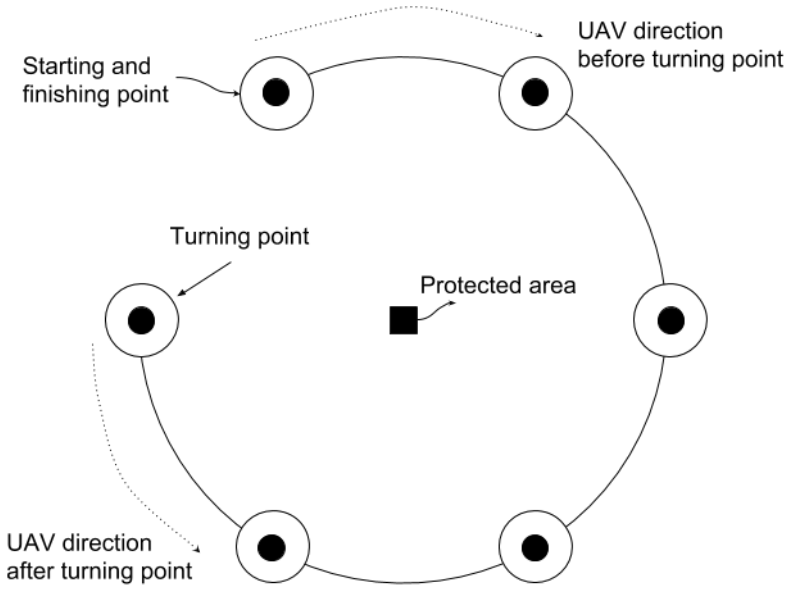
(c) Experiment 3 (d) Experiment 4

Figure 3.15. Experimental configurations

Four experiments are performed varying the radius and angle with the maximum height fixed at 5 meters as in table 3.1. The nodes are placed as in figure 3.15 accordingly. The pilot flies the UAV following the perimeter of the circle and around each node as in figure 3.16 for testing. The trajectory was designed to find the configurations and to cover all angles of the sound. The nodes record the audio data in these configurations. Each audio file is fed into the detection models, and the color maps are generated. If the models are valid, the color maps should have a noticeable pattern. When the patterns are visible, the optimal configurations can be chosen by comparing the patterns of each configuration.



(a) Trajectory for experiment 1, 3 and 4



(b) Trajectory for experiment 2

Figure 3.16. Trajectory for experiments

3.9 Near Real-time Environment

To provide users with a tool to monitor detection results using web browsers, a web page with dashboards plotting detection results and performance using ElasticSearch [58] and Kibana [59]. Each node runs a detection model and transmits the result to a Tornado [60] server written in Python periodically using WebSocket. According to the standards [61], "WebSocket is a technology providing for bi-directional, full-duplex communication channels, over a single Transmission Control Protocol (TCP) socket in web browsers and web servers, but it can be used by any client or server application"

3.10 Summary

This chapter provided the methodologies for this research, including the framework and experiment configurations.

CHAPTER 4. EXPERIMENTS

Audio detection methods introduced in the previous chapter are tested. This chapter elaborates detailed experiment environments and the results. It starts with details of results of data collection including types of sounds included in the dataset and the basic analysis on the raw signals. Then, section 4.2 describes the features of the collected signals. Section 4.3 introduces the process of training SVM and CNN. Lastly, section 4.4 shows test results using six nodes deployed in configurations as shown in the previous chapter.

4.1 Training Data Collection

Audio data was collected using six nodes at McAllister Park in Lafayette. Figure 4.1 describes the area where the experiments are done. Data for the negative class is raw audio data of the environment noises without UAVs flying around while data in the

Figure 4.1. McAllister Park

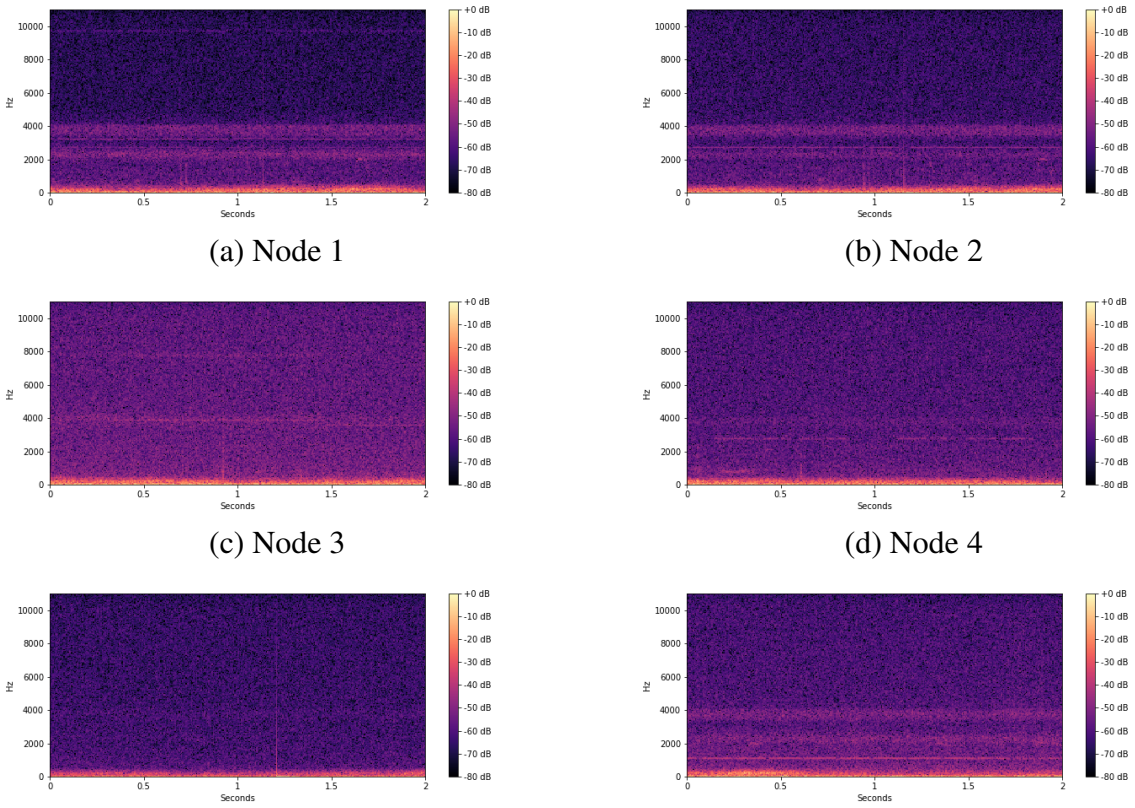
Table 4.1. Composition of the training dataset

Type of sounds	Minutes in positive class	Minutes in negative class
Quiet surrounding	1	1
Wind	6	6
Human voice	1	1
Airplane	0.5	0.5

positive class includes raw audio files where the UAV is audible as well as background noises. The type of noises recorded in the training audio files includes sounds of insects across the whole recording, occasional human voices, airplanes, and winds as in 4.1. Strong wind takes up half of the total wind recordings, and the rest is composed of milder wind. Human voices or airplane sounds were recorded along with other sounds like winds. To enable classification with noises, the training dataset of the negative and the positive labels include the same type of noises so that the only difference is the UAV sound. The recordings last 14 minutes in total for each class.

Additionally, each recording has different sound quality across the nodes although hardware models for sound cards and the nodes were the same. Figure 4.2 is the plots of the background noises recorded by six nodes at the same time when the surroundings were quiet. The amplitudes on the range of 0 Hz to 4000 Hz are different from each node. It results in incorrect classification if the training dataset is composed of one node. All six nodes were engaged in the recording process to prevent this issue.

he training dataset was randomly divided into two parts, training and validation, to evaluate these models while developing them. 80% of the data was used as the training data, and 20% was used for evaluation.



(a) Node 1

(b) Node 2

(c) Node 3

(d) Node 4

(e) Node 5

(f) Node 6

Figure 4.2. Spectrograms of six nodes in a quiet environment

4.2 Feature Extraction

Figure 4.4 shows STFT of different sounds for two seconds to show differences in sounds. The X-axis represents time, and the Y-axis represents frequency. STFT is obtained from raw amplitude sequences and expressed in dB scale to plot the spectrogram. Figure 4.4 (a) records quiet surrounding without other noises. The amplitude stays consistent and even from 0 Hz to 4000 without noticeable patterns. Strong wind appears as bright colors in random spikes in a lower frequency range between 0 Hz to 500 Hz as in figure 4.4 (b). Human voices appear as short signatures with various patterns depending on the person and the contents. It is shown from 0.1 to 0.6 seconds and from 1.7 to 2.0 seconds on figure 4.4 (c). UAV sounds have obvious patterns as figure 4.4 (d) and (e) when it is within a certain distance. However, when the UAV is far like in 4.4 (f), it can hardly be differentiated with the background noise. Figure 4.5 plots MFCC of the same sounds in figure 3.9. MFCC shows similar features as STFT does, but the differences between the sounds are not as apparent as STFT.

As mentioned in the previous chapter, feature extraction and classification are performed periodically as near real-time processing is assumed. The features are calculated every few seconds and put into the model that returns 0 or 1.

As in figure 4.3, under the near real-time environment, the system should grab a small slice of audio, called a chunk, from the audio stream and extract features of the chunk, which is represented as X_mfcc or X_spectrogram. Repeatedly, the system grabs the next audio chunk and extracts the features. The chunk sizes in use are 2, 1 and 0.5 seconds. Functions from librosa library, `librosa.feature.mfcc` and `librosa.stft`, are used for MFCC and STFT respectively. The code for calculating MFCC using librosa is as listing 4.1, and the code for calculating STFT is as listing 4.2.

```
start = 0
mfcc = []
for raw signals:
    chunk = raw[start:start+chunk_size]
    # get mel-scale spectrogram
    S = librosa.feature.melspectrogram(chunk, n_mels=128)
    # get 16 MFCCs from melspectrogram in db scale
    mfcc_chunk = librosa.feature.mfcc(S=librosa.power_to_db(S), n_mfcc=16)
    mfcc.append(mfcc_chunk)
```

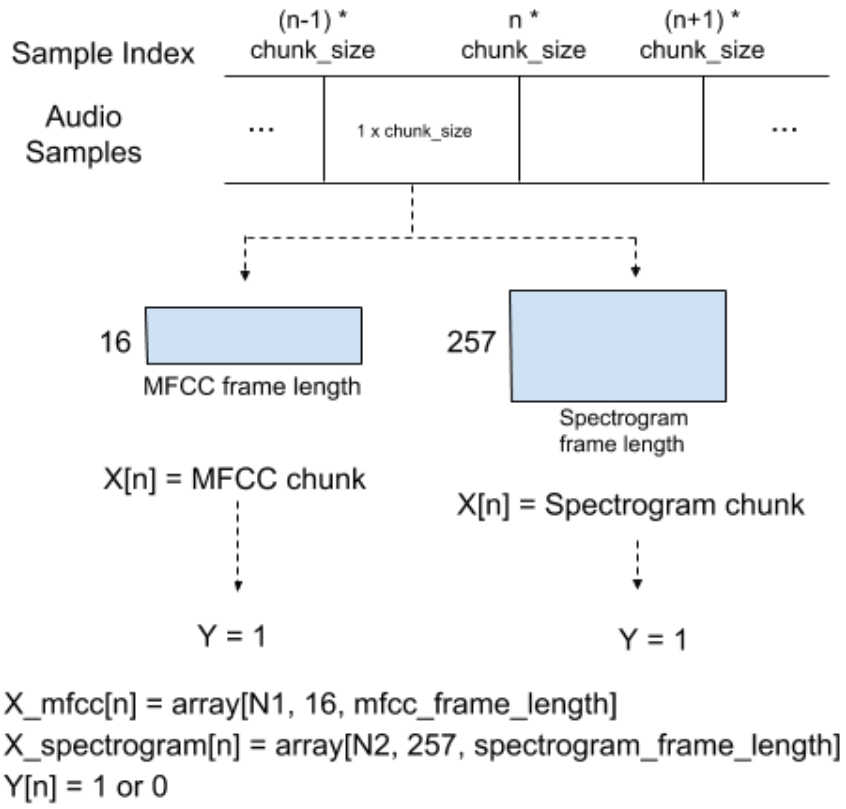


Figure 4.3. MFCC and STFT as input data and coresponding Y

```
    start += window_size
```

Listing 4.1: Pseudo code for MFCC method

```
start = 0
stft = []
for raw signals:
    chunk = raw[start:start+chunk_size]
    # get STFT
    S = librosa.stft(chunk)
    # transfer to db scale
    stft_chunk = librosa.amplitude_to_db(S)
    stft.append(stft_chunk)
    start += window_size
```

Listing 4.2: Pseudo code for STFT method

4.3 Model Training

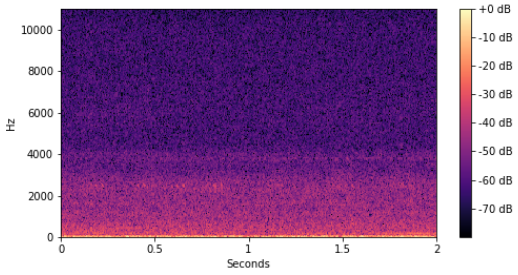
The detection models are binary classifiers with two classes representing the UAV sound and the background sound. Support vector machines (SVM) and convolutional neural networks (CNN) are trained with the features mentioned in the previous section.

4.3.1 Support Vector Machines (SVM)

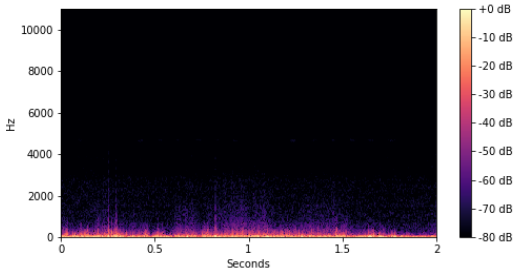
For SVM, scikit-learn library is used for implementation. Modules such as `sklearn.svm` and `sklearn.model_selection.GridSearchCV` are primarily used. Kernel radial basis function is chosen as the kernel function for non-linear classification. The parameters that should be found are C and gamma as mentioned in the previous chapter. C and gamma are found through the grid search method, exhaustively searching for the best performing parameter pair over given parameter values as below.

- C: $C = 10^i$, where $i = 1, 2, ..., 14, 15$
- gamma: $\gamma = 10^i$, where $i = -15, -14, ..., -2, -1$

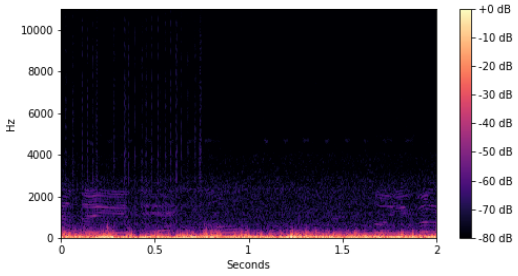
For each parameter, the model was cross-validated by the factor of three. The code snippet for declaring grid search model is as listing 4.3. Among the arguments in the code, `cv` stands for cross validation, and `scoring` means evaluation method, which is the f1 score.



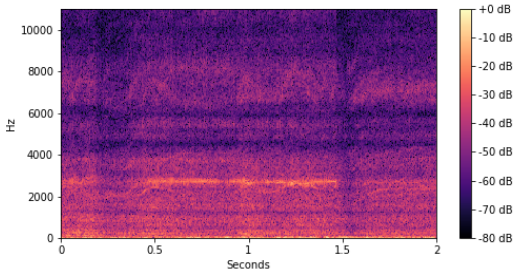
(a) Quiet environment



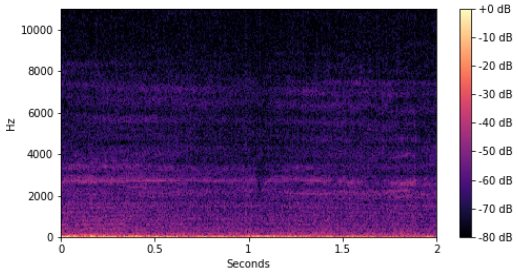
(b) Strong wind



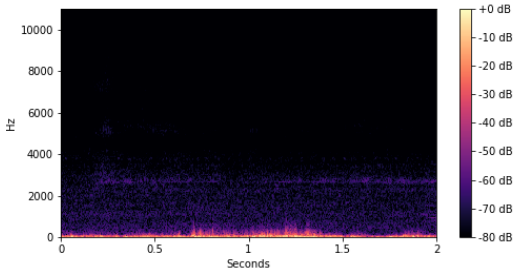
(c) Human voice and wind



(d) UAV flying next to a node

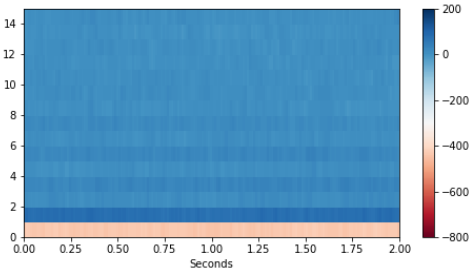


(e) UAV flying within 10 meters

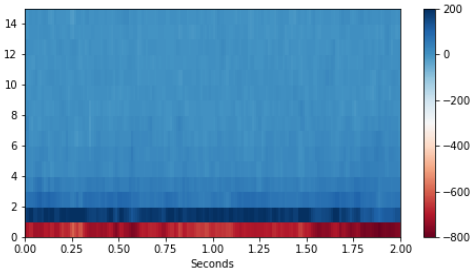


(f) UAV flying more than 30 meters away

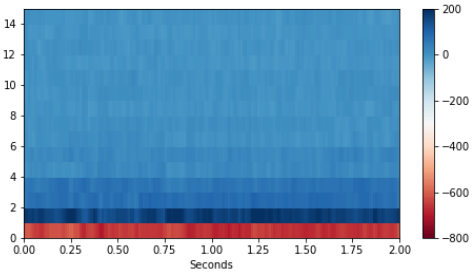
Figure 4.4. Spectrograms of different sounds



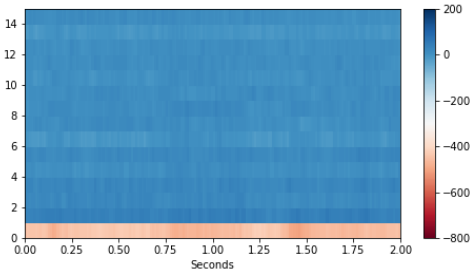
(a) Quiet environment



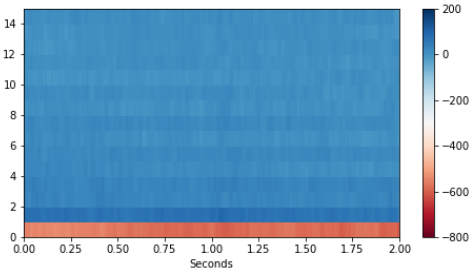
(b) Strong wind



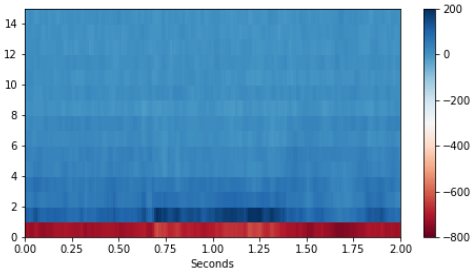
(c) Human voice and wind



(d) UAV flying next to a node



(e) UAV flying within 10 meters



(f) UAV flying more than 30 meters away

Figure 4.5. MFCCs of different sounds

Table 4.2. Results of training STFT-SVM

Chunk size	C	Gamma	F1-score	Chunk size	C	Gamma	F1-score	Chunk size	C	Gamma	F1-score
0.5	10^2	10^{-7}	0.704	1	10^{10}	10^{-10}	0.740	2	10^3	10^{-9}	0.785
		10^{-9}	0.732			10^{-12}	0.774			10^{-11}	0.724
		10^{-11}	0.584			10^{-14}	0.521			10^{-13}	0.569
	10^3	10^{-7}	0.690		10^{11}	10^{-10}	0.740		10^5	10^{-9}	0.778
		10^{-9}	**0.768**			10^{-12}	0.768			10^{-11}	**0.795**
		10^{-11}	0.700			10^{-14}	0.521			10^{-13}	0.726
	10^4	10^{-7}	0.671		10^{12}	10^{-10}	0.740		10^7	10^{-9}	0.778
		10^{-9}	0.754			10^{-12}	**0.787**			10^{-11}	0.775
		10^{-11}	0.730			10^{-14}	0.521			10^{-13}	0.749
	10^5	10^{-7}	0.671		10^{13}	10^{-10}	0.740		10^9	10^{-9}	0.778
		10^{-9}	0.739			10^{-12}	0.772			10^{-11}	0.775
		10^{-11}	0.766			10^{-14}	0.521			10^{-13}	0.750

Table 4.3. Results of training MFCC-SVM

Chunk size	C	Gamma	F1-score	Chunk size	C	Gamma	F1-score	Chunk size	C	Gamma	F1-score
0.5	10^3	10^{-6}	0.691	1	10^4	10^{-8}	0.744	2	10^3	10^{-6}	0.661
		10^{-8}	0.722			10^{-9}	0.772			10^{-8}	0.746
		10^{-10}	0.724			10^{-10}	0.770			10^{-10}	0.733
		10^{-12}	0.555			10^{-11}	0.732			10^{-12}	0.531
	10^5	10^{-6}	0.691		10^5	10^{-8}	0.744		10^5	10^{-6}	0.661
		10^{-8}	0.721			10^{-9}	0.766			10^{-8}	0.743
		10^{-10}	**0.761**			10^{-10}	**0.779**			10^{-10}	0.784
		10^{-12}	0.703			10^{-11}	0.771			10^{-12}	0.729
	10^7	10^{-6}	0.691		10^5	10^{-8}	0.744		10^7	10^{-6}	0.661
		10^{-8}	0.721			10^{-9}	0.766			10^{-8}	0.743
		10^{-10}	0.735			10^{-10}	0.770			10^{-10}	**0.786**
		10^{-12}	0.720			10^{-11}	0.778			10^{-12}	0.687

Table 4.4. Convolutional Neural Network Parameters for MFCC

Layer	Component	Parameter
Convolutional layer 1	Kernel size	[13,13]
	Initializer	Xavier
	Number of filters	16
	Activation function	ReLU
Pooling layer 1	Pool size	[3, 3]
	Stride size	2
	Padding	'SAME'
Convolutional layer 2	Kernel size	[3, 3]
	Initializer	Xavier
	Number of filters	16
	Activation function	ReLU
	Padding	'SAME'
Pooling layer 2	Pool size	[3, 3]
	Stride size	2
	Padding	'SAME'
Dense layer 3	Output size	100
	Activation	ReLU
Dropout layer 3	Dropout rate	0.5
Dense layer 4	Output size	10
	Activation	ReLU
Dropout layer 4	Dropout rate	0.5

Parameters are given as a Python dictionary data structure, `parameters`. This process is repeated for MFCC and STFT alternating the chunk size. The training results are shown in table 4.2 and 4.3. The results with the highest accuracy are bolded and chosen as the final model.

```
model = GridSearchCV(svm.SVC(), parameters, cv=3,
                     scoring='f1')
```

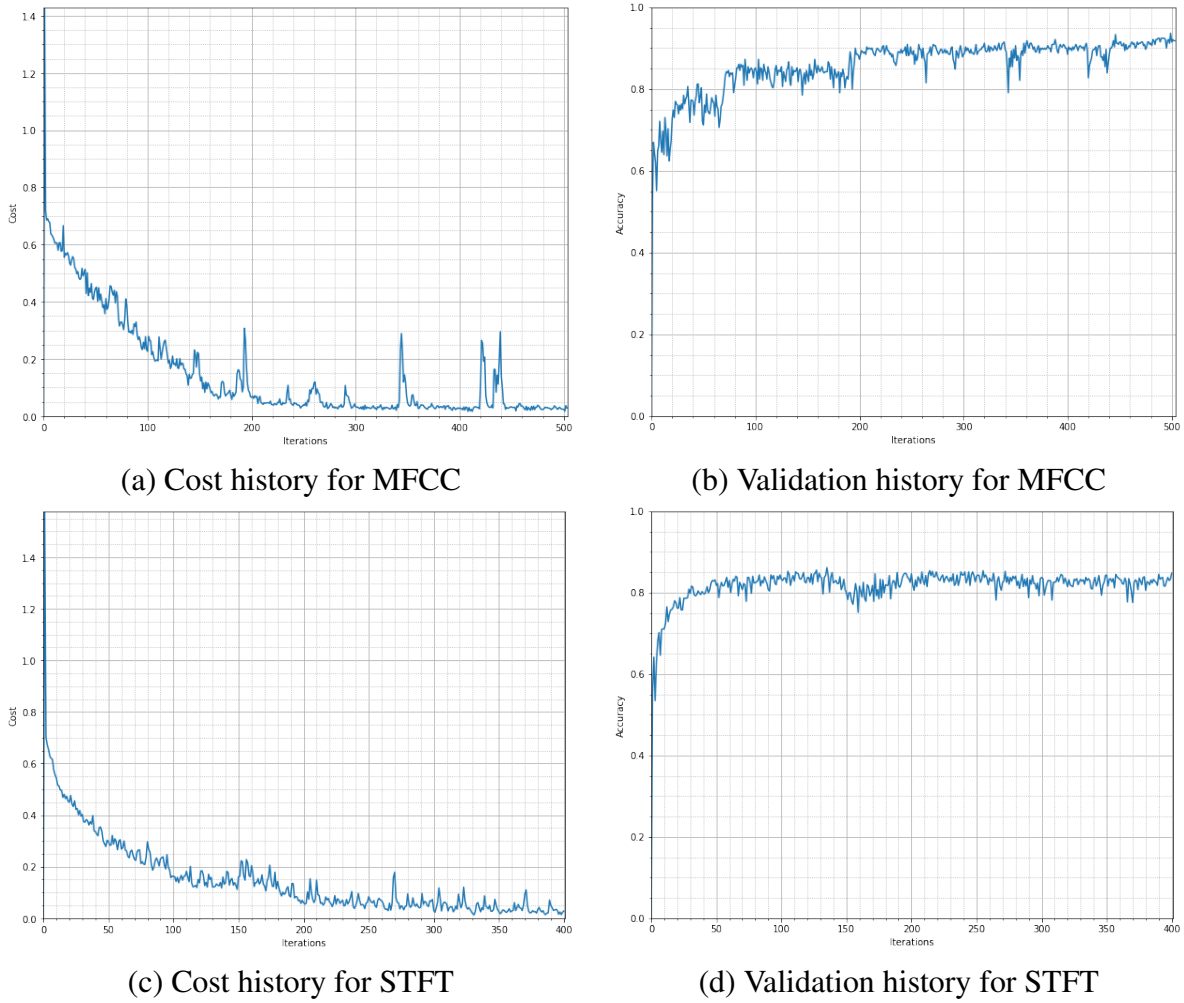
Listing 4.3: Grid search model definition

4.3.2 Convolutional Neural Networks (CNN)

Contrary to SVM, training a CNN is based on heuristics that the parameter values are selected manually because an exhaustive search for the best combination is not

Table 4.5. Convolutional Neural Network Parameters for STFT

Layer	Component	Parameter
Convolutional layer 1	Kernel size	[3, 3]
	Initializer	Xavier
	Number of filters	16
	Activation function	ReLU
Pooling layer 1	Pool size	[3, 3]
	Stride size	2
	Padding	'SAME'
Convolutional layer 2	Kernel size	[4, 4]
	Initializer	Xavier
	Number of filters	8
	Activation function	ReLU
	Padding	'SAME'
Pooling layer 2	Pool size	[4, 4]
	Stride size	3
	Padding	'SAME'
Dense layer 3	Output size	500
	Activation	ReLU
Dropout layer 3	Dropout rate	0.5
Dense layer 4	Output size	50
	Activation	ReLU
Dropout layer 4	Dropout rate	0.5



(a) Cost history for MFCC

(b) Validation history for MFCC

(c) Cost history for STFT

(d) Validation history for STFT

Figure 4.6. Cost and validation history per iteration

feasible. Also, the training process was based on a single fold. The considered factors are number of layers, size of filters, number of filters, learning rate, and dropout.

Table 4.4 shows the final MFCC-CNN architecture with two convolutional layers and two dense layers. Considering that 16 coefficients were used for MFCC, the size of the filter, [13, 13], can be seen too large. However, it gives higher accuracy than using a smaller filter. The patterns are not repeated along the axis of MFCC because the frequency range is unique to each sound. Training iteration is set to 1,000, and the process is stopped when the cost curve and validation curve start to become constant.

Similar to the MFCC-CNN model, STFT-CNN model is built with two convolutional layers and two dense layers. The major difference is the filter shape of the first convolutional layer because the STFT has more values per frame. The filter size of the model is [3, 3] because rectangular-shaped filters did not converge during training as the values were more diverse.

Figure 4.6 shows the graphs of the cost and accuracy per iteration. For MFCC with 2-second chunks, figure 4.6 (a) and (b) shows the curves, finishing the training at iteration around 500 with 0.9 of accuracy. According to figure 4.6 (c) and (d), STFT with 2-second chunks is finalized around the 400th iteration with 0.85 of accuracy.

4.4 Test Environment

Test data is collected in four configurations as mentioned in section 3.8. Figure 4.7 shows how nodes are placed when they are in semi-circular formations such as in experiment 3 with a radius of 75 meters. Figure 4.8 is the view of the nodes from the sky in experiment 4 with the radius of 100 meters.

The UAV is tethered like a kite to a string for safety reasons. Two people stay close to the UAV and fly the UAV along the nodes. One person pilots the UAV using a mobile application. Another person follows the UAV and holds the string back to stop the UAV if the UAV becomes out of control as in figure 4.9. The UAV flew about 5 meters above the ground on average, but the height kept changing as the UAV is lightweight and the wind is strong.

4.5 Test Results

Test results for each model and experiment are shown in color maps from figure 4.11 to figure 4.18. Figure 4.10 shows the mapping between the color maps and the actual trajectory of the UAV flying from node 1 to node 6. Due to strong winds and the weight of the UAV, the UAV could not fly straight from one node to another. However, the range of UAV is mostly limited to 20 meters, shown as the yellow area around the perimeter.

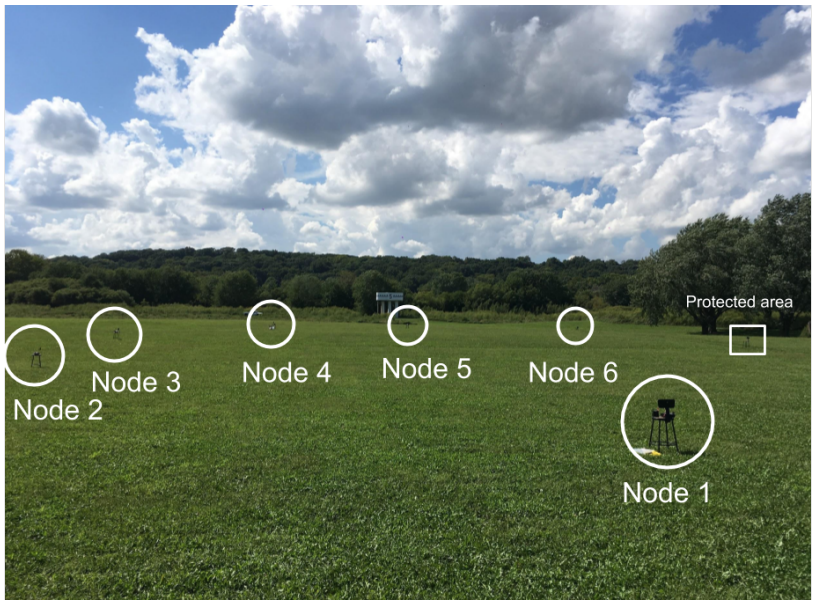


Figure 4.7. Node placement of experiment 3

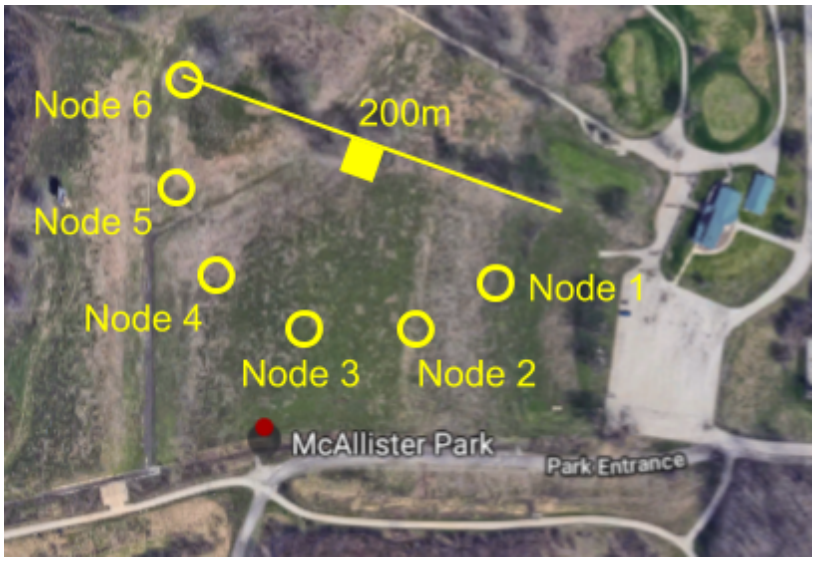


Figure 4.8. Bird's-eye view of the nodes in experiment 4

Figure 4.9. UAV flying around a node

The X-axis of the color map is minutes, and the Y-axis is the node number. Dark areas represent negative detection results, and bright areas represent positive results. Each red box indicates a period when UAV sounds are significant, meaning the UAV is close to the corresponding node. For example, as in figure 4.10, there are six boxes, and two of them are mapped to the trajectory. When the detection is successful, the red box areas should match bright areas.

In terms of modeling, STFT-SVM model has the best results among the four models. Each model has its unique characteristics. MFCC-SVM model with the 2-second chunk size cannot distinguish between two classes, returning all negative results for every experiment. While the red box areas of experiment 1 match the path for the 1-second chunk size and the 0.5-second chunk size, it can hardly be seen for experiment 2 because there were constant plane sounds for experiment 2. The model is not able to clearly distinguish UAV sounds and plane sounds, leading to dominant false positive results. The paths do not match according to the results of MFCC-SVM for experiment 3 and 4.

Compared to MFCC-SVM, STFT-SVM model returns better results with more explicit paths. Among the three STFT-SVM models, false negatives are the least when the chunk size is 2 seconds.

Although MFCC is widely used for audio classification tasks, it can be bad for UAV detection because both wind and UAV have a stronger amplitude on lower frequency bands. MFCC contains more dense information of sounds as it represents sounds with several coefficients, while STFT is relatively an intermediate feature. For that reason, UAV can be better distinguished when STFT is used.

MFCC-CNN with 1-second chunks shows a clear path with matching bright areas particularly for experiment 1. In experiment 2 and 4, meaningful patterns are difficult to find. For experiment 3, there is a higher portion of true positives than other MFCC-CNN results specifically for the 1-second chunk size, showing solid bright areas in most of the red boxes.

STFT-CNN is ambiguous as each chunk size shows different results. Its 2-second chunk size model can be said to be random as it is hard to see patterns. For the 1-second

chunk size, the patterns are noticeable but very light with a lot of false negatives while there are a lot of false positives for the 0.5-second chunk size.

Whereas the SVM model shows noticeable patterns with STFT, the results of CNN are ambiguous. CNN has more fluctuations than SVM, and the patterns are more random, so it is hard to find a solid dark or bright area. SVM is more appropriate for this task, considering the simplicity of the model. Also, training SVM is simpler because the grid search method can easily find the parameters.

The optimal configuration can be derived after selecting STFT-SVM with the 2-second chunk size for the model. The results of the model are shown in figure 4.19 with synchronized time axis to allow comparison. Considering the distance between bright areas, experiment 3 from figure 4.19 (b) shows the best results. Experiment 1 has a higher portion of overlapped areas compared to experiment 3 and 4. Experiment 4 shows that bright areas are far apart while bright areas in experiment 3 are closely located. It is evident that configurations of experiment 3 have a higher chance of detecting UAVs without blind areas when a UAV flies between nodes.

Using these color maps, not only the path but also the speed of the UAV can be estimated from the length of the bright area and the distance between nodes. It can give additional insight on analyzing threat by providing data on how long the UAV stayed around a node.

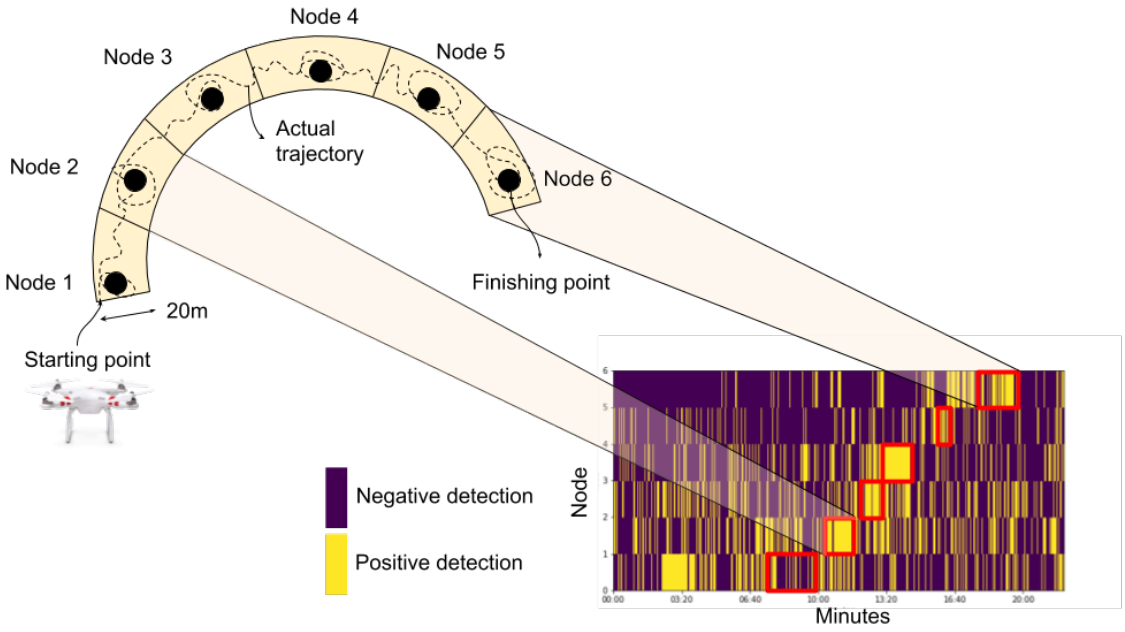
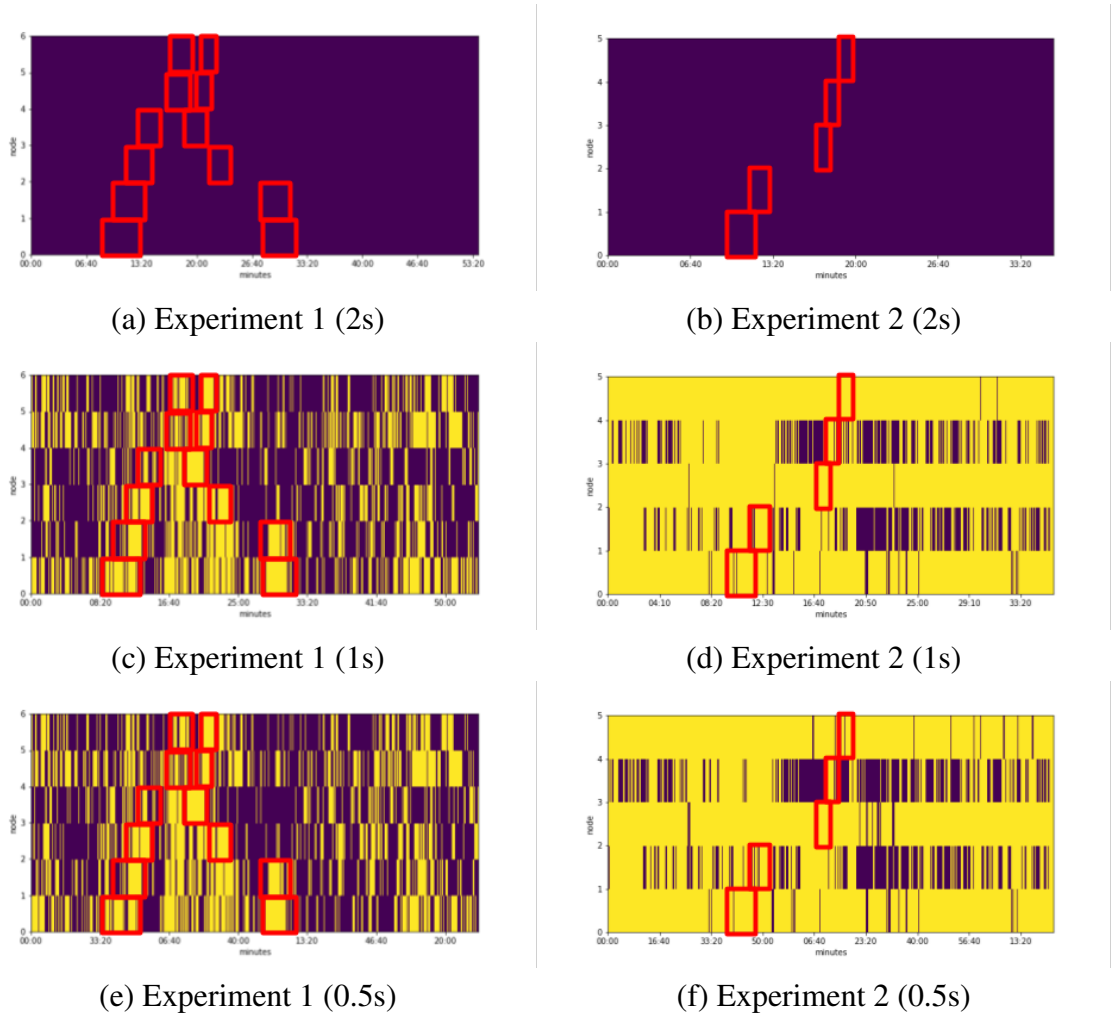


Figure 4.10. Mapping between result color maps and trajectory



(a) Experiment 1 (2s)

(b) Experiment 2 (2s)

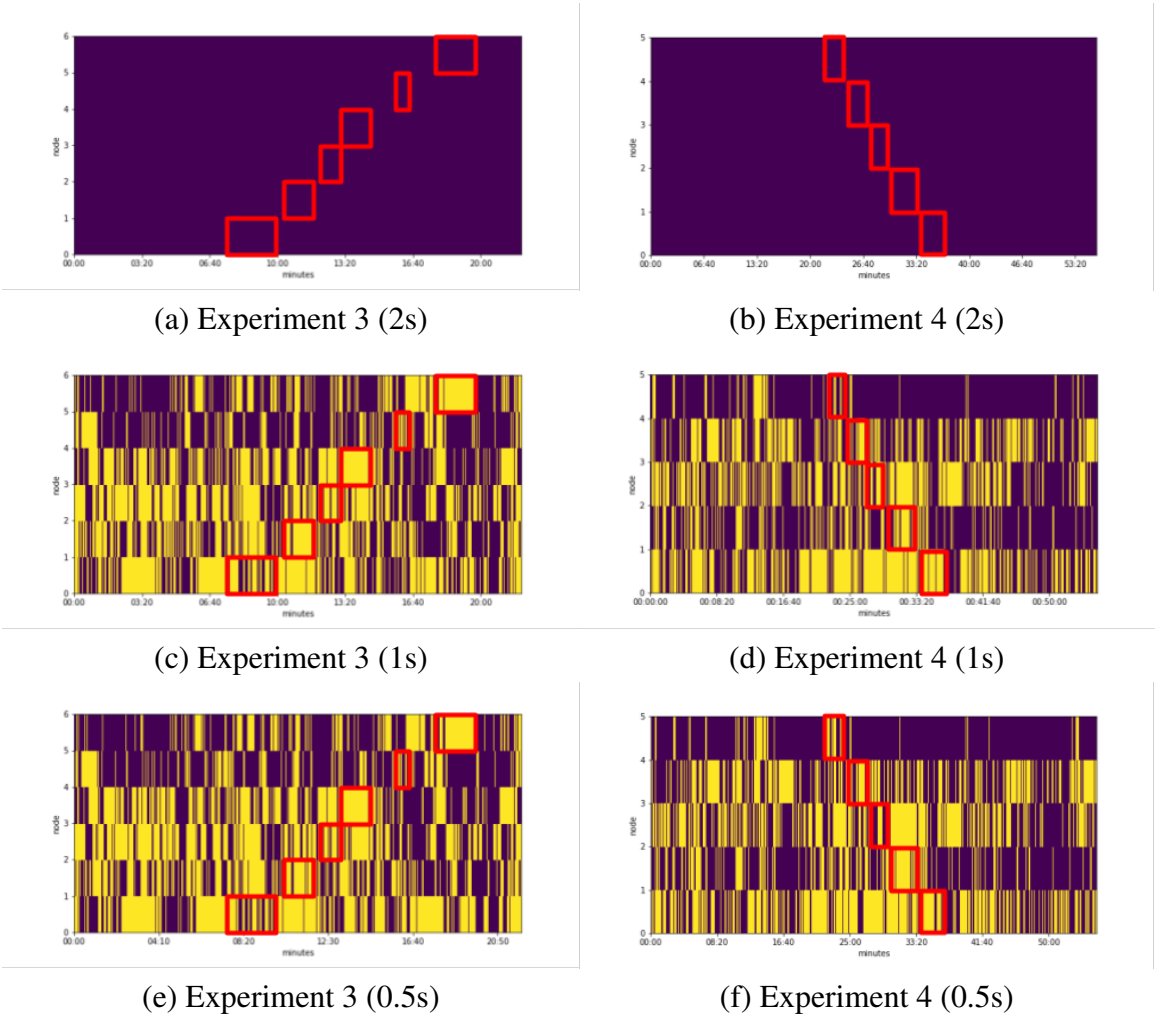
(c) Experiment 1 (1s)

(d) Experiment 2 (1s)

(e) Experiment 1 (0.5s)

(f) Experiment 2 (0.5s)

Figure 4.11. MFCC-SVM model for experiment 1 and experiment 2



(a) Experiment 3 (2s)

(b) Experiment 4 (2s)

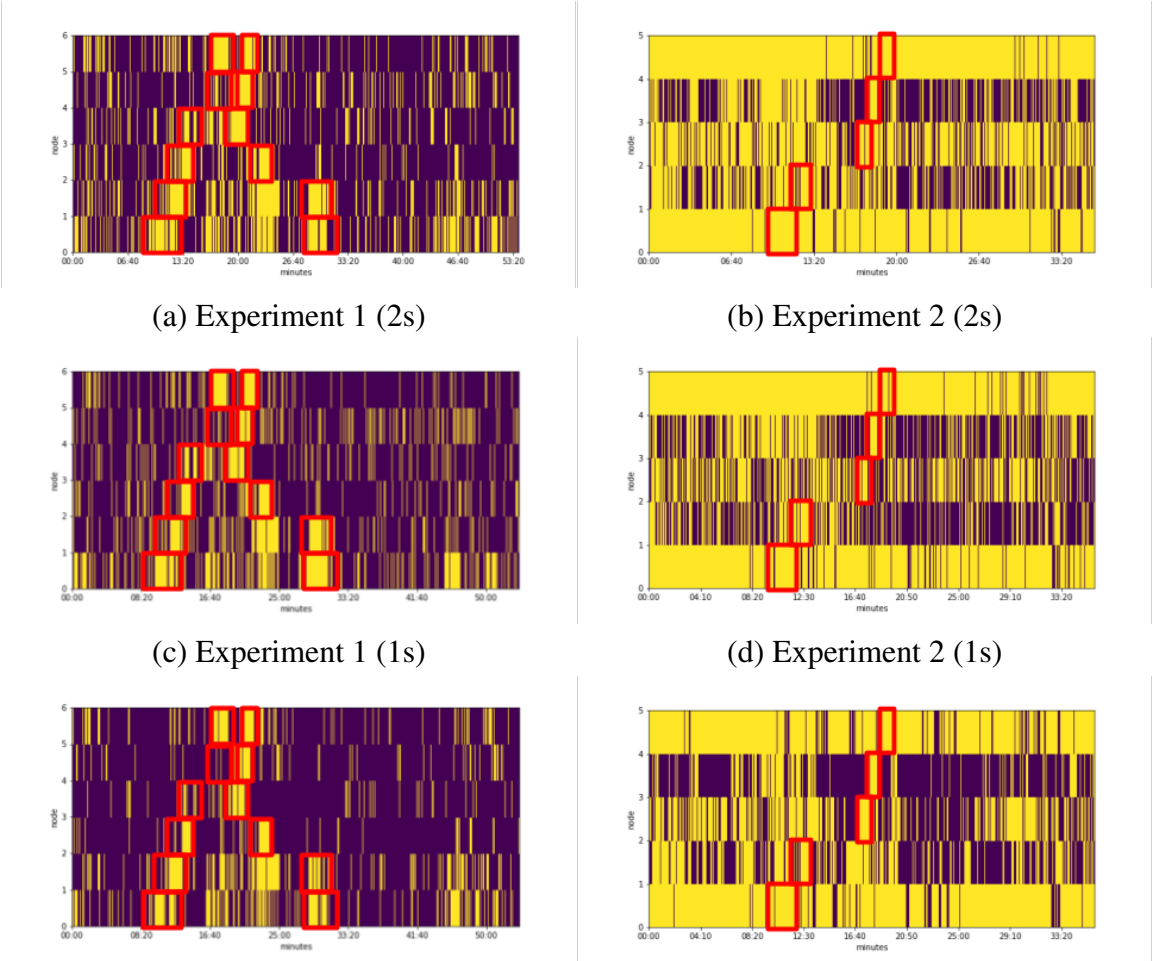
(c) Experiment 3 (1s)

(d) Experiment 4 (1s)

(e) Experiment 3 (0.5s)

(f) Experiment 4 (0.5s)

Figure 4.12. MFCC-SVM model for experiment 3 and experiment 4



(a) Experiment 1 (2s)

(b) Experiment 2 (2s)

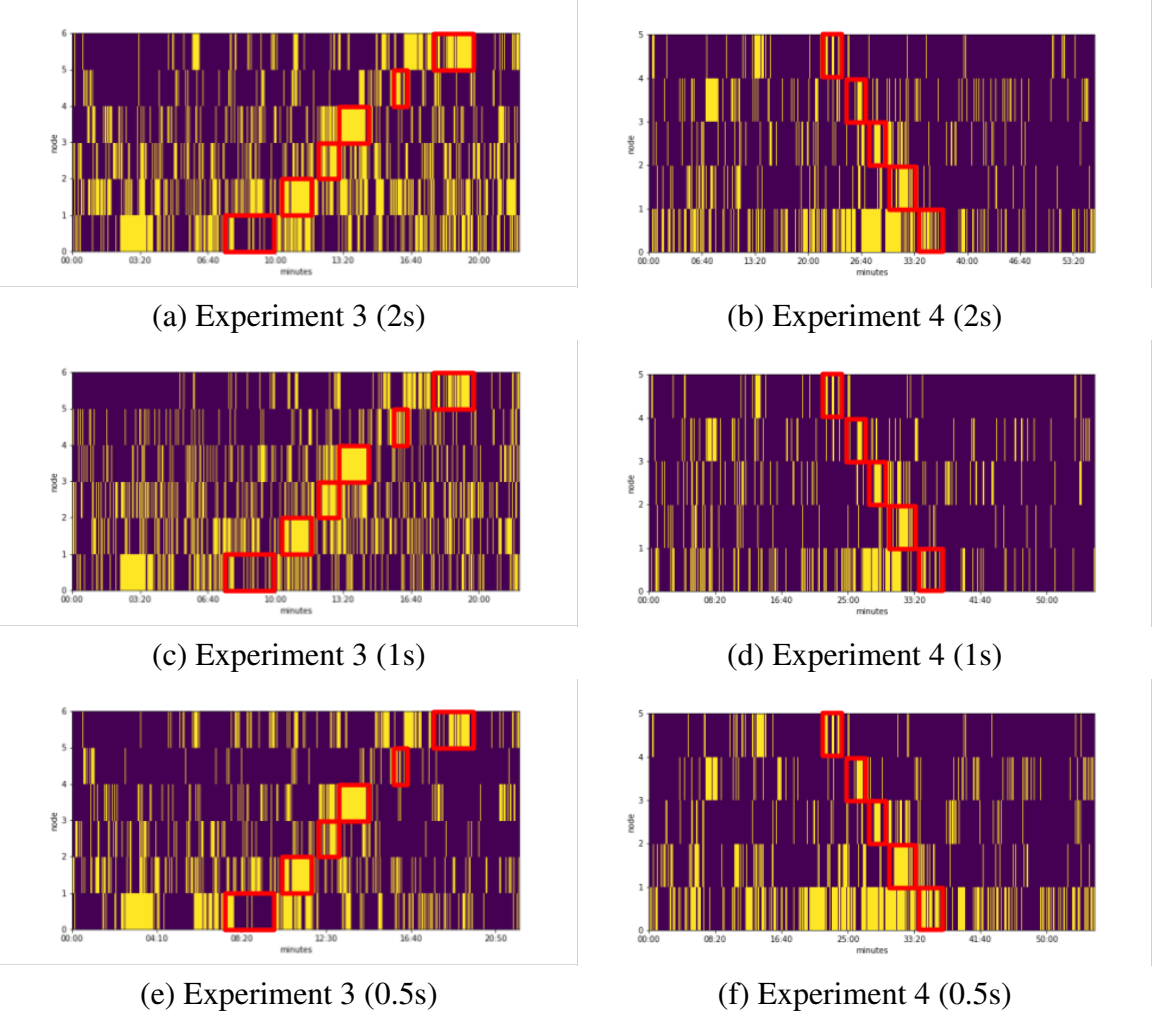
(c) Experiment 1 (1s)

(d) Experiment 2 (1s)

(e) Experiment 1 (0.5s)

(f) Experiment 2 (0.5s)

Figure 4.13. STFT-SVM model for experiment 1 and experiment 2



(a) Experiment 3 (2s)

(b) Experiment 4 (2s)

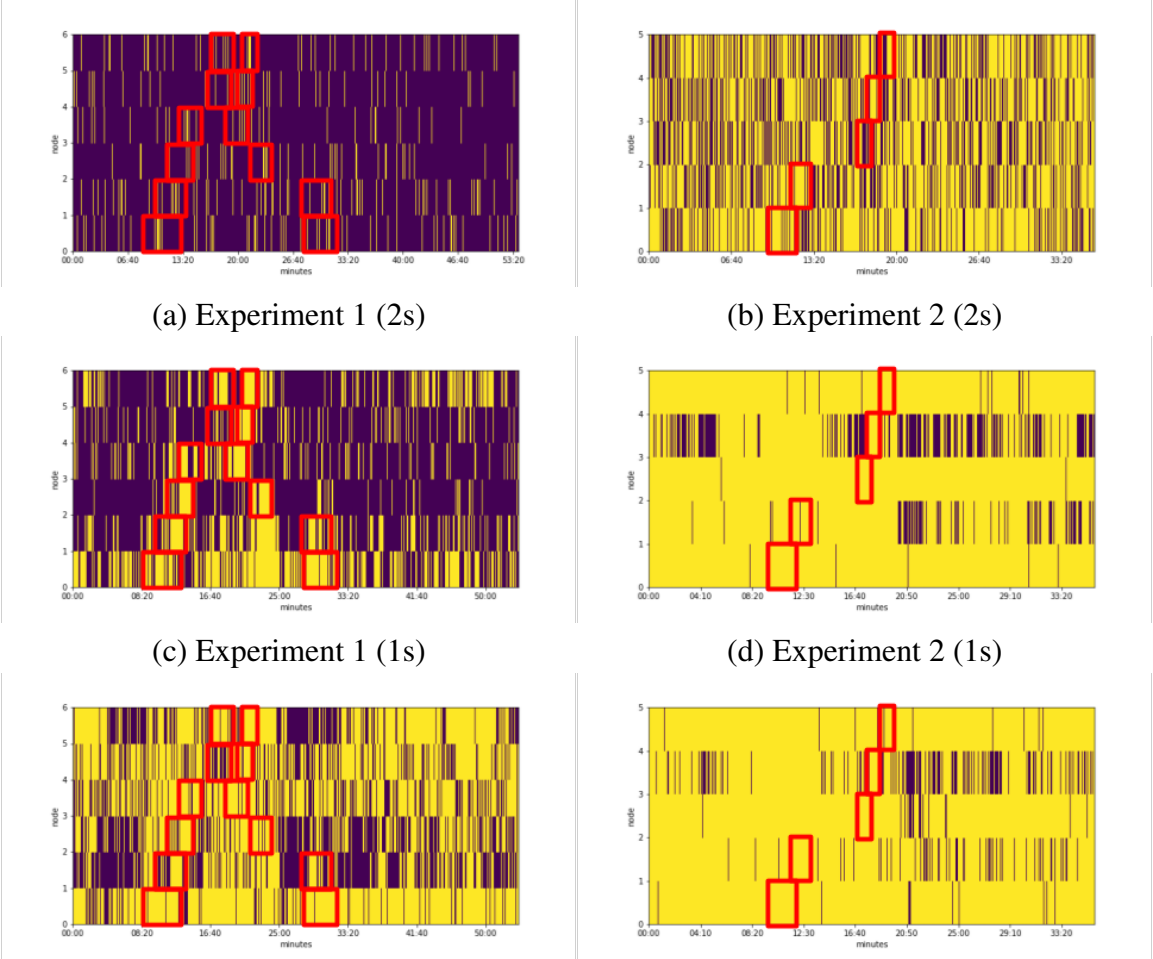
(c) Experiment 3 (1s)

(d) Experiment 4 (1s)

(e) Experiment 3 (0.5s)

(f) Experiment 4 (0.5s)

Figure 4.14. STFT-SVM model for experiment 3 and experiment 4



(a) Experiment 1 (2s)

(b) Experiment 2 (2s)

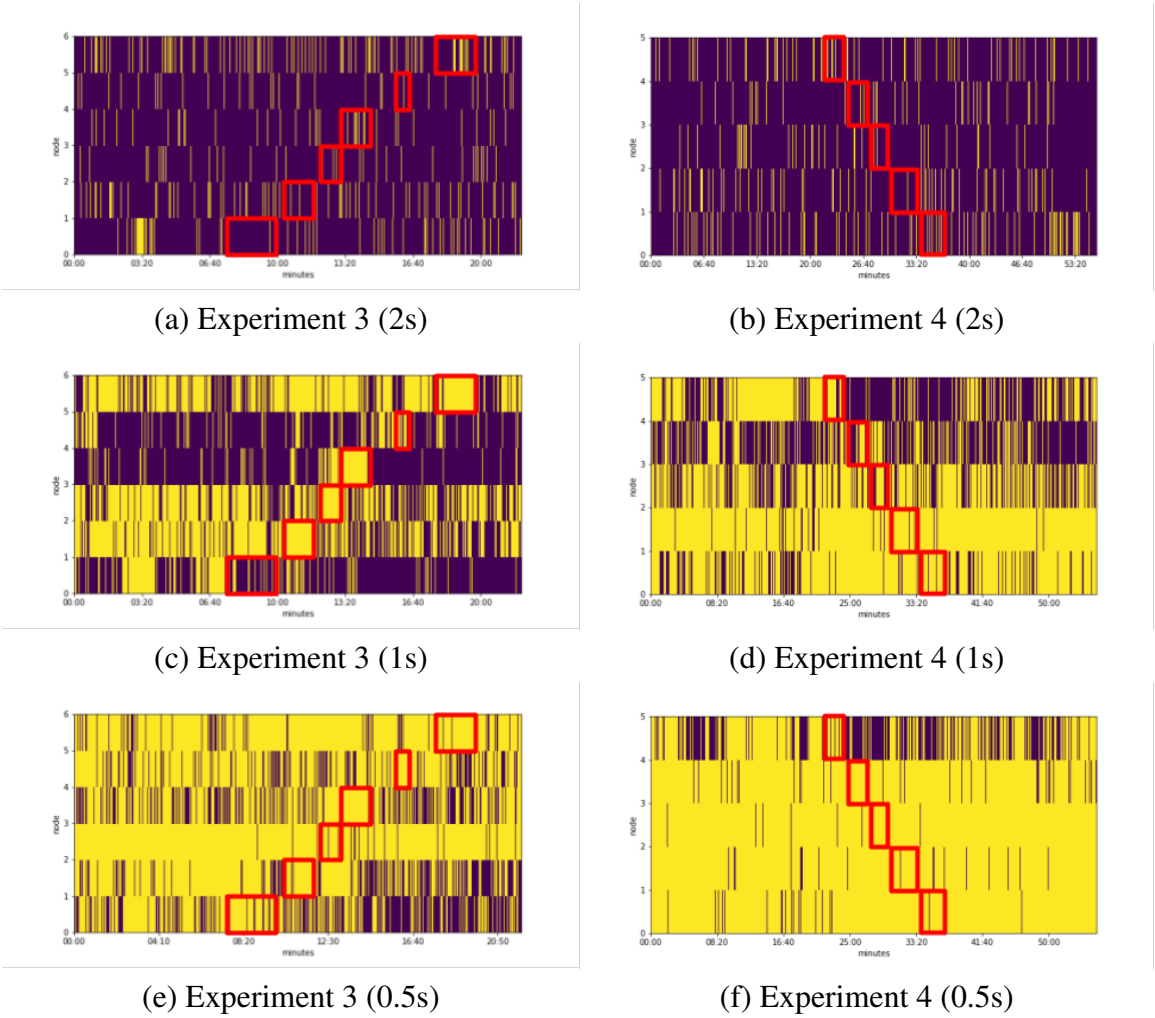
(c) Experiment 1 (1s)

(d) Experiment 2 (1s)

(e) Experiment 1 (0.5s)

(f) Experiment 2 (0.5s)

Figure 4.15. MFCC-CNN model for experiment 1 and experiment 2



(a) Experiment 3 (2s)

(b) Experiment 4 (2s)

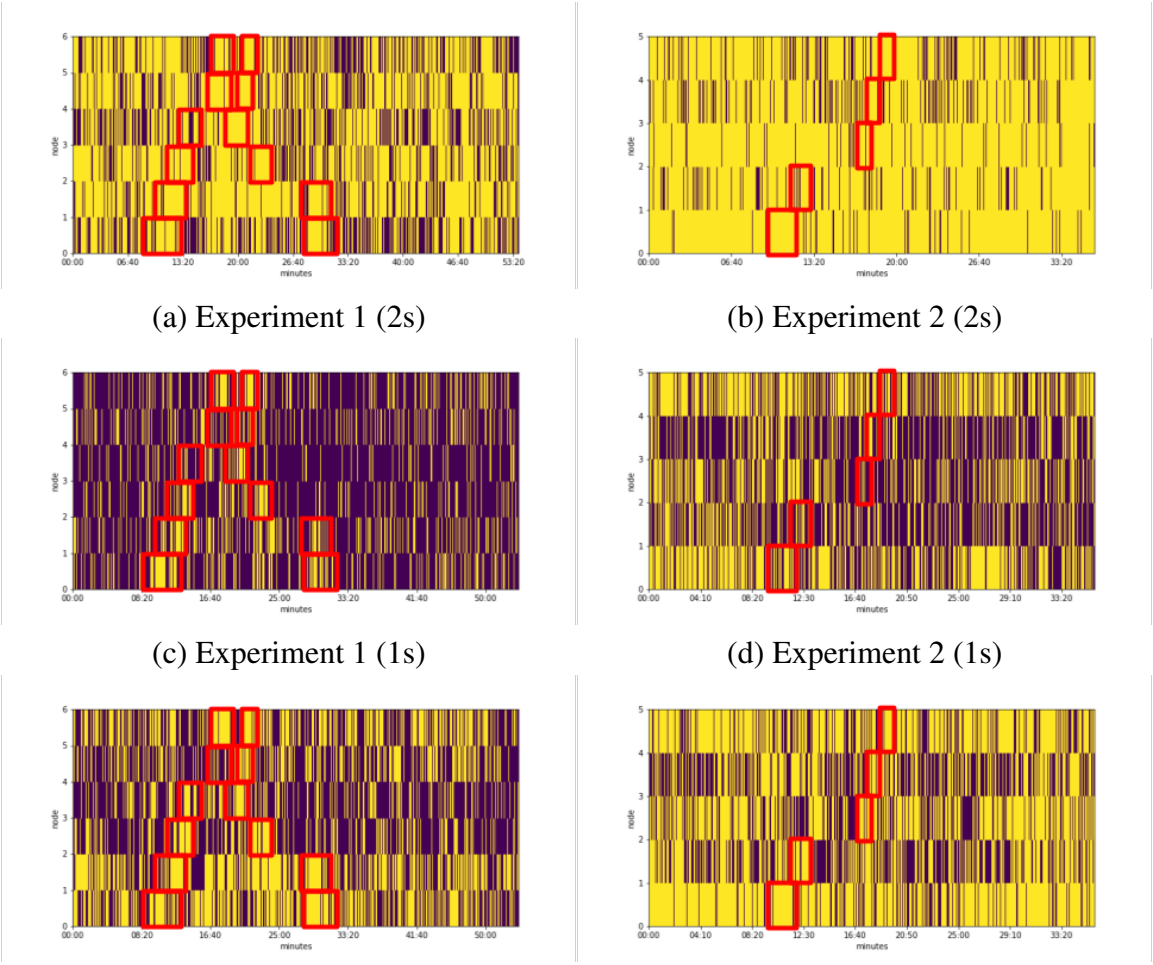
(c) Experiment 3 (1s)

(d) Experiment 4 (1s)

(e) Experiment 3 (0.5s)

(f) Experiment 4 (0.5s)

Figure 4.16. MFCC-CNN model for experiment 3 and experiment 4



(a) Experiment 1 (2s)

(b) Experiment 2 (2s)

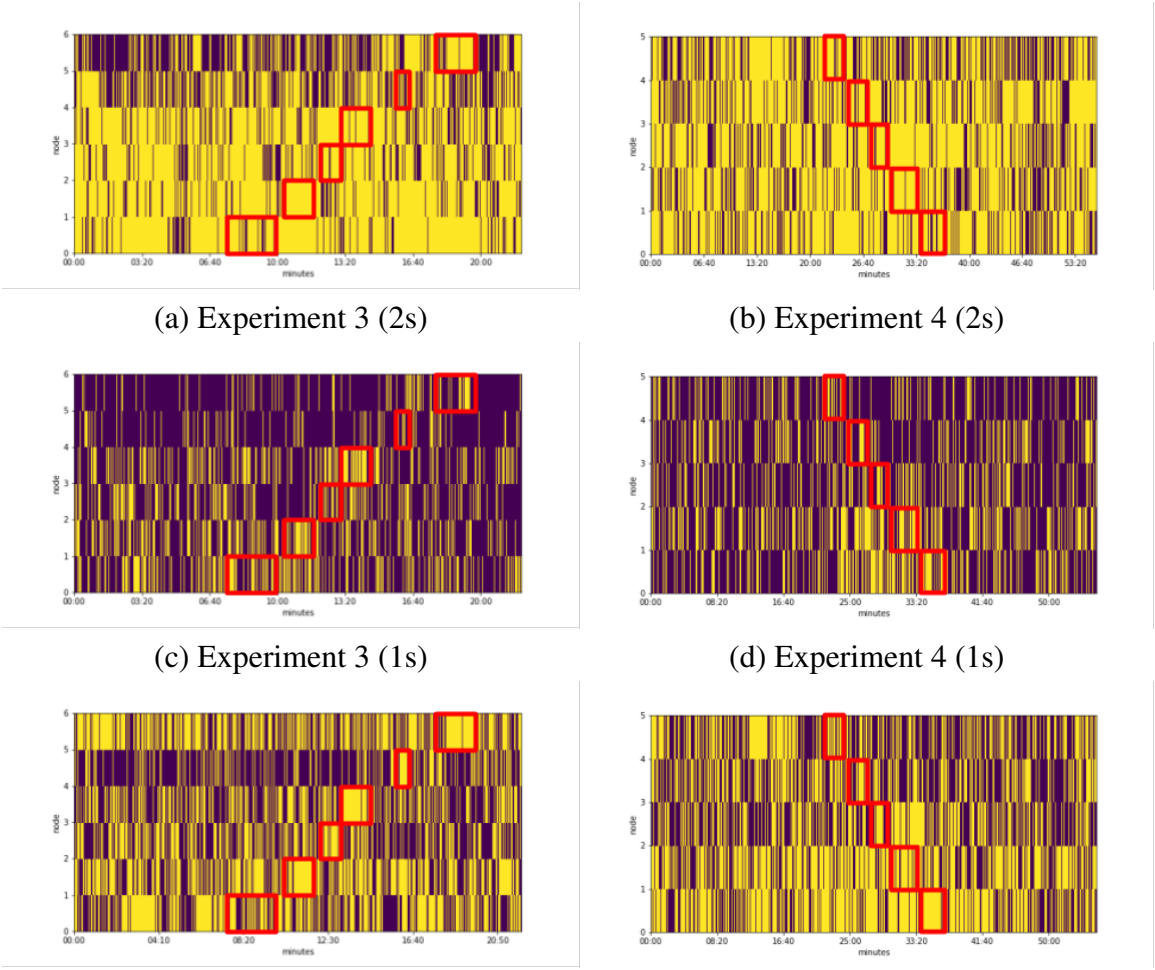
(c) Experiment 1 (1s)

(d) Experiment 2 (1s)

(e) Experiment 1 (0.5s)

(f) Experiment 2 (0.5s)

Figure 4.17. STFT-CNN model for experiment 1 and experiment 2



(a) Experiment 3 (2s)

(b) Experiment 4 (2s)

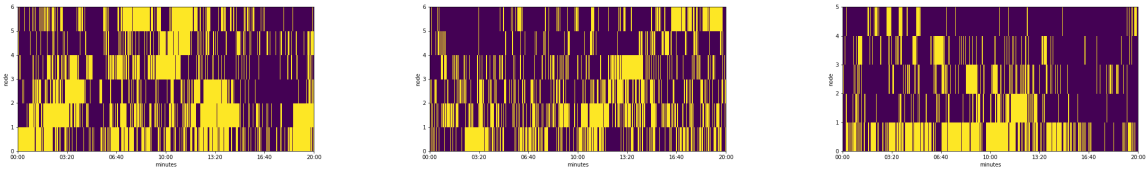
(c) Experiment 3 (1s)

(d) Experiment 4 (1s)

(e) Experiment 3 (0.5s)

(f) Experiment 4 (0.5s)

Figure 4.18. STFT-CNN model for experiment 3 and experiment 4



(a) Experiment 1

(b) Experiment 3

(c) Experiment 4

Figure 4.19. Results of experiment 1, 3 and 4 by STFT-SVM model with the 2-second chunk size

4.6 Near Real-time Simulation

The system is proposed as a near real-time system. Simulations are done for the near real-time environment. Under the near real-time environment, the node runs two processes and repeats them periodically. One process grabs the audio chunk that is either half, one or two seconds. Then the process computes the features of the chunk and puts the feature vector into the shared memory. The other process feeds the feature vector into the detection model and sends the result to the server. Sample data sent to the server is as below:

```
doc = {
    'device': 'node1',
    'record_time': September 27th 2018, 18:14:42.959,
    'detection_latency': 0.41,
    'detection_probability': 0.98,
    'departure_time': September 27th 2018, 18:15:24.103
}
```

Nodes send the probability of a UAV flying around as `detection_probability` along with the time that the recording happened as `record_time`. Users can monitor the probability of a UAV flying around using a dashboard in near real-time as in figure 4.20.

For a near real-time system, providing data with a small latency is important. To support users with performance monitoring, the server collects detection latency and network latency. `detection_latency` is the time taken to compute the features and run a detection model, whose average is 0.25 seconds and constant throughout the process. Also, once the data is sent to the server, the server checks the time of arrival and calculates the network delay. Network latency can also be monitored as in figure 4.21. In the middle of the figure, there is a spike in latency, meaning the data was not sent to the server and the packets are lost for about 12 seconds indicating a bottleneck. Bottlenecks happen when the server handles multiple connections with six nodes. The server manages the connected clients in an array. When the data is transmitted to the server, switching connections every second can have a burden on a single threaded server program, leading to slow

Figure 4.20. Detection dashboard

transmission and bottlenecks as the packets are piled up. This is an issue that should be solved in further research by changing the server code and configuration.

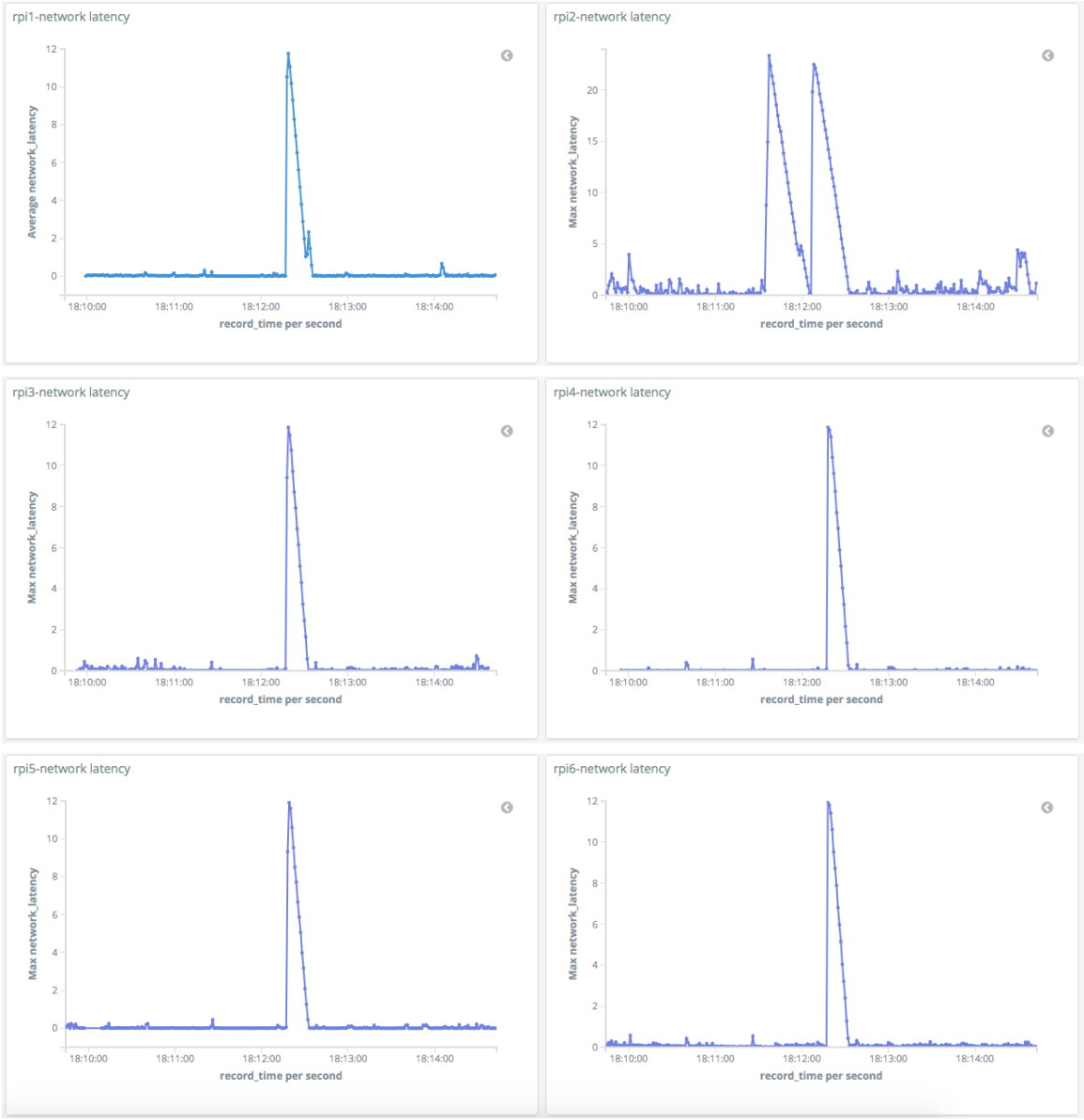


Figure 4.21. Latency dashboard

CHAPTER 5. SUMMARY

This paper introduced a near real-time acoustic UAV detection system with multiple listening nodes using machine learning models. After the data collection phase, SVM and CNN were built with two features, MFCC and STFT. Considering the near real-time system, the features were calculated after cutting the audio stream into chunks of two, one or half seconds. There are four combinations of features and models as well as three versions per combination based on the chunk size, returning twelve models in total. To train SVM, exhaustive search method was used to find the best parameter while CNN was built by selecting the parameters manually.

Four node configurations were devised to find the best way to place six listening nodes. Twelve models were run for each configuration, generating color maps to show the paths of the UAV flying along the nodes. STFT-SVM model showed the path most clearly with the least false negatives with 2-second chunk size. Among the four configurations, the configuration for experiment 3 showed the best results in terms of the distance of detection results on the color maps.

5.1 Limitation

There are limitations regarding modeling and methodology.

5.1.1 Methodology

The research needs to improve the validity for the following reasons.

First, the detectability of each model is not confirmed before testing the four configurations. It should be verified that the models can detect a UAV by testing one node before testing with multiple nodes to find the optimal configuration.

Second, the configurations should be proposed and tested rigorously. The proposed configurations have 50, 75, 100 meters for the radius and 30 degrees for the angle, but only one configuration has 60 degrees for the angle. Missing parameter

combinations can make the research incomplete. Reasons for the parameters and their values to be selected should be clarified.

Third, the test environment was not completely controlled. The training data was collected considering all the factors suggested in chapter 3, whereas the test data was collected on only one case without considering every factor. Factors such as the weather or the UAV hardware were not controlled enough to make the results hard to reproduce.

Last, an evaluation needs to be quantitative, based on numerical analysis. The research compares the models and the configurations by visually measuring differences in the color maps, which makes it hard to evaluate the results.

5.1.2 Modeling

First, the signals had a lot of noises, and were not normalized. Normalization of the audio signals is needed, and noise should also be reduced. As shown in the previous chapter, the detection results were different although the same sounds were recorded because microphones and sound cards have different qualities. There are known ways to normalize signals including scaling, which is a technique to divide the signals by the maximum value of the total audio file. However, this cannot be applied for the real-time environment because the maximum value keeps changing as new signals are fed.

Second, the quality of the test data is not consistent with the training data because of equipment issues. Below are the factors that can affect the results:

- The UAV was tethered to a string. When the UAV became out of range because of wind, the string was pulled and the resistance could have affected the audio signals.
- After several experiments, the UAV was not able to fly. The pilot held the UAV and walked along the nodes instead of flying it.

Third, the models are not strong enough, leading that they cannot differentiate the UAV when the noises are dominant. Strong models can be built by having various noise classes so that the system can classify sounds in detail or by using different models to enable detection of overlapped classes.

Fourth, wireless networking should be configured with more consideration to enable connection when the nodes are far apart.

More work should be done to minimize delays and bottlenecks so that the system can detect UAVs that move fast to the target area.

5.2 Future Work

5.2.1 Methodology

First, adding a preliminary step to find the average detection of a single node can be considered. Node configurations can be conceived with more clarity based on the detection range of a node, providing a foundation to propose configurations.

Second, more research should be done to define a quantitative evaluation method such as the success rate of the system on multiple tests flying the UAV across the boundary.

First, adding a preliminary step to find the average detection of a single node can be considered. Node configurations can be conceived with more clarity based on the detection range of a node, providing foundation to propose configurations.

Second, more research should be done to define a quantitative evaluation method such as the success rate of the system on multiple tests flying the UAV across the boundary.

5.2.2 Modeling

First, research can be done on normalization or noise reduction for real-time signals. Second, multi-class models can be trained with more audio data rather than binary classification. Separate classes for wind, airplanes and birds can be defined for clearer distinctions. Different types of machine learning models can be used to detect multiple classes at the same time.

Additionally, machine learning and rule-based methods can be combined to find which node is closer to the UAV. It is hard to find where the UAV is when a louder UAV approaches and multiple nodes detect it. Utilizing the fact that each sound has its own frequency range, the system can monitor if the UAV is approaching closer by calculating the magnitude of specific frequency range.

Also, more work should be done to make a system with minimum networking barriers by using a faster protocol. Considering that the response time should be more prioritized, the web-based monitoring tool can be simplified by using lighter protocols than Websocket. Using UDP to allow faster transmission even though there can be lost packets can be one of the possible ways to improve the system.

Lastly, a scalable system can be built, which handles more detection results as the system scales out with more nodes, or that streams audio data to make it as a surveillance system.

REFERENCES

[1] Drone Industry Analysis: Market trends & growth forecasts - business insider. [Online]. Available: http://www.businessinsider.com/drone-industry-analysis-market-trends-growth-forecasts-2017-7

[2] D. Joshi. Commercial unmanned aerial vehicle (UAV) market analysis industry trends, companies and what you should know. [Online]. Available: http://www.businessinsider.com/commercial-uav-market-analysis-2017-8

[3] "Prison drones drug smuggling gang jailed." [Online]. Available: http://www.bbc.com/news/uk-42341416

[4] J. Warrick, "Use of weaponized drones by ISIS spurs terrorism fears."

[5] W. Shi, G. Arabadjis, B. Bishop, P. Hill, R. Plasse, and J. Yoder, *Detecting, Tracking, and Identifying Airborne Threats with Netted Sensor Fence*. [Online]. Available: http: //www.intechopen.com/books/sensor-fusion-foundation-and-applications/ detecting-tracking-and-identifying-airborne-threats-with-netted-sensor-fence

[6] J. M. Goppert, A. R. Wagoner, D. K. Schrader, S. Ghose, Y. Kim, S. Park, M. Gomez, E. T. Matson, and M. J. Hopmeier, "Realization of an autonomous, air-to-air counter unmanned aerial system (CUAS)," in *2017 First IEEE International Conference on Robotic Computing (IRC)*, pp. 235–240.

[7] A. R. Wagoner, D. K. Schrader, and E. T. Matson, "Towards a vision-based targeting system for counter unmanned aerial systems (CUAS)," in *2017 IEEE International Conference on Computational Intelligence and Virtual Environments for Measurement Systems and Applications (CIVEMSA)*, pp. 237–242.

[8] L. Belz, "Counter-UAV system from airbus defence and space protects large installations and events from illicit intrusion." [Online]. Available: http://company.airbus.com/news-media/press-releases/airbus-group/ financial_communication/2015/09/ 20150916_airbus_defence_and_space_counter_uav.html

[9] Airbus expands counter-UAV product range with new portable jamming system. [Online]. Available: http://www.airforce-technology.com/news/newsairbus-expands-counter-uav-product-range-with-new-portable-jamming-system-5708956/

[10] DroneShield. [Online]. Available: https://www.droneshield.com/

[11] OpenWorks engineering. [Online]. Available: https://openworksengineering.com/

[12] Skolnik, *Introduction to Radar Systems*. Tata McGraw Hill, google-Books-ID: zlZom9QkjCkC.

[13] M. Benyamin and G. H. Goldman, "Acoustic detection and tracking of a class i UAS with a small tetrahedral microphone array." [Online]. Available: http://www.dtic.mil/docs/citations/ADA610599

[14] . Gven, O. Ozdemir, Y. Yapici, H. Mehrpouyan, and D. Matolak, "Detection, localization, and tracking of unauthorized UAS and jammers," in *2017 IEEE/AIAA 36th Digital Avionics Systems Conference (DASC)*, pp. 1–10.

[15] S. Park, Y. Kim, E. T. Matson, and A. H. Smith, "Accessible synthetic aperture radar system for autonomous vehicle sensing," in *2016 IEEE Sensors Applications Symposium (SAS)*, pp. 1–6.

[16] V. Mitra, C.-J. Wang, and S. Banerjee, "Lidar detection of underwater objects using a neuro-SVM-based architecture," vol. 17, no. 3, pp. 717–731.

[17] L. Wallace, A. Lucieer, and C. S. Watson, "Evaluating tree detection and segmentation routines on very high resolution UAV LiDAR data," vol. 52, no. 12, pp. 7619–7628.

[18] N. Jeong, H. H., and M. E. T., "Evaluation of low-cost lidar sensor for application in indoor uav navigation," in *2018 IEEE SAS*, p. ?

[19] S. Ojha and S. Sakhare, "Image processing techniques for object tracking in video surveillance- a survey," in *2015 International Conference on Pervasive Computing (ICPC)*, pp. 1–6.

[20] A. Rozantsev, "Vision-based detection of aircrafts and uavs," p. 116, 2017.

[21] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," vol. 521, no. 7553, pp. 436–444. [Online]. Available: https://www.nature.com/articles/nature14539

[22] E. E. Case, A. M. Zelnio, and B. D. Rigling, "Low-cost acoustic array for small UAV detection and tracking," in *2008 IEEE National Aerospace and Electronics Conference*, pp. 110–113.

[23] S. Jeon, J. W. Shin, Y. J. Lee, W. H. Kim, Y. Kwon, and H. Y. Yang, "Empirical study of drone sound detection in real-life environment with deep neural networks," in *2017 25th European Signal Processing Conference (EUSIPCO)*, pp. 1858–1862.

[24] H. Lee, Y. Largman, P. Pham, and A. Y. Ng, "Unsupervised feature learning for audio classification using convolutional deep belief networks," in *Proceedings of the 22Nd International Conference on Neural Information Processing Systems*, ser. NIPS'09. Curran Associates Inc., pp. 1096–1104. [Online]. Available: http://dl.acm.org/citation.cfm?id=2984093.2984217

[25] M. Benzeghiba, R. De Mori, O. Deroo, S. Dupont, T. Erbes, D. Jouvet, L. Fissore, P. Laface, A. Mertins, C. Ris, R. Rose, V. Tyagi, and C. Wellekens, "Automatic speech recognition and speech variability: A review," vol. 49, no. 10, pp. 763–786. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167639307000404

[26] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A. r. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath, and B. Kingsbury, "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups," vol. 29, no. 6, pp. 82–97.

[27] G. Tzanetakis and P. Cook, "Musical genre classification of audio signals," vol. 10, no. 5, pp. 293–302.

[28] C. Clavel, T. Ehrette, and G. Richard, "Events detection for an audio-based surveillance system," in *2005 IEEE International Conference on Multimedia and Expo*, pp. 1306–1309.

[29] A. Mesaros, T. Heittola, A. Eronen, and T. Virtanen, "Acoustic event detection in real life recordings," in *2010 18th European Signal Processing Conference*, pp. 1267–1271.

[30] K. J. Piczak, "Environmental sound classification with convolutional neural networks," in *2015 IEEE 25th International Workshop on Machine Learning for Signal Processing (MLSP)*, pp. 1–6.

[31] R. Serizel, V. Bisot, S. Essid, and G. Richard, "Acoustic features for environmental sound analysis," in *Computational Analysis of Sound Scenes and Events*. Springer, Cham, pp. 71–101. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-319-63450-0_4

[32] J. Salamon, C. Jacoby, and J. P. Bello, "A dataset and taxonomy for urban sound research," in *Proceedings of the 22Nd ACM International Conference on Multimedia*, ser. MM '14. ACM, pp. 1041–1044. [Online]. Available: http://doi.acm.org/10.1145/2647868.2655045

[33] K. J. Piczak, "Environmental sound classification with convolutional neural networks," in *2015 IEEE 25th International Workshop on Machine Learning for Signal Processing (MLSP)*, pp. 1–6.

[34] A. Mesaros, T. Heittola, and T. Virtanen, "TUT database for acoustic scene classification and sound event detection," in *2016 24th European Signal Processing Conference (EUSIPCO)*, pp. 1128–1132.

[35] T. Heittola, E. akr, and T. Virtanen, "The machine learning approach for analysis of sound scenes and events," in *Computational Analysis of Sound Scenes and Events*. Springer, Cham, pp. 13–40. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-319-63450-0_2

[36] P. Cano, E. Batle, T. Kalker, and J. Haitsma, "A review of algorithms for audio fingerprinting," in *2002 IEEE Workshop on Multimedia Signal Processing.*, pp. 169–173.

[37] I. T. Jolliffe, "Principal component analysis and factor analysis," in *Principal Component Analysis*, ser. Springer Series in Statistics. Springer, New York, NY, pp. 115–128. [Online]. Available: https://link.springer.com/chapter/10.1007/978-1-4757-1904-8_7

[38] Y. Xu, Q. Huang, W. Wang, P. Foster, S. Sigtia, P. J. B. Jackson, and M. D. Plumbley, "Unsupervised feature learning based on deep models for environmental audio tagging," vol. 25, no. 6, pp. 1230–1241.

[39] Raspberry pi 3 model b. [Online]. Available: https://www.raspberrypi.org/products/raspberry-pi-3-model-b/

[40] Audio injector. [Online]. Available: http://www.audioinjector.net/

[41] Long-range dual-band AC1200 wireless USB 3.0 wi-fi adapter. [Online]. Available: https://www.amazon.com/Long-Range-Dual-Band-Wireless-External-Antennas/dp/B00VEEBOPG

[42] WNR2000v5 n300 wireless router. [Online]. Available: https://www.netgear.com/support/product/WNR2000v5.aspx

[43] AlsaProject. [Online]. Available: https://www.alsa-project.org

[44] Virtualenv virtualenv 16.0.0 documentation. [Online]. Available: https://virtualenv.pypa.io/en/stable/

[45] PyAudio: PortAudio v19 python bindings. [Online]. Available: https://people.csail.mit.edu/hubert/pyaudio/

[46] PortAudio - an open-source cross-platform audio API. [Online]. Available: http://www.portaudio.com/

[47] Librosa. [Online]. Available: https://librosa.github.io/

[48] scikit-learn: machine learning in python scikit-learn 0.19.1 documentation. [Online]. Available: http://scikit-learn.org

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

[50] BCM43438 details - cypress semiconductor | datasheets. [Online]. Available: https://www.datasheets.com/en/details/bcm43438-cypress-semiconductor-85839902

[51] Parrot AR.drone 2.0 elite edition. [Online]. Available: https://www.parrot.com/global/drones/parrot-ardrone-20-elite-edition

[52] V. Tiwari, "MFCC and its applications in speaker recognition."

[53] L. R. Rabiner and R. W. Schafer, "Introduction to digital speech processing," vol. 1, no. 1, pp. 1–194. [Online]. Available: http://dx.doi.org/10.1561/2000000001

[54] J. B. Allen and L. R. Rabiner, "A unified approach to short-time fourier analysis and synthesis," vol. 65, no. 11, pp. 1558–1564.

[55] L. Deng, G. Hinton, and B. Kingsbury, "New types of deep neural network learning for speech recognition and related applications: an overview," in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 8599–8603.

[56] G. Guo and S. Z. Li, "Content-based audio classification and retrieval by support vector machines," vol. 14, no. 1, pp. 209–215.

[57] V. N. Vapnik, "An overview of statistical learning theory," vol. 10, no. 5, pp. 988–999.

[58] Elasticsearch: RESTful, distributed search & analytics | elastic. [Online]. Available: https://www.elastic.co/products/elasticsearch

[59] Kibana. [Online]. Available: https://www.elastic.co/products/kibana

[60] Tornado web server tornado 5.1.1 documentation. [Online]. Available: https://www.tornadoweb.org/en/stable/

[61] I. Fette and A. Melnikov, "The WebSocket protocol." [Online]. Available: http://www.rfc-editor.org/info/rfc6455