UAV Payload Identification With Acoustic Emissions And Cell Phone Devices

THESIS

Hunter G. Doster, 2d Lieutenant, USAF

AFIT-ENG-MS-22-M-026

AFIT-ENG-MS-22-M-026

UAV Payload Identification With Acoustic Emissions And Cell Phone Devices

THESIS

Presented to the Faculty
Department of Electrical and Computer Engineering
Graduate School of Engineering and Management
Air Force Institute of Technology
Air University
Air Education and Training Command
in Partial Fulfillment of the Requirements for the
Degree of Master of Science in Computer Engineering

Hunter G. Doster, B.S. Computer Engineering
2d Lieutenant, USAF

March 24, 2022

AFIT-ENG-MS-22-M-026

UAV Payload Identification With Acoustic Emissions And Cell Phone Devices

THESIS

Hunter G. Doster, B.S. Computer Engineering
2d Lieutenant, USAF

Committee Membership:

Barry E. Mullins, Ph.D., P.E.
Chair

Timothy H. Lacey, Ph.D., CISSP
Member

Robert F. Mills, Ph.D.
Member

AFIT-ENG-MS-22-M-026

Abstract

This thesis explores the ability to use cell phone devices for Unmanned Aerial Vehicle (UAV) acoustic payload detection. Past researchers demonstrated the ability to use UAV acoustic signatures to determine whether a UAV carries a payload and the weight of that payload. The experiments in past research were conducted at close range with high-quality microphones. This research expands the field of study by testing acoustic payload detection using cell phone devices and at far range. The Department of Defense (DoD) is particularly interested in acoustic payload detection due to rising security threats from UAVs. Although these detection systems increase security, the ever on-going arms race leads to counter techniques which make these detection systems ineffective. Therefore, there is a need for new drone detection systems which use new technology. Acoustic emissions are a unique property all drones expel, and these emissions provide new stimulus for drone detection systems. An acoustic drone detection system does not require line-of-sight and is difficult to spoof, so an acoustic drone detection system which identifies payload weight using cell phones would prove useful.

Cell phones are commonplace worldwide. Due to this fact, there is a growing desire to use cell phones for hazard detection. The ability to use a common cell phone to detect a far off hazardous UAV would improve security in many contested environments.

This research develops the prototype HurtzHunter to demonstrate acoustic payload detection with cell phone devices to collect UAV acoustic emissions, then uses the emissions to train an AI capable of UAV payload classification. The HurtzHunter system tests acoustic payload detection with 7 different recording devices at 7 m, 10 m, 20 m, 30 m, 40 m, 50 m, 75 m, and 100 m ground distance from the drone. At each distance, the experiment runs 6 flights each with a unique payload attached to the drone. 80% of the acoustic emissions train a Support Vector Machine (SVM), and 20% tests the SVM.

The methodology in this research shows the HurtzHunter design achieves an 82.81 - 99.93% payload prediction accuracy based on the configuration. In short, this research provides novel insight into the maximum range for UAV payload detection using acoustic emissions, and provides insight into the ability to use cell phone devices for payload detection.

Acknowledgements

Thank you to Jesus. Without you, none of this would be possible.

Thank you to my wife Lauren. Without your support, I would never accomplish anything.

Thank you to my advisor Dr. Mullins. You are the best faculty member at AFIT. Few come close to the level of excellence, care, and support you give towards your work and students.

Thank you to my committee members Dr. Mullins, Dr. Lacey, and Dr. Mills for giving their time and advice during my research.

Thank you to my sponsor AFRL/RYAA for providing all the resources and help needed to accomplish this research. None of this would be possible without your support.

Thank you to AFRL/RYAR for flying my experiments and allowing me to accompany you to the Air Force Bombing range.

Hunter G. Doster

Table of Contents

List of Figures

List of Tables

Table Page

List of Acronyms

Abbreviation		Page
ADC	Analog to Digital Conversion	11
AFRL	Air Force Research Lab	2
AGL	Above Ground Level	75
AI	Artificial Intelligence	9
CDC	Center for Disease Control	85
CSVM	cubic SVM	87
dB	decibels	13
DCT	Discrete Cosine Transform	10, 14
DFT	Discrete Fourier Transform	13
DoD	Department of Defense	iv
DSP	Digital Signal Processing	10
FFT	Fast Fourier Transform	10, 14
GP4	Google Pixel 4	4, 140
GS20	Samsung Galaxy S20	4, 140
GS8	Samasung Galaxy S8	4, 140
H	hyperplane	27
IDE	Integrated Development Environment	50
IDFT	Inverse Discrete-Time Fourier Transform	22

Abbreviation		Page
IED	Improvised Explosive Device	1
IoT	Internet of Things	2
iP11P	iPhone 11 Pro	4, 140
LPCC	Linear Predictive Cepstral Coefficients	32
LSVM	linear SVM	89
MFCCs	Mel-Frequency Cepstrum Coefficients	7
ML	Machine Learning	4
NIOSH	National Institute of Occupational Safety and Health	85
QSVM	quadratic SVM	89
RF	Radio Frequency	32
RPM	Revolutions Per Minute	8
RVMP	RodeVideoMicPro	3, 140
SV	support vectors	27
SVM	Support Vector Machine	v, 4, 26
UAV	Unmanned Aerial Vehicle	iv, 1

UAV Payload Identification With Acoustic Emissions And Cell Phone Devices

I. Introduction

1.1 Overview and Background

On November 4, 1780 Joseph-Michel and Jacques-Etienne flew the first unmanned aircraft in history. The aircraft was a hot air balloon which rose to 1,000 meters in altitude before landing about a mile away from the take-off point [1]. This event sparked a technological fire focused on unmanned aircraft development, and this fire burned its way into nearly every sector of society. Today, there are over 1.25 million UAVs in operation [2]. The price of a UAV, or drone, ranges from \$30 to over \$35,000, and the public can buy these remotely with the click of a button or at their local store [3]. Today, people use them for recreation and industry. For example, Amazon uses drones to deliver packages with their new system Prime Air [4]. This technology brings much excitement, but as with any new technology, drones bring new security threats to civilian and military settings.

Due to the increasing affordability of these drones, US adversaries are using them for military tactics. In 2017, Syrian forces found an ISIS drone Improvised Explosive Device (IED) plant. The terrorist organization used captured US drones to develop their own Unmanned Aerial Vehicle (UAV) and turn them into IEDs [5]. This example is just the tip of the iceburg, many more adversaries are developing drone technology and using them in military tactics. Beyond the military, illegal drug traders use small UAVs to ship drugs across the US southern boarder [6]. In response to these threats, there is an increasing need for drone detection systems. Currently, there are drone detection systems using RADAR, visual recognition, and radio frequency. Another promising solution for droned detection uses acoustic emissions. Small drones emit unique sounds while they fly, and this sound directly corresponds to features of the drone such as model and attached payloads.

1.2 Problem Statement

Current drone detection platforms focus on RADAR, image recognition, LiDAR, or Radio Frequency. Although these platforms are useful, they can be spoofed, and some are limited to direct line of sight. Therefore, the need for a new zero-touch, zero line-of-sight detection system would be useful in drone methods. A solution to this problem is an acoustic UAV detection system. Acoustics are a unique quality all drones emit. They are difficult to spoof in comparison to other methods, and acoustics do not require line of sight. An effective payload detection system would allow the identification of a hazardous drone when other methods do not. Such a capability would play an integral role in an drone detection system. Therefore, this research focuses on the ability to detect drone payloads using acoustic emissions.

Previous work demonstrates the ability to detect the presence of a drone, as well as the payload carried by a drone using acoustic emissions. Payloads were identified at close range using high-quality microphones and a laptop [7][8]. This research expands the field by testing acoustic detection at far range with cell phone devices. The Air Force Research Lab (AFRL) is interested in the ability to use Internet of Things (IoT) devices for acoustic drone detection. This research proposes a prototype which uses cell phone devices to detect payload weight from 7 m to 100 m from the drone. Cell phones use microphones commonly found on many IoT devices, so the prototype in this research can be deployed on many IoT device for payload detection.

1.3 Research Goals

The goal of this research is to demonstrate an acoustic payload detection method using cell phone devices and determine the maximum range of acoustic payload detection. Previous research successfully characterized payload weight with acoustic emissions at 7 m ground distance from the drone. Past research used the RodeVideoMicPro (RVMP) and a MacBook Pro to record these acoustic emissions [7][8]. This research expands the field of study by using acoustic emissions to characterize payload weight and with cell phone devices at a range of 7 - 100 m from the drone.

The work in this research is relevant because it develops a prototype which tests acoustic detection at long range using cell phones. Cell phones are commonplace world wide, so they are present in most contested environments. By using recording devices already found in an environment, new recording devices would not need to be deployed, allowing for drone detection in previously unreachable environments. In other words, if the US military cannot deploy microphones in an adverse environment, they would be able to use cell phones already in that environment for drone detection.

This research seeks to answer the following questions:

- Can cell phones detect UAV payload weight with sound?
- What is the maximum range of acoustic UAV payload detection?
- Does the cell phone model affect detection accuracy?
- Does a high-quality microphone perform better than a cell phone for acoustic payload detection?
- Does the manufacturing process for a specific recording device affect detection accuracy?

1.4 Hypothesis

This research hypothesizes that cell phones are able to determine the weight of a payload carried by a UAV from acoustic emissions at up to 100 m away ground distance from the UAV. This research develops the drone detection system called HurtzHunter. HurtzHunter aims to provide a UAV acoustic detection method using cell phone devices. This research uses HurtzHunter to determine the maximum detection range for acoustic emissions and compare prediction quality between recording devices. Seven recording devices collect acoustic emissions from a hovering drone at increasing distances with different payloads attached. The collected acoustic data trains and tests a Machine Learning (ML) algorithm and the results are used to support this hypothesis.

1.5 Approach

This research develops HurtzHunter, a drone payload detection system using acoustic emissions. This prototype uses 7 recording devices, 1 quad-copter drone, and a ML algorithm. The recording devices are the three Samasung Galaxy S8 (GS8), one Samsung Galaxy S20 (GS20), one Google Pixel 4 (GP4), one iPhone 11 Pro (iP11P), and one RVMP. The drone is the Mavic Air 2. The ML algorithm is the Support Vector Machine (SVM). The microphones record acoustic sound while the drone hovers for 1 min. The microphones collect acoustics for 48 flights in total. Flights are conducted at 7 m, 10 m, 20 m, 30 m, 40 m, 50 m, 75 m, and 100 m away from the microphones, and at each distance the drone performs 6 flights, each with a different payload weight. The payload weights are 56 g, 112 g, 168 g, 224 g, and 280 g. HurtzHunter uses the sound data from the microphones to train and test an SVM. This research uses the test results to answer the five questions Section 1.3 Research Goals lists.

The experimental data was taken in August at the Air Force Bombing range in Avon Park, FL. A pilot from AFRL/RYAR flew the Mavik Air 2 for all 48 flights over the course of three days.

1.6 Assumptions and Limitations

This research is bounded by the following assumptions and limitations:

1. The experiment in this research is conducted on the flight line at the Air Force Bombing Range in Avon Park, FL. The setting is an open area with no structures nearby, so there is minimal sound interference from structures. However, the setting has the following uncontrollable ambient sounds: cars, aircraft, birds, wind, and insects such as cicadas. The weather during experimentation is very hot with temperatures greater than 90 degrees F and humidity near 100%. The hot weather may cause the drone to need additional thrust for lift affecting the acoustic emissions for each payload. This research does not include experiments run in different weather conditions to evaluate this effect.

2. This research is limited to one drone for testing. Different drones may render different acoustic signatures, and this research does not explore the effect of different UAV platforms on payload prediction performance.

3. This research is limited to one repetition for the experiment Chapter 4 describes. There was limited flight time on the Air Force Bombing range, and only one repetition was accomplished One repetition limits the statistical analysis capable in this research.

1.7 Contributions

Previous work accomplished UAV payload weight identification at close range (∼7 m) using a high-quality microphone and MacBook pro as recording devices [7][8]. This research expands the body of knowledge by performing acoustic payload detection from 7 - 100 m ground distance from the UAV with cell phone devices. This research provides evidence to indicate an optimal range for acoustic payload detection and the difference in accuracy between a high-quality microphone and 4 different cell phone devices. The cell phone devices use small microphones commonly found on many IoT devices, and the results provide evidence for the detection capability of many IoT devices using similar recording utilities. Lastly, this research assesses the impact of manufacturing on payload prediction. The experiment uses seven total recording devices for acoustic payload detection. Three of these devices are the same model cell phone. Therefore, this research compares the prediction results between devices of the same make and model to assess the impact of manufacturing on acoustic payload prediction.

1.8 Thesis Overview

There are six chapters in this thesis. Chapter 2 gives an overview of UAVs, then goes into digital signal processing and the acoustic features from the UAV which are used for payload detection, then discusses the ML this research uses for payload detection, and finally discusses prior research in this field of study. Chapter 3 describes the design of the HurtzHunter prototype created in this research for acoustic payload detection. Chapter 4 details the experiment in this research which tests the HurtzHunter prototype. Chapter 5 discusses the results of the experiment. Finally, Chapter 6 summarizes the research, discusses future work for this field of study and proposes an optimal design choice for acoustic payload prediction.

II. Background and Literature Review

2.1 Overview

This chapter provides background knowledge of UAVs, Mel-Frequency Cepstrum Coefficients (MFCCs), SVMs, and previous research done for acoustic UAV detection and classification. Section 2.2 describes what a UAV is, what they are used for, and why drone detection systems are needed. Section 2.3 introduces and explains the MFCC feature extraction technique and what it is used for. Section 2.4 describes SVMs. Section 2.5 discusses the related research to this field of study, and Section 2.6 discusses the contribution this research makes to that field of study.

2.2 Unmanned Aircraft Systems

In 2019, the FAA estimated there were 1.25 million UAVs in operation and expected the amount to triple by 2023 [2]. A UAV, also known as an UAV or drone, is defined as an unmanned aircraft which is controlled remotely. Drones range in price from $30 to $35,000 and are becoming common use for many industries [3]. For example, Amazon is conducting done delivery in their new Prime Air Program [9]. Drone diffusion into society has brought fun and excitement, but it has also brought new threats. The malicious use of drones poses significant privacy and security threats to civilian and military settings [4].

In 2011, the Iran military claimed a US drone flew into restricted airspace and used drone techniques to take over the US RQ-170 Sentinel drone and land it on Iran soil. Drones have also been used in the illegal drug trade and in terrorist activity [4]. For this reason, drone detection systems are needed to mitigate risks. Researchers have accomplished drone detection and classification using RADAR, visual recognition, radio frequency, and acoustic emissions, but more capabilities need to be developed

[7]. The particular area of study, which the research in this thesis focuses on, is the use of drone acoustic emissions for payload identification.

2.2.1 Lift and Weight Relationship

There are many different types of drones such as fixed and non-fixed wings. The experiments in this research focuses on non-fixed copter models. A copter model generally has 4 to 8 circular propellers on top of the drone which spin to create lift [10]. Figure 1 displays the Mavic Air 2 quad copter UAV model.

Figure 1: Mavic Air 2 Quad Copter [11]

Based on the weight of the UAV and the payload carried, the motors and propellers have to generate different levels of thrust to fly [7]. For heavier loads, the motors and propellers rotate at faster rates than lighter loads. The rate of rotation is often measured in Revolutions Per Minute (RPM). The propeller's RPM is the main contributor to sound emitted by a drone. Since the RPM is correlated to lift, the sound can be used to identify features of the drone [7].

The fundamental sound a drone emits is referred to as the pitch. The pitch fluctuates during flight as the work needed to maintain flight changes. Wind conditions, payload weight, flight pattern and other factors contribute to the change in pitch [7]. There are many techniques to digitally extract drone features from the sound a drone emits for classification. One of the most common feature extraction processes are the MFCCs [12].

2.3 Mel-Frequency Cepstrum Coefficients

MFCCs are the most broadly used audio recognition technique [12]. MFCCs convey different frequency components of an acoustic signal [7]. In 1980, Davis and Mermelstein designed MFCCs to digitally describe how the human phonetic system perceives an audio signal [13]. Today, systems use MFCCs for wireless audio sensor networks, multi-media retrieval, detecting biodiversity metrics in natural environments, terrorist threat detection, home invasion, and much more [12].

MFCCs are a series of coefficients whose values give specific information of what is present in a sound. For example, Artificial Intelligence (AI) uses MFCCs to digitally perceive what a person is saying [12]. Figure 2 shows the process for creating MFCCs.

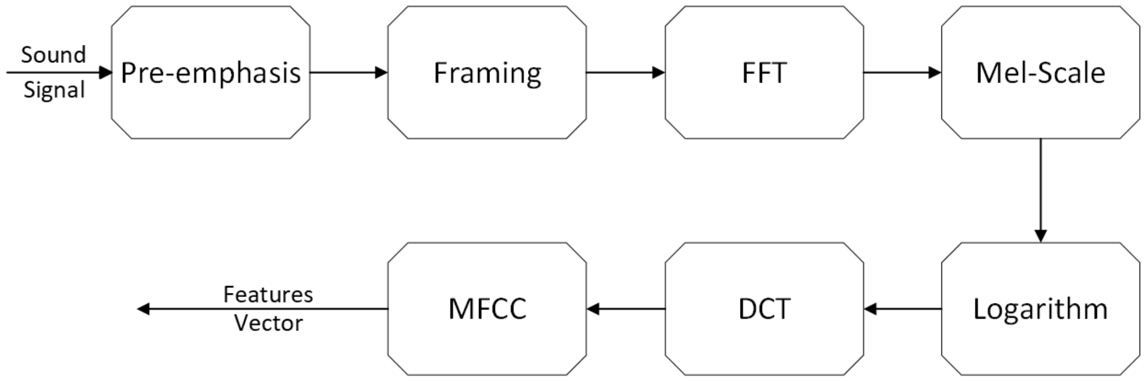


Figure 2: MFCC Generation Process [7]

The rest of this section describes the process seen in Figure 2 in great detail. To help understand, here is a general description of the process. First, acoustic emissions are collected by a recording device, this audio is converted to a digital form. Figure 2 calls this step Pre-emphasis because different microphones prepare the acoustic emissions in different ways. Once in digital form, the acoustic data is divided into sound constant frames. The frequencies in a sound constant frame do not change. To explain, think of an audio recording of someone saying the phrase ”The Matrix”. Each letter in this phrase is a different sound. So, dividing the audio recording of this phrase into sound constant frames creates 9 short audio recordings each containing the sound of a letter: ’t’, ’h’, ’e’, ’m’, ’a’, ’t’, ’r’, ’i’, and ’x’. Then a Fourier Transform (in this case the Fast Fourier Transform (FFT)) places each frame on a spectrum in the frequency domain. After, the spectrums are then mel-scaled to reflect perceptual changes in frequency rather than just any change in frequency. Next, the logarithm of the amplitude is taken to convert the loudness of each frequency in a perceptually relevant way. Lastly, the Discrete Cosine Transform (DCT) extracts MFCCs from each frame.

2.3.1 Digital Signal Processing

MFCC extraction is a type of Digital Signal Processing (DSP) DSP is the technique of digitally representing a signal. DSP has countless uses such as removing interference or noise from a signal, providing methods for long distance communication, creating music and more. [14].

2.3.2 Convolution

A signal is any variable that carries or contains information which can be conveyed, displayed, or manipulated [14]. The sounds we hear are a type of signal. Signals carry many attributes, one of which is frequency. Frequency is a rate of oscillation measured in Hz. Each signal is comprised of one or more frequencies that give the signal its distinct sound. The process of frequencies combining to form a signal is called convolution [14]. In DSP, signal are manipulated through convolution to generate specific sound. Reverse convolution, called deconvolution, is the opposite process of convolution. It breaks down a signal into its different components. MFCC extraction is essentially a deconvolution process to extract signal components correlated to human speech.

The first step in performing DSP operations such as deconvolution is to record a signal and convert it to a digital format.

2.3.3 Analog to Digital Conversion

Analog to Digital Conversion (ADC) is the process of converting a recorded signal to a digital format. All sounds in nature are found in their analog form. Sound capturing devices, such as a microphone, first capture the analog form of a signal [14], then an ADC converts it to a digital format. An analog signal is continuous while the digital signal is discrete. In some cases such as live music, the analog signal is sufficient for use, but other cases such as ML need the digital form. [14]. This need is because only the discrete form of a signal can be represented in bits. Thus, the first step in DSP is converting the signal from analog to digital.

Typically, a system uses a microphone and ADC to create the digital signal. Sometimes, systems consider the microphone and ADC as different devices, but most microphones include the ADC in them. As shown in Figure 3, when a sound reaches a microphone, the physics of the microphone gather the sound in analog form. The quality of the microphone determines the accuracy of the recorded analog signal. Once recorded, the ADC samples the analog signal at some rate, quantizing each sample into a binary number for digital representation [14]. Then, DSP uses the binary representation for processes such as deconvolution.

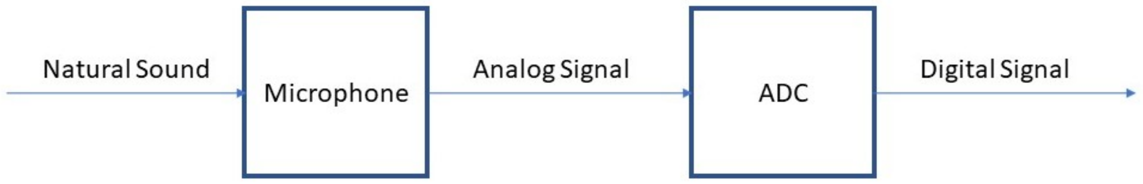


Figure 3: Analog To Digital Conversion

Microphone and ADC features such as filters, sample rate, and bit size affect the quality of the digital signal. Higher quality systems are always better for feature extraction algorithms such as MFCCs. However, higher quality systems are larger in size and cost more money. Determining the proper microphone and ADC combination is an optimization problem that changes for each situation. After generating the digital signal, DSP uses the frequency domain to extract features such as MFCCs.

2.3.4 The Frequency and Time Domain

The frequency and time domain are essential for DSP. The time domain shows the strength of a signal over time. The strength is referred to as the pressure of the signal and is often measured in volts. The pressure of the signal fluctuates over time according to the signal behavior. Figure 4 displays a signal recording displayed in the time domain of someone saying the letter “s”.

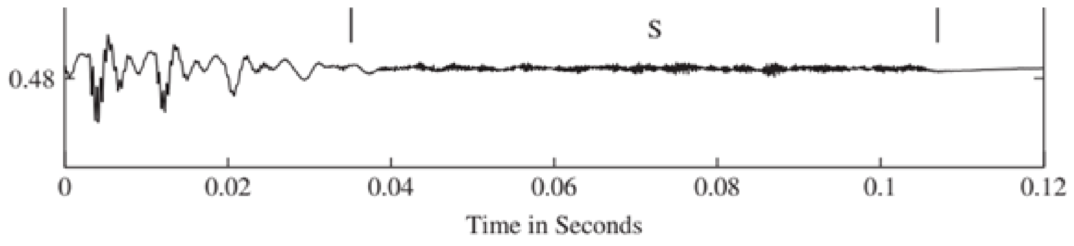


Figure 4: Time Domain Representation Of A Signal [10]

While the time domain shows how a signal behaves as time passes, it does not convey the relevant frequencies found in the signal. It is here that the frequency domain comes into play. When a signal is represented in the frequency domain, the actual frequencies found in the signal are on the x-axis, and the power/magnitude, often expressed in decibels (dB), of each frequency is on the y–axis [14].

To convert the digital signal from the time domain to the frequency domain, a transformation is needed. The most common way to convert a signal to the frequency domain is the through the Discrete Fourier Transform (DFT) [14]. After taking the DFT of a signal, the graphical representation of the signal is called the spectrum because it displays the spectrum of frequencies present in the signal [14]. Figures 5 and 6 show an audio recording of middle ”C” on the piano in the time domain and frequency domain.

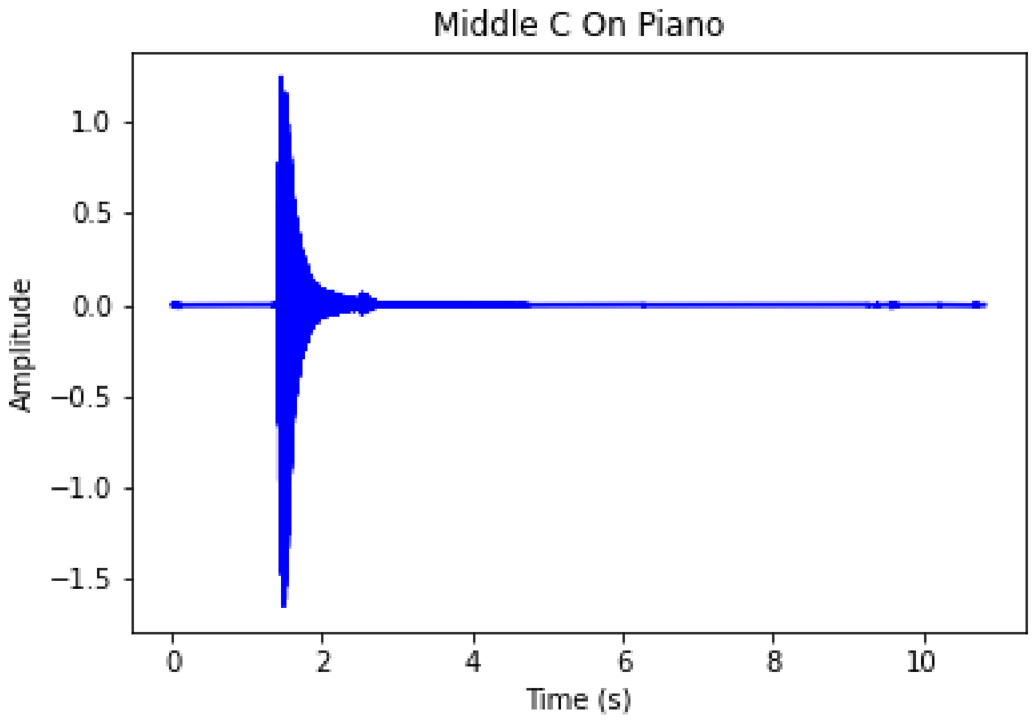


Figure 5: Middle C Time Domain Graph

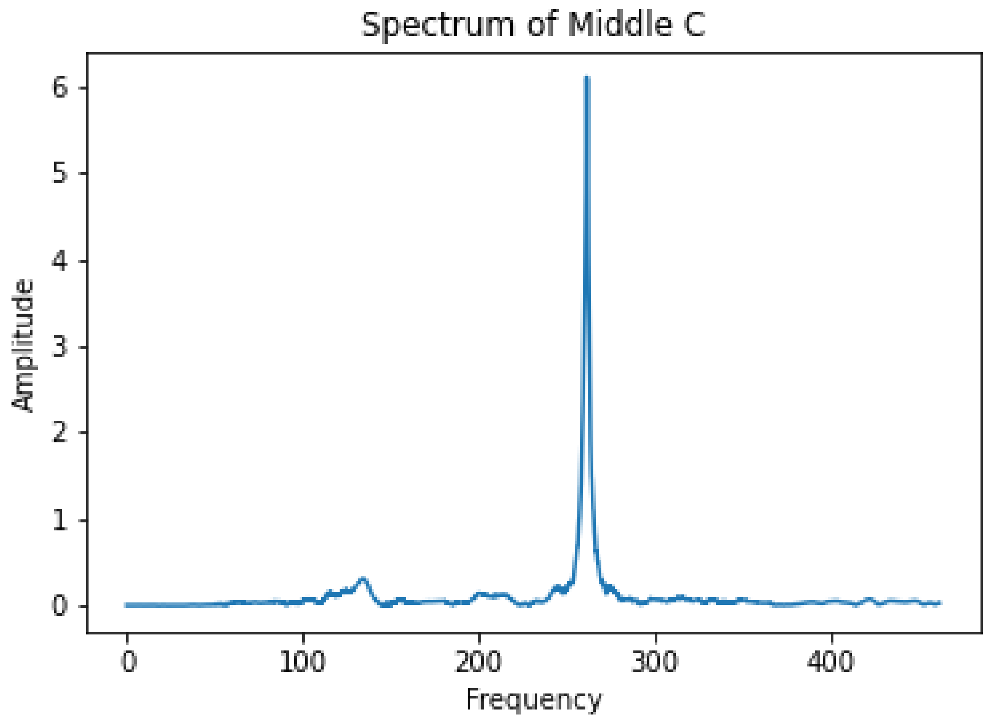


Figure 6: Middle C Frequency Domain Graph

In Figure 5, the "C" note on the piano is struck at ~ 1.75 s. The sound is loud at first and then fades. In Figure 6, the spectrum shows this note is just above 250 Hz which makes sense because a perfect middle "C" is 256 Hz [15].

There are many ways to transform a signal into the frequency domain such as the FFT, DCT, and DFT. Therefore, creating the spectrum of a signal is often referred to as a Fourier Transform.

After the spectrum is taken, the next step to represent a signal is by its spectrogram. The spectrogram is a graphical combination of the frequency and time domain.

2.3.5 Spectrogram

A spectrogram represents how the frequencies of a model change over time [16]. Figure 7 shows a spectrogram of drone acoustic emissions when no payload is carried. In a spectrogram, time is the independent variable and frequency is the dependent.

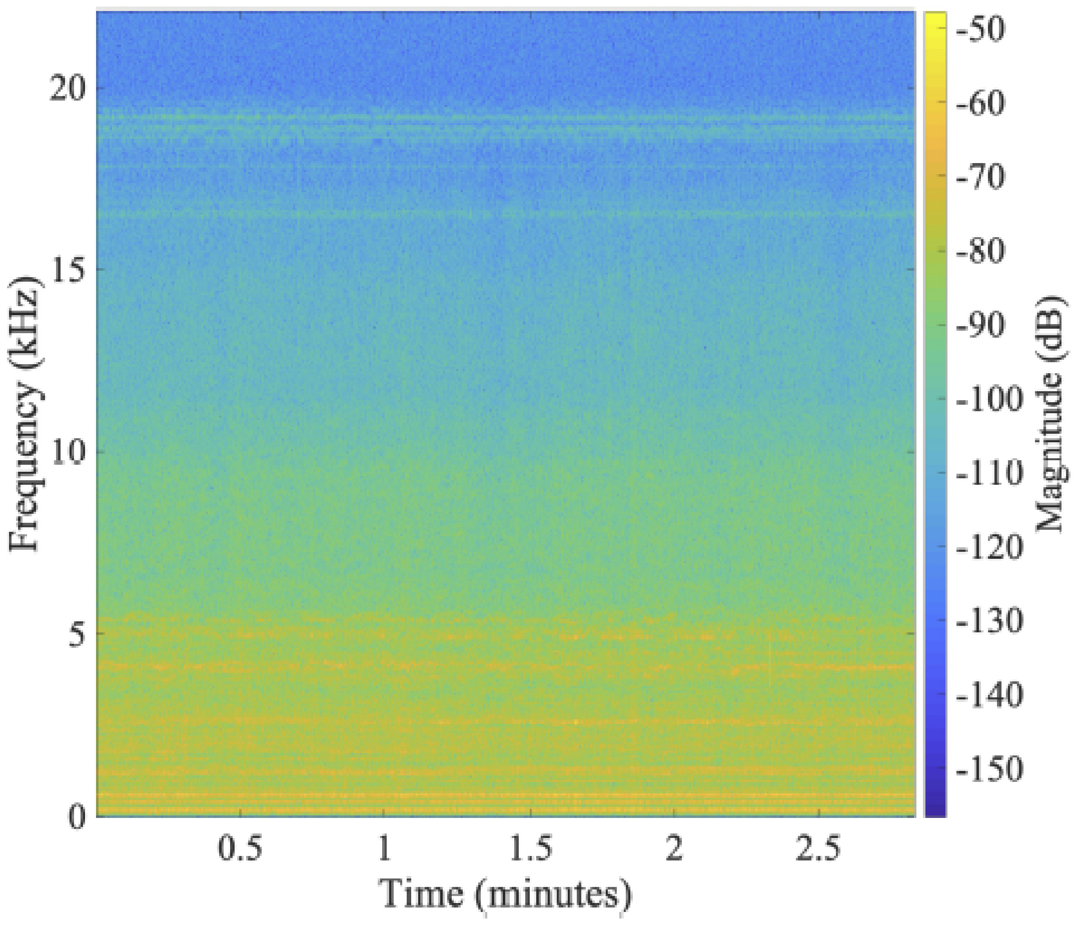


Figure 7: Spectrogram Of Drone Acoustic Emissions [7]

A spectrogram uses small samples or frames from a signal. A frame is a small duration of time in a signal where the frequencies are assumed constant. Each frame undergoes a Fourier Transform, and the spectrogram is a graph which shows each spectrogram over time. The frequencies in the spectrogram of each frame are color coded in the spectogram. In Figure 7 the loudest frequencies are bright yellow and the softest are dark blue. The color scale allows the spectrogram to convey which frequencies are most prevalent over time.

A spectrogram uses a linear scale for frequency [16]. It is a good model, and relevant information can be extracted for DSP and ML practices. However, MFCCs use a log-scale because humans perceive sound logarithmically, not linearly. So, the spectrogram needs to be scaled logarithmically. The MFCC generation process uses mel-scaling and the log-spectrum to achieve a perceptually relevant representation [10][13].

2.3.6 Mel-Scaling and Mel-Spectrogram

Humans perceive changes in sound logarithmically. As a result, some changes in frequency sound the same to humans. Therefore, a better representation of changes in sounds humans can perceive is pitch. A change in pitch indicates a change in sound that humans can detect. The unit for frequency is in Hz while the unit for pitch is in mel (derived from the word melody) [16]. For perceptual relevance, a spectrum needs to be converted from frequency (Hz) to pitch (mel). This conversion is called mel-scaling [16]. Figure 8 shows the relationship between frequency and pitch.

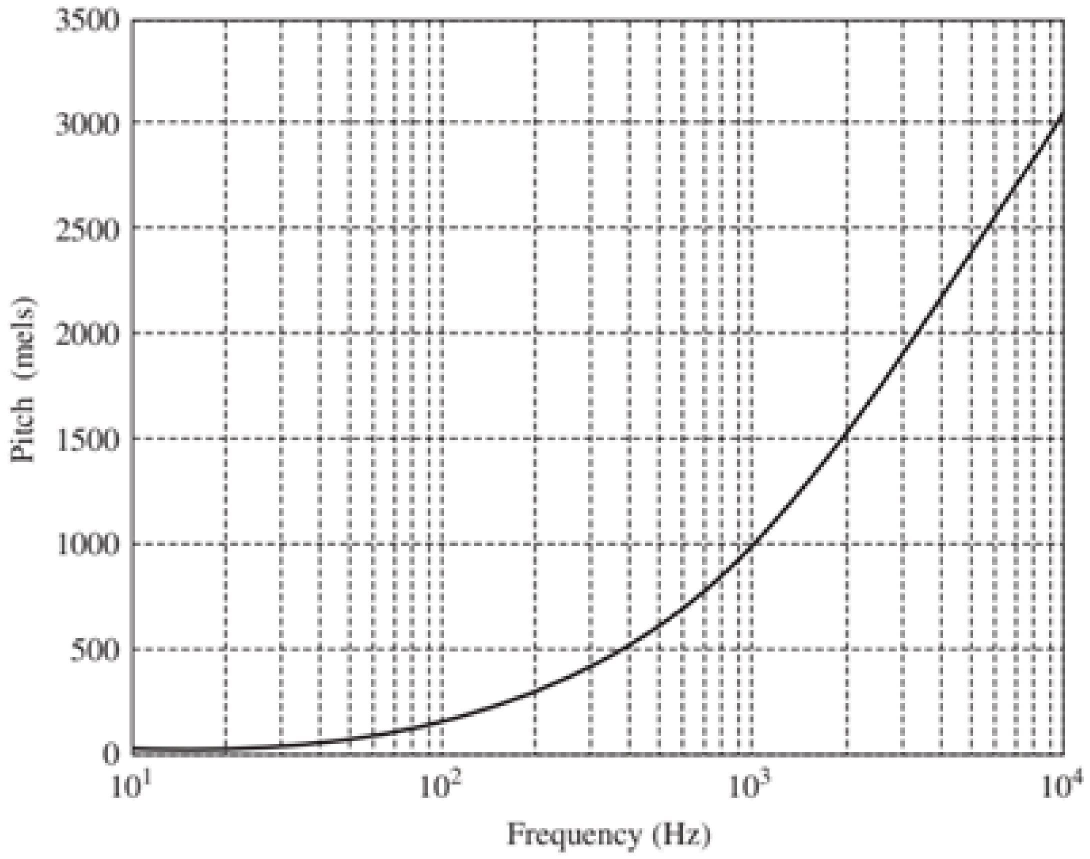


Figure 8: Frequency to Pitch Relationship [16]

In figure 8, frequency (Hz) is on the x-axis and pitch (mel) is on the y - axis. Th relationship between Hz to Mel on the graph is logarithmic and follows the equation [16]:

$$\text{pitch in mels} = 1127 \log_e(1 + \text{Frequency}/700) \tag{1}$$

The conversion shows the logarithmic relationship between frequency and mels and that humans do no perceive every change in frequency the same. Therefore, a conversion to the mel-scale is needed to portray the signal in a perceptually relevant way. The researchers in [13] used a mel-scale conversion to invent MFCCs.

Their research shows that the mel-scale along with other logarithmic scaling allows for digital representation of perceptually relevant sound. Their research invents MFCCs to portray every phonetic sound in the alphabet. The initial step to generate MFCCs is converting the frequencies to the mel-scale using mel filter banks.

Mel filter banks are a series of triangular filters used on the spectrum of the signal. Each mel filter bank represents a range of frequencies that sound the same to humans. Figure 9 shows the mel filter banks.

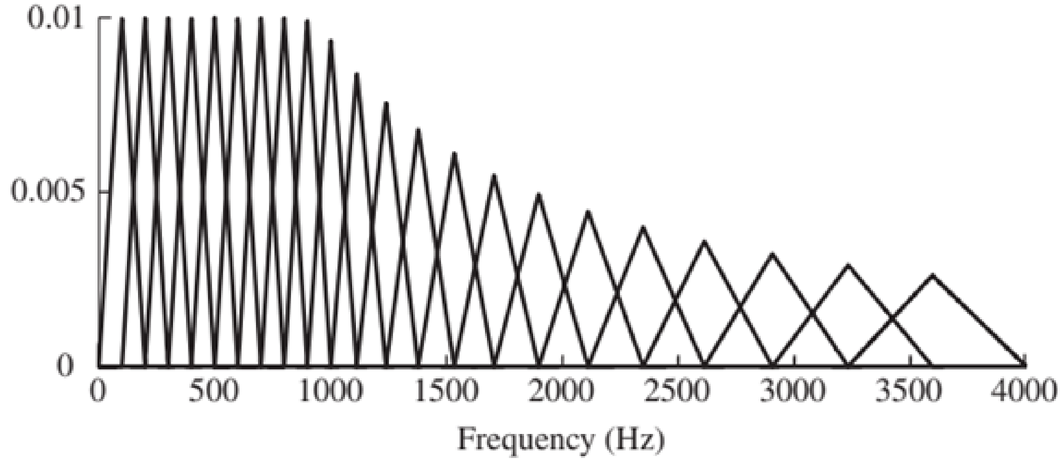


Figure 9: Mel Filter Banks [16]

This graph is a spectrum with frequency is on the x-axis and amplitude is on the y-axis. The over lapping triangles are the mel filter banks, and each triangle represents a unit in mels. The base of each mel filter bank spans the frequencies which sound the same to humans. The filters at higher frequencies have a wider base because because humans cannot perceive changes in higher frequencies as well as they can for lower frequencies. These triangular filters put all frequencies which are perceptually the same into the same mel unit. After applying these filters to a signal, the x-axis becomes pitch, and a change in the x-axis represents a perceptual change in sound.

After mel-scaling, the next step in MFCC generation involves the concept of cepstrums. A baseline understanding of human speech generation is needed to describe cepstrums. Therefore, the next section discusses human speech generation.

2.3.7 Human Speech

In the most basic form, human speech is the convolution of a glottal pulse and a vocal tract frequency response. The glottal pulse is the sound breathe and vocal chords create, and the vocal tract shapes this sound to create perceptually relevant sounds called phonemes. Phonemes are the distinct part of languages we hear such as a vowel or consonant. In American English, the phonemes are made up of vowels, diphthongs, semivowels, and consonants. Figure 10 shows the human vocal path which creates the signals that carry phonemes.

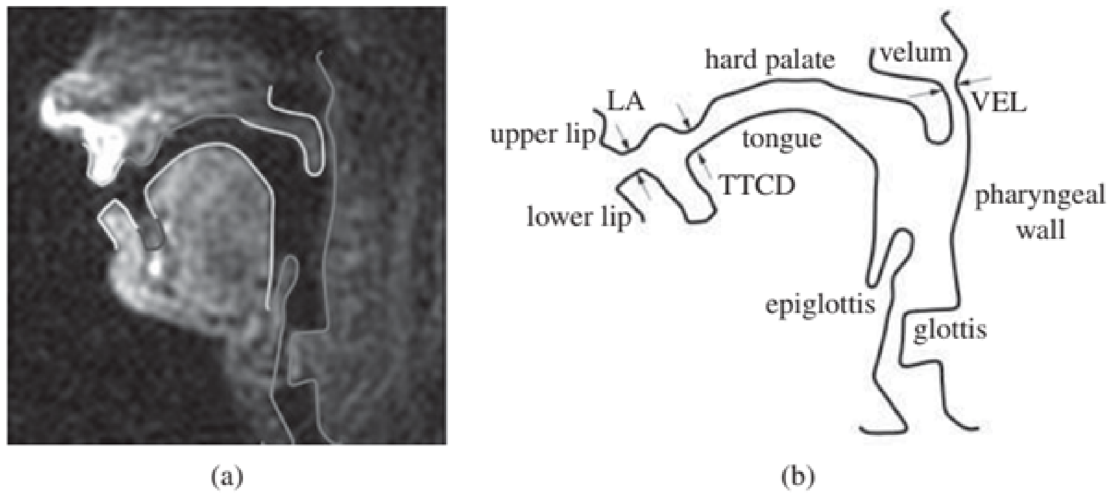


Figure 10: Human Vocal Tract [16]

Part a of Figure 10 is a scan of the human vocal tract, and part b is a labeled diagram of the tract. The glottis in part b creates the sound, then as the sound travels up through the vocal tract, it is shaped by parts such as the tongue to make specific phonemes, and recognizable speech exits the upper and lower lip. The recognizable speech is comprised of formant frequencies which are unique to every phoneme [16]. The formant frequencies, called formants, are the unique identifiers for every phoneme which if collected can be used to identify the speech.

The extraction of these formants is not a simple process. MFCCs are a set of 40 coefficients which represent the formants of a sound, so speech is digitally recognized by the value of the 40 MFCCs [13]. MFCC extraction is based on the mel-frequency cepstrum [13][16]. The first step in creating the mel-frequency cepstrum is converting the spectrum of a signal into a log-spectrum.

2.3.8 Log-Spectrum

Similar to the difference between frequency and pitch, the researchers in [17] discovered that the amplitude of each frequency on a spectrum does not correlate to the way humans perceive loudness. They found that scaling the amplitude on a spectrum logarithmically makes it perceptually relevant. Therefore, a process similar to mel-scaling converts the amplitude of a spectrum in a perceptually relevant way. The log-spectrum is the result of this process. This idea may seem contradictory since loudness is generally measured in decibels. However, the log-spectrum really is an additional log scaling of amplitude. In the log-scale, a change in the intensity of a frequency represents a change humans can detect.

Using the logarithm of a spectrum to convey perceptual relevance was first practiced in 1963 by Boger, Healy, and Tukey [17]. Their paper "The Quefrency Alanysis of Time Series for Echoes" demonstrated a revolutionary method for observing the presence of an echo in a signal [16]. Their research found that the logarithm of a Fourier spectrum would show periodic behavior if an echo were present, and that further analysis of the log-spectrum would provide thorough identification of the echo

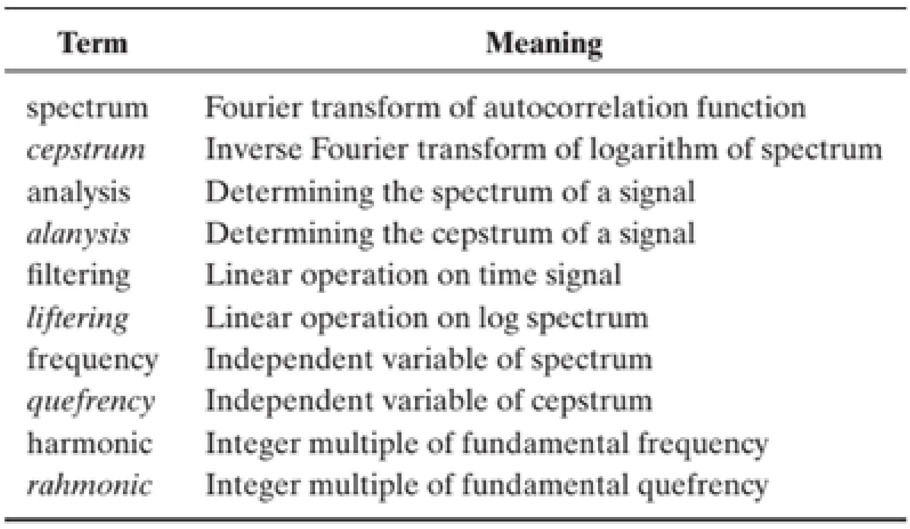
[16].

Because the log-spectrum displayed the echo in a periodic way, the identity of the echo could be extracted by taking the Fourier Transform of the log-spectrum [16]. To explain, first think of a digital signal in the time domain. The signal is periodic in nature, and by taking the Fourier Transform, the specific frequencies causing the periodic behavior are separated and represented as the independent variable in the spectrum. Since the log-spectrum is periodic, the Fourier Transform extracts the periodic components of the log-spectrum. These periodic components are then used to identify new attributes such as the echoes and harmonics of the signal [17]. The researchers in [17] called this new model a cepstrum which is a spectrum of a log-spectrum. Mel-frequency cepstrum coefficients uses the word "Cepstrum" because the cepstrum of the signal provides the coefficients.

2.3.8.1 Cepstrum

The word "cepstrum" was derived from flipping the first four letters of "spectrum." The reason for this play on words is because a cepstrum is technically in the time domain but serves the same purpose of a spectrum in frequency domain. The authors of [17] explain this phenomenon as, "In general, we find ourselves operating on the frequency side in ways customary on the time side and vice versa".

To shape this new model where the time and frequency domains interchange, Bogert et al. created many new terms by transposing letters in familiar forms such as spectrum to cepstrum [16]. Figure 11 shows the terms developed in [17] such as alanysis, liftering, quefrency, and rhamonic.



Term	Meaning
spectrum	Fourier transform of autocorrelation function
cepstrum	Inverse Fourier transform of logarithm of spectrum
analysis	Determining the spectrum of a signal
alanysis	Determining the cepstrum of a signal
filtering	Linear operation on time signal
liftering	Linear operation on log spectrum
frequency	Independent variable of spectrum
quefrency	Independent variable of cepstrum
harmonic	Integer multiple of fundamental frequency
rahmonic	Integer multiple of fundamental quefrency

Figure 11: Cepstrum Components [16]

In general, the cepstrum is a Fourier Transform of the log-spectrum. In [17], the exact transform was the Inverse Discrete-Time Fourier Transform (IDFT). An inverse Fourier transform is the process of transforming a signal from the frequency domain to the time domain. The inverse transformation is the opposite operation of the DFT or any Fourier Transform. Normally, the original time domain representation of a signal would result from the IDFT, but in this case the IDFT is taken from the log-spectrum rather than the spectrum, so the result is a cepstrum. In a cepstrum, quefrency is the independent variable. High quefrencies correspond to rapidly changing components (in frequency) of the signal, while low quefrencies correspond to slowly varying components [16]. Bogart et al discovered that isolated peaks at multiples of quefrency in the cepstrum correspond to echoes. Later in 1967, Michael Noll discovered the cepstrum could be used for detecting pitch period and voicing because speech includes the same type of repetitive time domain features as a signal with an echo [16]. Figure 12 displays the log-spectrum and cepstrum of voiced speech. The independent variable "Frequency [ms]" of the cepstrum is quefrency because it is frequency measured in ms.

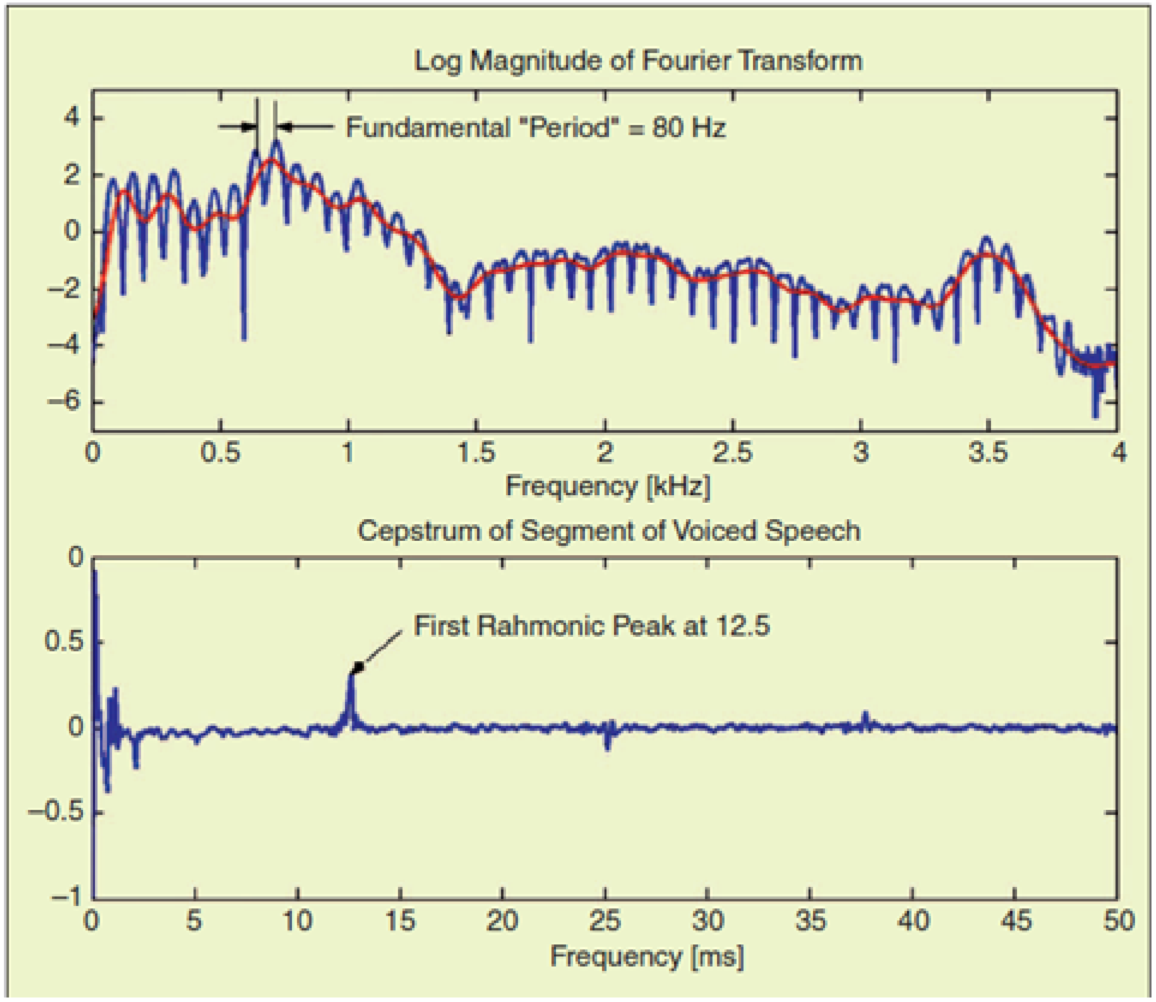


Figure 12: Log-Spectrum And Cepstrum Of Voiced Speech [18]

The cepstrum separates the glottal pulse and vocal tract of human speech. The glottal pulse is at the high quefrencies (> 10 ms), and the vocal tract components are at the low quefrencies (< 10 ms). With the vocal tract separate from the glottal pulse, information such as formants in the vocal tract is extractable. Many techniques such as mel-frequency cepstrum coefficients use the quefrency values of the vocal tract to represent all the phonemes in human speech, and these techniques make speech recognition possible [16].

2.3.9 Mel-Frequency Cesptrum Coefficient Generation

MFCCs are exactly what they seem. They are coefficients extracted from the cepstrum of a signal. Except the cepstrum is derived from a mel-frequency log-spectrum rather than a log-spectrum. In other words, the frequencies of a mel-frequency log-spectrum are in the mel-scale to make them correspond with human speech. By the spectrum being phonetically relevant in both the independent and dependent variables, meaningful values can be extracted to determine the letters present in a speech signal [16].

Following the process in Figure 2, first, a microphone records a signal and converts it to the digital form. When collecting the signal, the microphone and ADC often perform pre-emphasis measures to increase the quality of signal. One example of pre-emphasis is improving the signal-to-noise ratio. Then, the signal is framed into phoneme constant frames. The phonemes need to be constant in each frame because each set of MFCCs correlate to only one phonetic sound. Then, the system takes the Fourier Transform of each frame. Often, systems use the FFT because of its low-cost computation and fast speed. Once in the frequency domain, the system converts each frame to the mel-scale for perceptual relevant pitch evaluation. Then it converts the spectrum to the log-spectrum for perceptually relevant loudness. At this stage, the system has the mel-frequency log-spectrum. The last step performs the DCT of the mel-frequency log-spectrum creating MFCCs.

The DCT performs nearly the same function as the IDFT. MFCCs use the DCT over the IDFT for the following reasons. The DCT runs faster and requires less computational resources. The results of the DCT are only real while the IDFT results are both real and imaginary. And, the DCT causes dimensional reduction.

The cosine nature of the DCT allows different frequencies to be compared with the signal for coefficient extraction. In other words, different cosine functions are used in the DCT, and each cosine corresponds to an MFCC. Typically, a system uses 40 different cosines, each corresponding to a specific MFCC. Of the 40, only the first 12-13 are used for speech analysis. The reason the lower coefficients are used is because these coefficients hold the vital information for speech recognition. In ML, different ranges of MFCCs or even all 40 are used for practices beyond speech recognition. For example, researchers in [7] use all 40 MFCCs to train a ML for drone payload detection using acoustic emissions.

2.4 MFCC Example Problem

This section walks through an MFCC example problem. Lets say the Earth has been overrun by evil robots, and we must find the chosen one who is destined to save the planet. A prophetess has provided an audio recording on a flash drive of the name of the chosen one, but the robots have destroyed all speakers on earth. So, we must use digital speech recognition to discover the name of the chosen one, specifically MFCCs. In addition to the flash drive, the computer provided has a ML algorithm trained to print the letter on the screen that an array of 40 MFCCs correspond to.

Following the procedure in Figure 2, we divide the audio recording into phonetic constant sounds. We assume a 40 ms frame is short enough to divide the audio into phonetic constant sounds. Once framed, we take the FFT of each frame. With the spectrum of each frame, we convert the frequency to the mel-scale and take the log of the amplitude. The spectrum is now in a perceptually relevant scale. Then, we take the DCT of each frame with all 40 cosines which correspond to each of the 40 MFCCs. We now have an array of MFCCs for each frame. We feed these arrays to the provided ML and the letters are printed to the screen. We find that the letters are "NEO". Therefore, the chosen one is Neo.

2.5 Support Vector Machine

This research focuses on the Support Vector Machine for AI. Therefore, this section describes what a Support Vector Machine is.

A Support Vector Machine (SVM) is a type of Machine learning algorithm which uses hyperplanes to separate classes of data [19]. Figure 13 shows an SVM with a linear hyperplane.

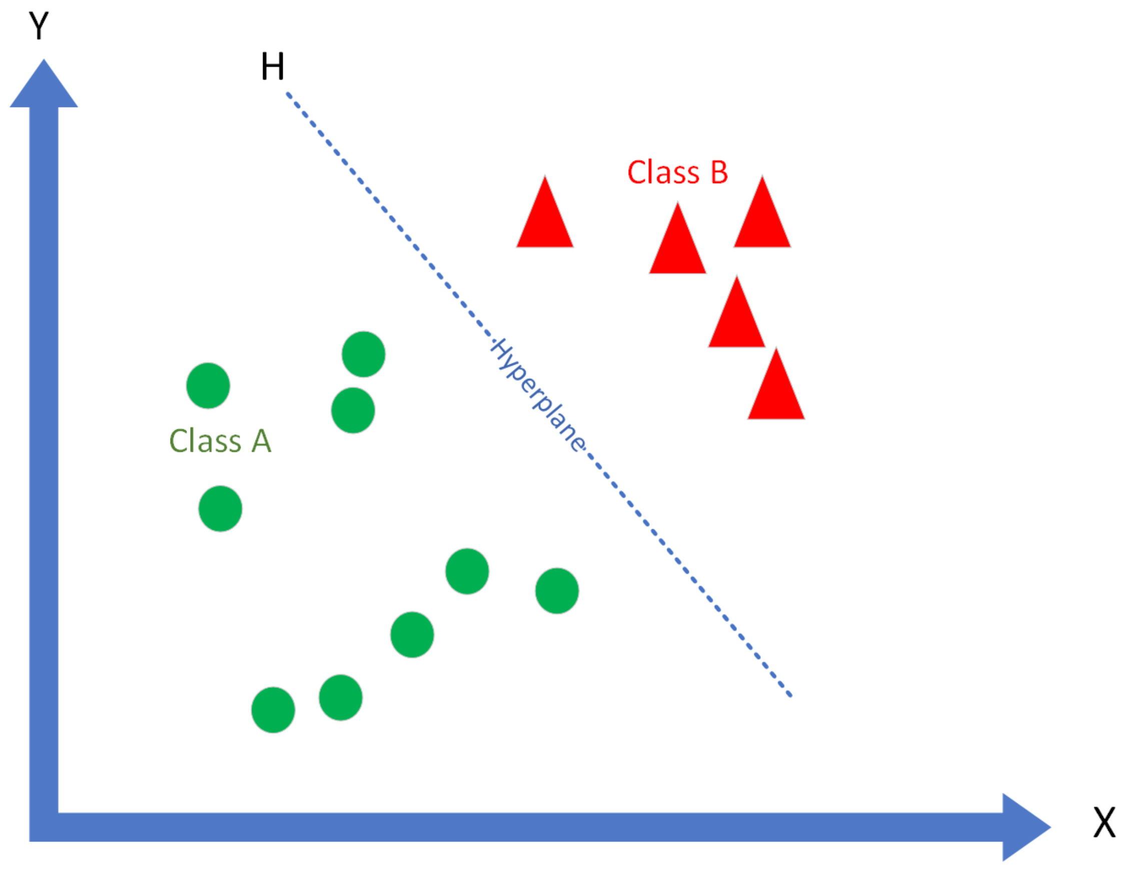


Figure 13: Basic Linear Support Vector Machine

In this figure, there is data from two different types of classes. The green circles are data of one class (Class A), and the red triangles are data of another class (Class B). The dotted line is the hyperplane. Each data point has a set of features which places it on one side of the line. In Figure 13, the features x and y place it on one side of the hyperplane, and based on the side it falls on, the SVM knows what class it belongs to. The rest of this section describes the process of creating an SVM. Figure 14 shows the components an SVM uses to generate hyperplanes and learn the difference between classes.

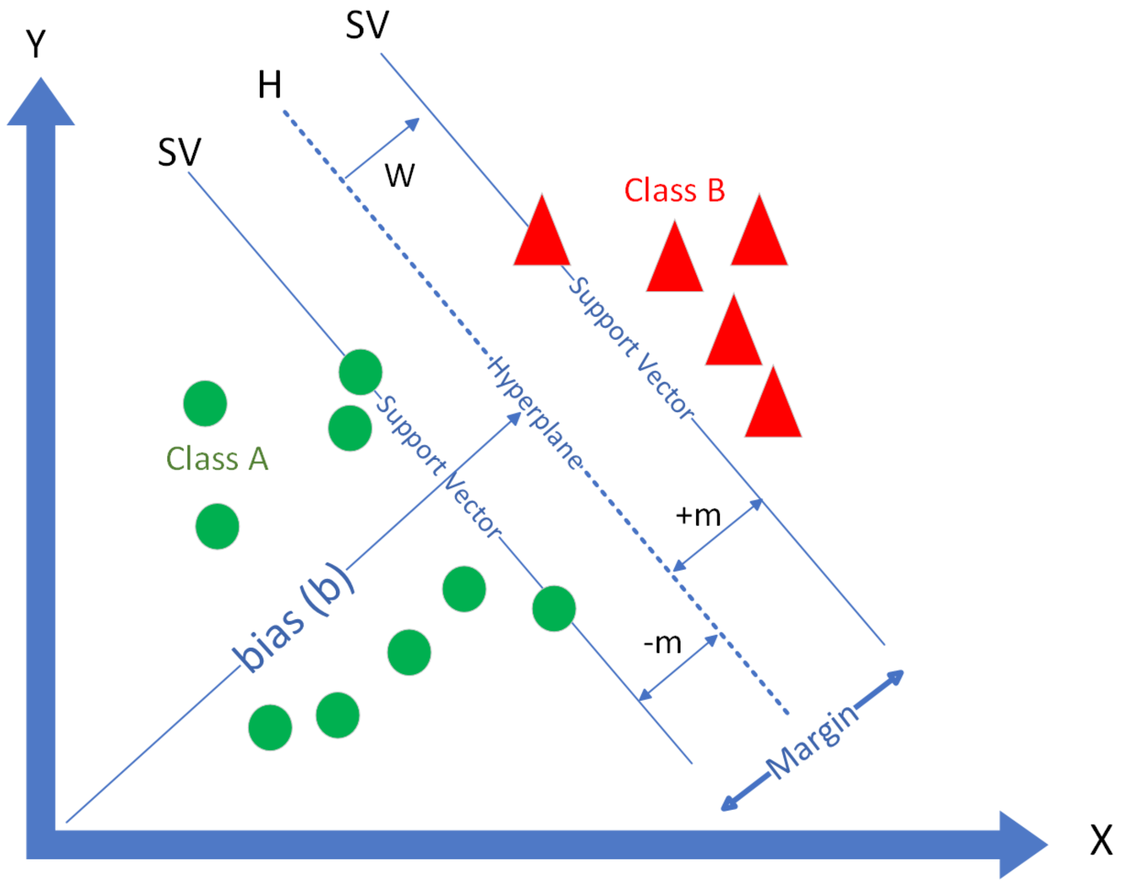


Figure 14: Detailed SVM Graph

The support vectors (SV) are the lines determined by the closest data points from each class to the hyperplane (H). W is the normal vector to the hyperplane called the weight. The distance from the support vectors to the hyperplane is -m and +m. The total distance between the classes is the margin. The bias (b) is the distance from the hyperplane to the origin.

When creating an SVM, the user introduces training data to an SVM with its class identity. In the case of Figure 14, the user introduces the green circles with the identity Class A and introduces the red triangles with the identity Class B. Using the features of each data point (x and y), the SVM plots the data in 2-D space. From here, the SVM fits a linear hyperplane between the classes which maximizes the margin between the classes. After finding the optimal hyperplane, the data points from each class which are closest to the hyperplane create the support vectors. The support vectors are used as the boundary between classes. The support vectors satisfy the equations [19]:

$$w * x + b = -1 : \text{ For Class A} \tag{2}$$

$$w * x + b = 1 : \text{ For Class B} \tag{3}$$

In these equations, x is the set of classifiers used to plot the data point, w is the weight, and b is the bias. In satisfying these equations, the SVM is now able to take any new data point, plug the classifiers into x for the equation:

$$w * x + b = \text{output} \tag{4}$$

and know if the data is of Class A or B based on if the output is greater than or less than 1.

2.5.1 The Kernel Trick

In many cases, a linear hyperplane is not sufficient for an SVM to fully classify data. The kernel trick is a technique commonly used to solve this problem. Instead of using a linear hyperplane, the kernel trick allows the SVM to use hyperplanes with greater dimensions. Figure 15 shows a linear kernel SVM, and Figure 16 shows a 6th degree polynomial kernal SVM

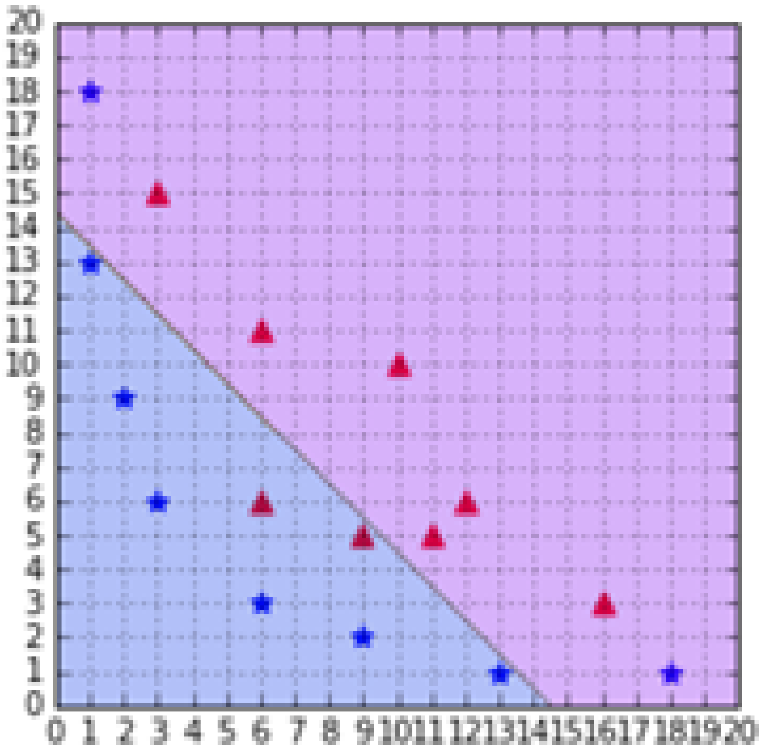


Figure 15: Linear Kernel SVM [19]

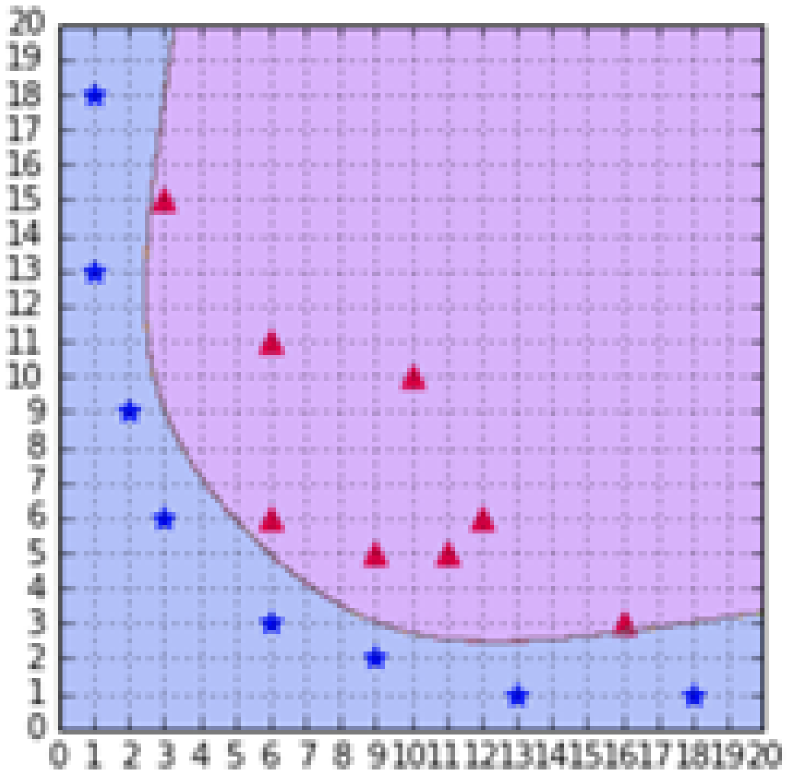


Figure 16: Sixth Degree Polynomial Kernel SVM [19]

In the figures, the two different classes are the blue dots and the red triangles. The linear hyperplane was not adequate to properly separate the two classes. Therefore, the SVM needs the sixth degree polynomial hyperplane to separate the classes. The use of the polynomial kernel is called the kernel trick. Hyperplanes with higher dimensonality are needed because many classifiers do not follow linear behavior. SVM kernels exist in every type of polynomial or dimensions. Although the kernel trick fixes this problem, hyperplanes with greater dimensonality create a greater risk of overfitting the data.

2.5.2 Overfitting Data

Overfitting data occurs when the SVM uses details in the classifiers which it should not. For example, if a researcher was using sound to classify data, it may be windy in some recordings and not in others. Therefore, some classes would contain wind noise while some did not. In using this audio data to train an SVM, the SVM may use the difference in wind to draw hyperplanes and classify data, rather than the targeted classifiers. In turn, the SVM may perform really well on the data used to train the SVM, but poorly on data recorded at another time or in a different environment. The best practice to avoid this is to increase the amount of training data used, and to collect training data at different times and in different environments.

SVMs are commonly trained by MFCCs for audio AI use cases. Because there are 40 MFCCs, using MFCCS to train an SVM results in a 40-dimensional space. Therefore, the kernel trick is often necessary to build a proper SVM with MFCCs, but the need for experiment validation is essential to avoid overfitting the data.

2.6 Related Work

Due to the diffusion of drones into modern day life, there is academic, military, and corporate interest in drone detection [7][20]. This interest has led to many developments for drone detection using a variety of signature methods. This section shows acoustic and non-acoustic detection. This section does not come close to covering the full research field, but these examples show the current state of drone detection.

2.6.1 General Drone Detection

In [21], researchers were able to use image processing to detect the presence of UAVs using Physically Based Rendering Toolkit (PBRT) and Faster R-CNN (convolution neural network). Their methodology achieved an 80.63% accuracy. Also, in [22], the authors were able to detect the presence of a drone using a computer vision-based approach. In their approach a 2-dimensional scale and Generic Fourier Descriptor features combined with thermal imaging were used to classify UAVs. They achieved a drone detection accuracy of 85.3%.

In [23], researchers used Radio Frequency (RF) to detect and classify UAV's. A naïve Bayes approach using the Markov model was implemented, and their system achieved an accuracy of 96.3%.

In [24], the authors were able to use RADAR for drone detection. Using Micro-Doppler signatures, their system achieved a 92% accuracy.

Image processing, RF, and RADAR are just a few approaches for drone classification. When comparing these technologies with acoustics, drone acoustic signatures propose a unique advantage. Acoustics offer advantages over other classification methods in environments with low visibility and low altitudes [4]. The acoustic noise produced by motors and blades of UAVs provide unique features which can clearly distinguish drones from other elements present in an environment. In addition, many applications such as LiDAR require direct line of site for drone detection while acoustics do not [7]. Furthermore, jamming and spoofing techniques for RADAR and RF, do not work for acoustic emissions. The next section discusses the research field for drone detection using acoustic signatures.

2.6.2 Acoustic Drone Detection

In [25], the authors designed a ML platform which uses Linear Predictive Cepstral Coefficients (LPCC) and MFCC techniques to detect and classify amateur drone sounds in the presence of other environmental sounds. They achieved a 96.7% detection accuracy using MFCC only and found that MFCC outperformed LPCC.

In [26], the authors used a combination of time and frequency domain features to fingerprint drone sounds. Their experiments were able to detect the presence of a drone in noisy environments which included a passing train, crowds, streets, and traffic noise. Overall, their process reached 95% accuracy for identifying drone acoustics.

In [27], using MFCCs, researchers were able to detect drone acoustics from 150 meters away in a quiet outdoor place. Additionally, they were able to detect drone acoustics in noisy environments with background noises from airplanes, people chatting, and cars passing. Their system achieved up to 86% accuracy.

In [28], authors were able to detect drone acoustics in environments with noisy air conditioners, car horns, children playing, dogs barking, drilling, and motors idling. Their system reported an 86.3% accuracy. They did not use MFCCs but used the Plotted Image Machine Learning algorithm from FFT graphical representation.

In [29], researchers developed a drone detection and position estimation system using a wireless acoustic sensor network with machine learning techniques. Their model reached 95% drone detection and positioning accuracy using acoustics.

While a plethora of research exists for drone detection using acoustic signatures, little research exists for payload identification using acoustics. The ability to identify the payload carried by a drone has many applications. A zero-touch payload identification process would mitigate risk as the user could identify if the drone is carrying something other than expected. For example, an acoustic payload detection system would be advantageous for the military by having a system which realizes the weight of the payload is incorrect, indicating the drone may have been captured or tampered with.

At the time of this research, only two papers have been published for UAV payload classification. Both papers use acoustic signatures to identify drone weights and are discussed in the next two sections.

2.6.3 Noise2Weight: On Detecting Payload Weight from Drone Acoustic Emissions

In spring of 2020, Ibrahim, Sciancalepore, and Pietro published a paper called "Noise2Weight: On Detecting Payload Weight from Drones Acoustic Emissions" [7]. Their research used MFCC components to classify specific payloads carried by a drone. Their system achieved a 98% accuracy for fingerprinting payloads from acoustic emissions.

Their research was the first published work for drone payload classification from acoustic signatures. In their work, two scenarios were modeled. Scenario one was based on a microphone deployed in a remote location such as a desert or agriculture field. They assumed this scenario would make it unfeasible to deploy a wireless camera to allow a remote administrator to immediately visualize a drone, so a system which could detect drones acoustically would be useful. They assumed that this environment would require a microphone deployed on a simple IoT device with limited computational capacity, so scenario one focused on detecting if a drone carried a payload or not.

In scenario two, the researchers simulated a scenario where the user would like to identify if a specific well-known drone is carrying a specific payload. In particular, they assumed a user is receiving a package via a drone-based delivery system and wants to identify if the package matches what was ordered. The user would like to identify the package without touching it, so the payload must be identified by analyzing the sound emitted by the drone. The researchers assumed the drone hovers in front of the house for a limited time in order to allow the acoustic signature to be identified. In the experiment for this scenario, the researchers tested variable payload weights from 0 to 500 grams in increments of 50 grams.

For both scenarios, the 3DR Solo drone was used to carry all payloads and emit all acoustic data. This drone was selected because it models a medium-sized commercial drone similar to what is used for current drone-delivery system. To record the acoustic emissions, the RodeVideoMicPro directional microphone was used. This is a high-quality microphone, averaging $300 in cost. For signal analysis, a MacBook Pro laptop was used. To control the drone and collect telemetry data, the Mission Planner software tool was used. This tool allowed the researchers to gather information about the rotational speed of the motors and the stability of the drone. The measurements collected by the microphone were stored on the laptop on the Audacity audio software tool.

All drone acoustic recordings were collected outside in an open environment. For each experiment, the microphone was placed 7 meters away from the hovering drone and each measurement lasted about 170 seconds. The researchers used matlab [30] for all acoustic feature extraction and identification processes.

From scenario one, the researchers concluded the pitch of the sound emitted by the drone was an adequate measure for determining whether the drone is carrying a payload or not. In their experiment, they varied payload weight and the time window each sample was gathered. The results showed that regardless of weight and time window, the pitch of the acoustic emissions shows whether the drone is carrying a payload or not. They found that increasing the payload weight led to an increase in the mean value for pitch frequency. This observation was consistent with their predictions that the higher the payload, the more thrust would be needed to carry the payload, so the rotors and motors would need to rotate faster leading to a higher average pitch frequency.

Furthermore, they observed that similar payloads weights overlapped in their pitch measurements, so they concluded it would be difficult to detect specific weight by solely looking at the pitch of sound. The researchers recommended repeating scenario one with a better quality high-sensitive microphone to mitigate this error. They also observed that heavier payloads did not overlap in pitch measurements as much as lighter weights.

By varying the time window, the researchers observed that larger window times result in less error. The results showed that a heavier payload weight and longer time window increased detection accuracy. For scenario one, they concluded that pitch is sufficient to determine if a drone is carrying a payload especially if the payload is heavy and the drone can be observed for long periods of time.

In scenario two, the MFCC coefficients of each acoustic sample were taken in order to determine payload weight. The researchers used the process shown in Figure 2 to extract the MFCCs. 40 MFCC coefficients were created for each sample. The payload weights and window size were varied. To show the effect of window size on classification, two windows sizes were used: 0.25 s and 1 s. The results showed that a longer time window led to more accurate results in the MFCC features.

The MFCC features were used to train 10 different Machine Learning classification algorithms. Of the 10, the researchers determined the SVM algorithm performed the best. Using the SVM, their process achieved a 98.4% classification accuracy.

2.6.4 Identification of Drone Payload Using Mel-Frequency Cepstral Coefficients and LSTM Neural Networks

Traboulsi and Barbeau used MFCCs to train a neural network to identify drone payload from acoustic emissions in [8]. A neural network is a trainable machine learning structure that emulates neurons in the human brain. The network is trained by data to detect features when similar events in the future occur. In this research, MFCCs extracted from drone acoustics were implemented into a Long Short-Term Memory (LSTM) recurrent neural network. Later, the network's ability to identify payload weight was tested by flying the same drone with similar payload weights.

For all experiments, the Parrot Mambo mini copter was flown indoors with payload weights from 0 to 28 grams attached. The researchers could attach a maximum weight of 28 g to the drone and still maintain flight. For each weight, two 8 second data collects were performed. The first data collect was conducted while the drone was hovering at 1 meter, and the second was conducted while the drone flew around at one meter.

Acoustic emissions were recorded using the microphone built into a late 2013 Model Macbook pro. Each sample was processed in matlab using a package with a built in LSTM model. Throughout experimentation, white noise from the Gaussian model was added to test different SNR ratios.

MFCCs were extracted from the 0 g, 20 g, and 28 g payload tests. These MFCCs were used to train the LSTM neural network. Then the neural network was tested with data gathered from experiments with 0 g, 10 g, 20 g, 24 g, and 28 g payload weight. For each experiment, the SNR was varied from -10 dB to 35 dB. At 35 SNR dB, the accuracy of the LSTM model converged on 87.5%.

The authors concluded that the payload weight of a drone can be identified by acoustic signature. They recommended gathering MFCCs from different drone models with differing payloads to train and test the LSTM neural network.

2.7 Contribution

At the time of this research, the researchers have not identified any other studies which explore the ability of cell phone devices to classify drone payloads from acoustic emissions. The ability to use cell phones for payload detection would be useful to detect and classify UAVs in contested environments. This research provides evidence for the ability to use cell phones to detect and classify drone payloads from acoustic emissions. Table 1 shows how this research contributes to the body of work in this field.

Table 1: Research Field Comparison

Study	**Acoustics**	**Outdoor**	**Presence**	**Positioning**	**Payload**	**Cell Phones**
Anwar et al. (2019) [25]	x	x	x			
Bernardini et al. (2017) [26]	x	x	x			
Jeon et al. (2017) [27]	x		x			
Kim (2018) [28]	x		x			
Yue et al (2018) [29]	x	x	x	x		
Ibrahim et al. (2020) [7]	x	x			x	
Traboulsi et al. (2021) [8]	x				x	
Doster (2022)	x	x			x	x

2.8 Summary

The exponential diffusion of UAVs into every day life raises many security concerns. To mitigate these concerns, drone detection systems are being developed. Using RADAR, cameras, radio frequencies, and acoustics, researchers have been able to detect the presence of a drone and detect its features using feature extraction algorithms and machine learning processes. Specifically, the acoustic emissions from a drone have been used to identify the location of a drone, features on the drone, and the weight of the payload it is carrying. In these scenarios, high-quality microphones are used to observe the acoustic emissions at close distances. Researchers recommend using even higher equality microphones for better feature extraction. This research seeks to determine if low-quality microphones commonly found on cell phones are able to classify payload weight from acoustic emissions.

III. HurtzHunter Design

3.1 Overview

This chapter presents the HurtzHunter prototype design developed in this research. Section 3.2 gives a summary of the design and use process for HurtzHunter. Section 3.3 presents the design goals of HurtzHunter. The cell phones HurtzHunter uses to record data are discussed in Section 3.4. Section 3.5 discusses the high-quality microphone used by HurtzHunter. Section 3.6 presents the Python scripts in HurtzHunter. Finally, Section 3.7 gives a step-by-step process to use HurtzHunter.

3.2 System Summary

This research develops HurtzHunter, a prototype design to test UAV detection capabilities. HurtzHunter is a UAV detection system which characterizes UAV payloads. Specifically, it uses the acoustic emissions from UAVs to determine the payload weight carried by the UAV. Past research demonstrates the ability to detect and classify payload weight using acoustic emission at short distances and with high-quality microphones [7][8]. HurtzHunter expands the research field by conducting payload classification at varied distances and with cell phone devices. It records acoustic stimulus at 7 - 100 m from the source. The prototype uses the following recording devices: three Samsung Galaxy S8s (GS8), one Samsung Galaxy S20 (GS20), one Google Pixel 4 (GP4), one iPhone 11 Pro (iP11P), one RodeVideoMicPro (RVMP). The researchers in [7] successfully used the RVMP to create a UAV payload prediction system for close distances. This research implements the RVMP into the HurtzHunter design to provide experiment validation and comparison with past research.

Figures 17 and 18 show the hardware setup for cell phones in HurtzHunter. In Figure 17 the phones are in the recording phase. In this phase, they are placed on the stand as shown in the figure. In this configuration, the phones are not connected to anything. Figure 18 shows a phone in the download phase. After recording, the phones are individually taken off the stand and connected to the computer via USB connection. The user manually saves the audio recordings on the computer in this phase.

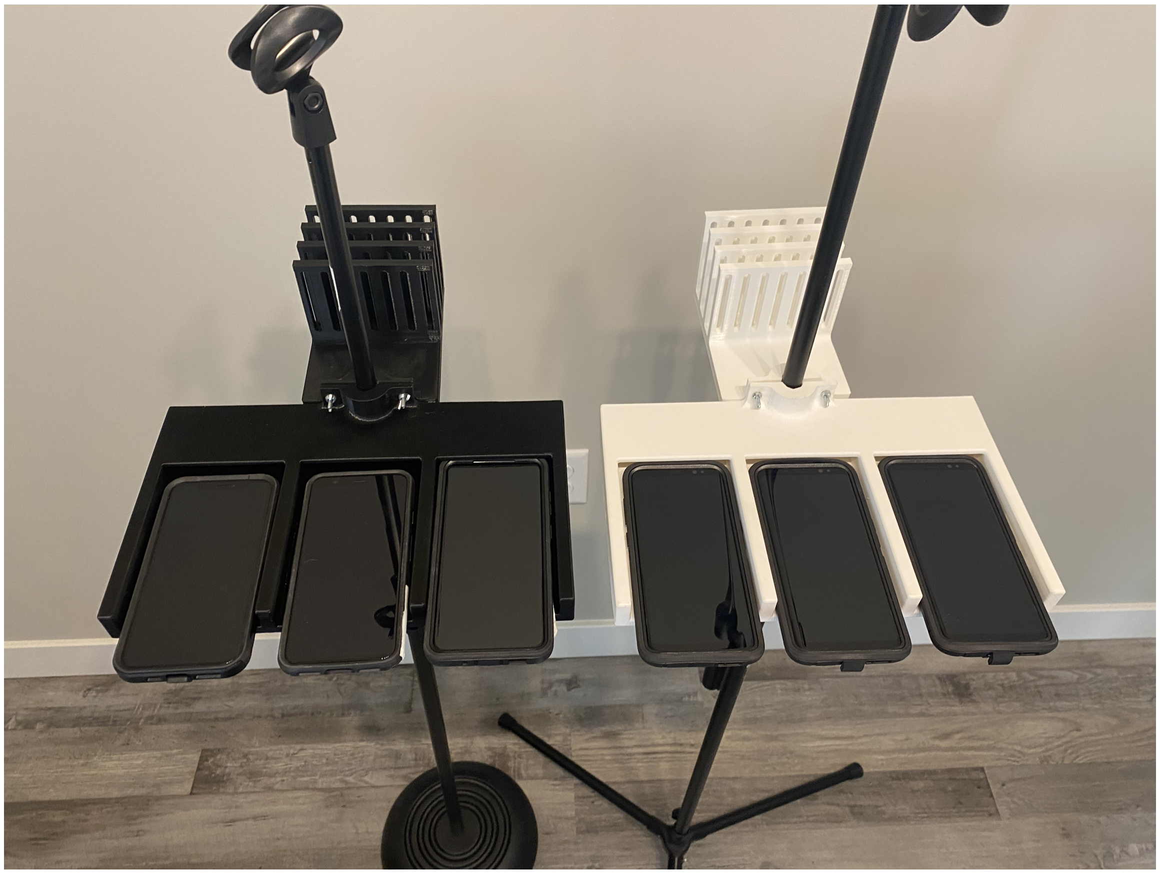


Figure 17: Phone Recording Setup

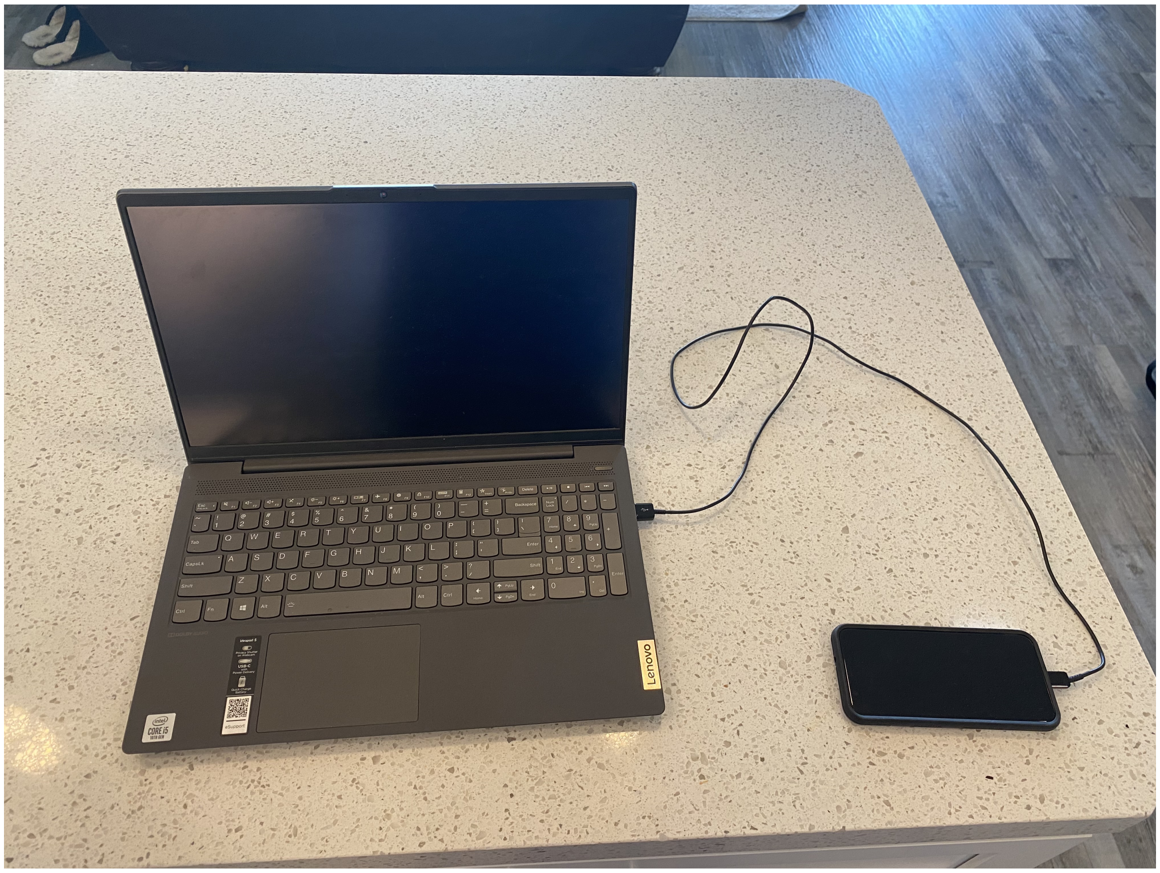


Figure 18: Phone Downloading Setup

This research refers to the recording phase as phase one, and the download phase as phase two for the cell phones:

1. Phase One: The cell phone recording phase. In this phase, the cell phones are placed on the stand as Figure 17 shows. The cell phones record the acoustic stimulus in this phase and are not connected to anything.

2. Phase Two: The cell phone download phase. In this phase, the user connects each cell phone to the computer via USB connection and manually saves the audio recordings to the computer. Figure 18 shows this configuration for one cell phone.

Unlike the cell phones, there is only one configuration for the RVMP. Figure 19 shows this configuration.

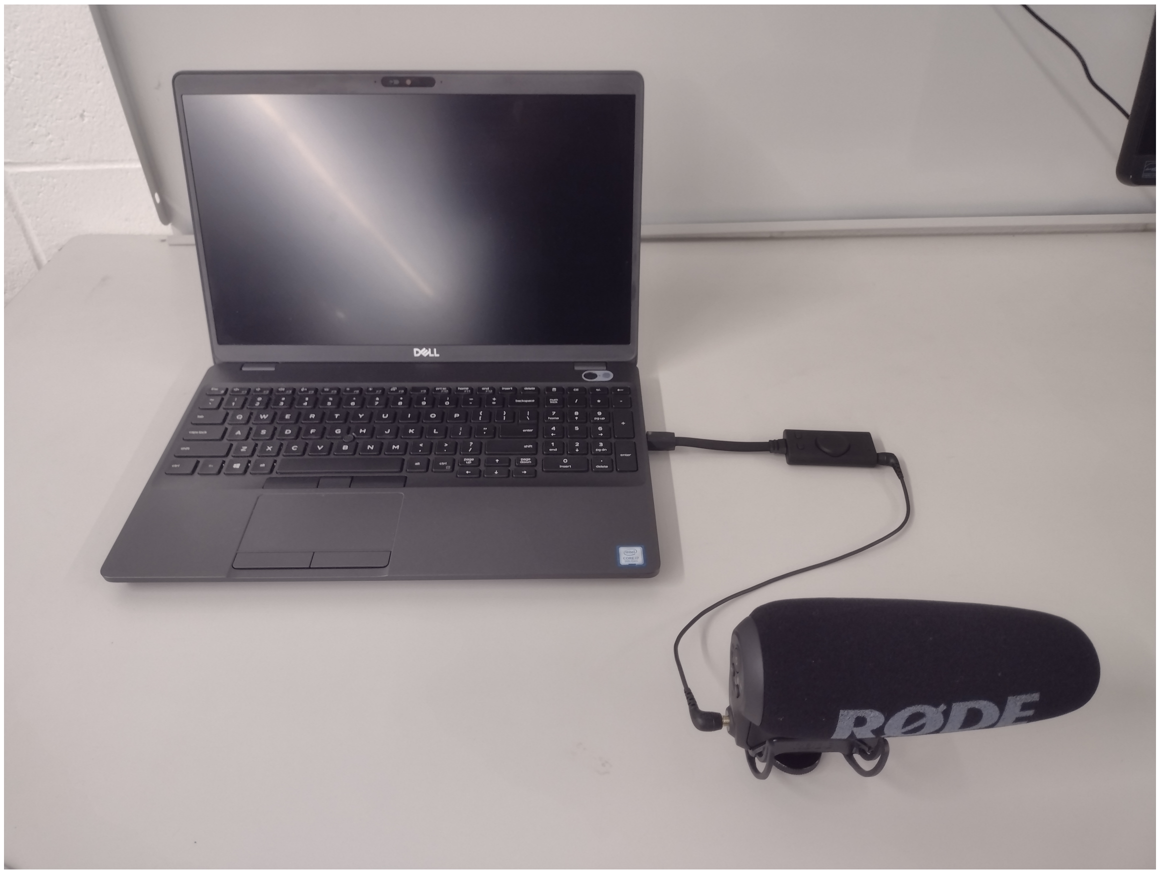


Figure 19: RodeVideoMicPro Hardware Setup

The RVMP cannot record and store audio without external audio software and storage. So, the microphone connects to the laptop during recording, and the design stores the audio directly onto the laptop using the Audacity [31] recording software. To connect the microphone to the computer, the design implements an Analog to Digital Converter (ADC). The specific ADC in this research is also an audio splitter to separate in-bound and out-bound audio. HurtzHunter uses the TechRise USB External Stereo Sound Adapter Splitter Converter as the ADC [32].

This research implements the HurtzHunter design using an HP Pavilion Business

Laptop with a 10th Gen Intel Core i5 [33] shown in Figure 18. The laptop in Figure 19 is not the laptop used in this research. Figure 20 shows the full HurtzHunter configuration.

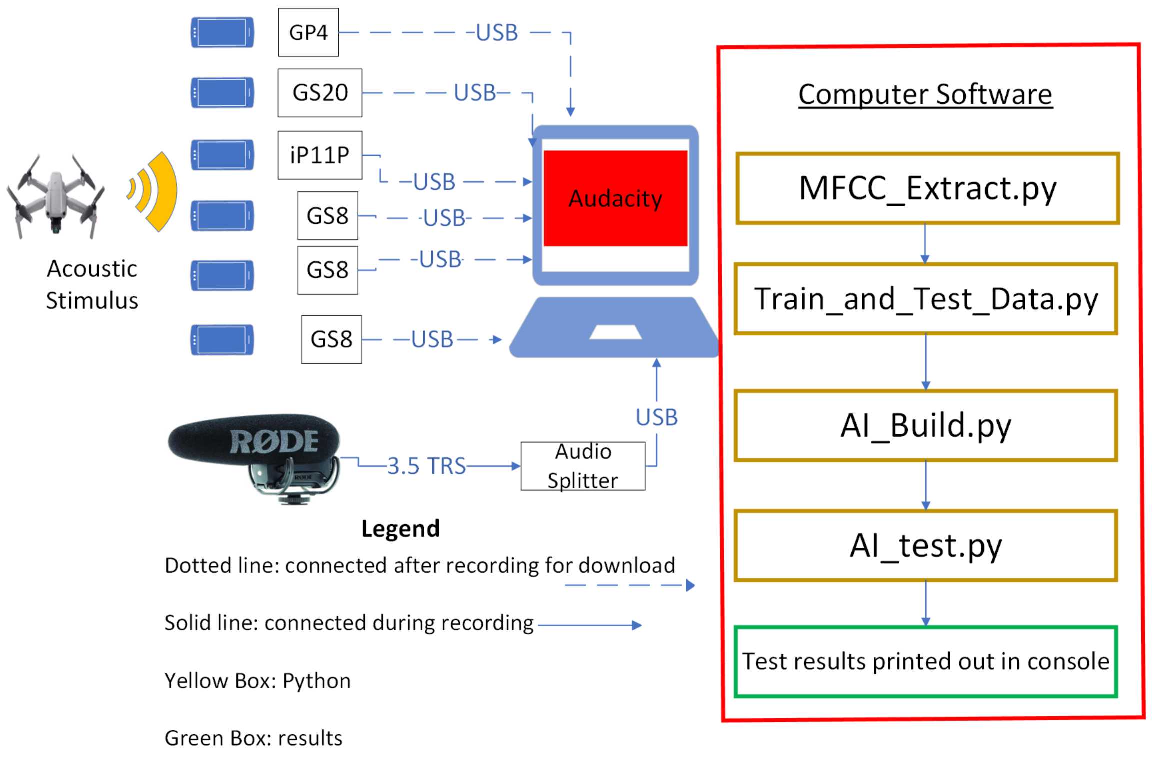


Figure 20: HurtzHunter Full Diagram [11]

The first stage in the HurtzHunter full diagram is the recording stage. The only device connected to the laptop during recording is the RVMP. After recording, the prototype enters phase two where the system stores the audio recordings on the computer. In phase two, the user manually connects each cell phone to the laptop one-by-one and downloads the audio files to the computer. The Audacity software stores the RVMP recordings in real time, so they do not need to be downloaded in phase two. After recording, the user manually extracts the recordings from Audacity to the computer's file system. Once all the audio files from each device are in the computer's file system, phase 3 begins.

In phase 3, HurtzHunter uses Python version 3.8.8 [34] scripts to build AIs for each recording device from their audio recordings. The user manually runs the Python scripts in JupyterNotebook [35].

In short, this research creates the HurtzHunter prototype as the platform to demonstrate drone detection capability using sound. While a drone hovers in flight, HurtzHunter collects the drone's acoustic emissions for one minute. After recording, the user manually stores the recorded audio to the computer's file system. Once on the file system, the system uses Python to extract MFCC audio features. Then it uses the MFCCs to train an AI capable of recognizing certain payload weights from sound. Last, the prototype tests the AI, printing the accuracy results to the console.

3.3 Model Design Goals

The design of HurtzHunter targets the following goals. HurtzHunter itself, would not be the design deployed in a contested environment, but HurtzHunter demonstrates the feasibility of an acoustic detection system using cell phone like devices.

- Low Cost: HurtzHunter uses recording devices commonly found in contested environments. Instead of having to buy and deploy new recording devices with processing capability, HurtzHunter shows the ability to use devices, specifically cell phones, already found in most environments, to detect UAV payload. The software in HurtzHunter is deployable on these type of devices at no cost.

- Portability: HurtzHunter demonstrates a portable acoustic drone detection system. The Python scripts in HurtzHunter are installable on a large range of cell phones and other computing devices. These devices are portable, and they are deployable nearly anywhere for drone detection.

- Recording Device Diversity: By using four different cell phone devices, this research demonstrates the ability to accomplish acoustic detection on a range of devices. In addition, many IoT devices use similar recording technology as cell phones for recording. Demonstrating acoustic payload detection on these cell phones not only provides robust evidence for acoustic detection using cell phones, but on any IoT device with similar recording technology.

- Sensor Comparison: HurtzHunter demonstrates the detection accuracy for a high-quality microphone (RVMP) and the cell phone. This research considers the RVMP a high-quality microphone because of its specialized design and cost. At about $300 USD, the RVMP is commonly used for high-quality audio featured on videos. It is industry standard for voice clarity [36]. Past research successfully used the RVMP to detect drone payload weight [7]. By using the RVMP and cell phone devices, the HurtzHunter design shows the difference in acoustic prediction accuracy between using a high-quality microphone and cell phone devices.

- Robust Prediction: The HurtzHunter design assesses the prediction accuracy of 7 different devices. This diverse prediction accuracy provides robust results to indicate the feasibility of acoustic UAV payload detection using cell phone devices.

- ML Optimization: As Chapter 4 discusses, HurtzHunter optimizes the ML used for payload detection allowing this research to recommend a specific design configuration for acoustic drone payload detection.

- Wide Range Detection: Past research only demonstrated drone detection at 7 m away from the drone. HurtzHunter tests drone detection from 7 to 100 m away from the drone.

3.4 Phones

This section discusses the hardware and software configuration for the cell phones. This design uses 5 Android and 1 Apple devices. The 6 phones use the same hardware configuration in the HurtzHunter design, but the iPhone requires a slightly different software configuration than the Androids.

3.4.1 Hardware

Cell phones use Micro Electrical-Mechanical System (MEMS) microphones. MEMS are tiny microphones embedded into a computer chip [37]. These microphones are small, inexpensive, and require very minimal power. Audio quality is not a priority for cell phone provides, so the small, inexpensive, minimal power consumption nature of the MEMS microphones make them ideal.

Typically, a cell phone has 2-3 MEMS microphones. Figures 21, 22, and 23 show the three MEMS microphone positions on the iP11P.

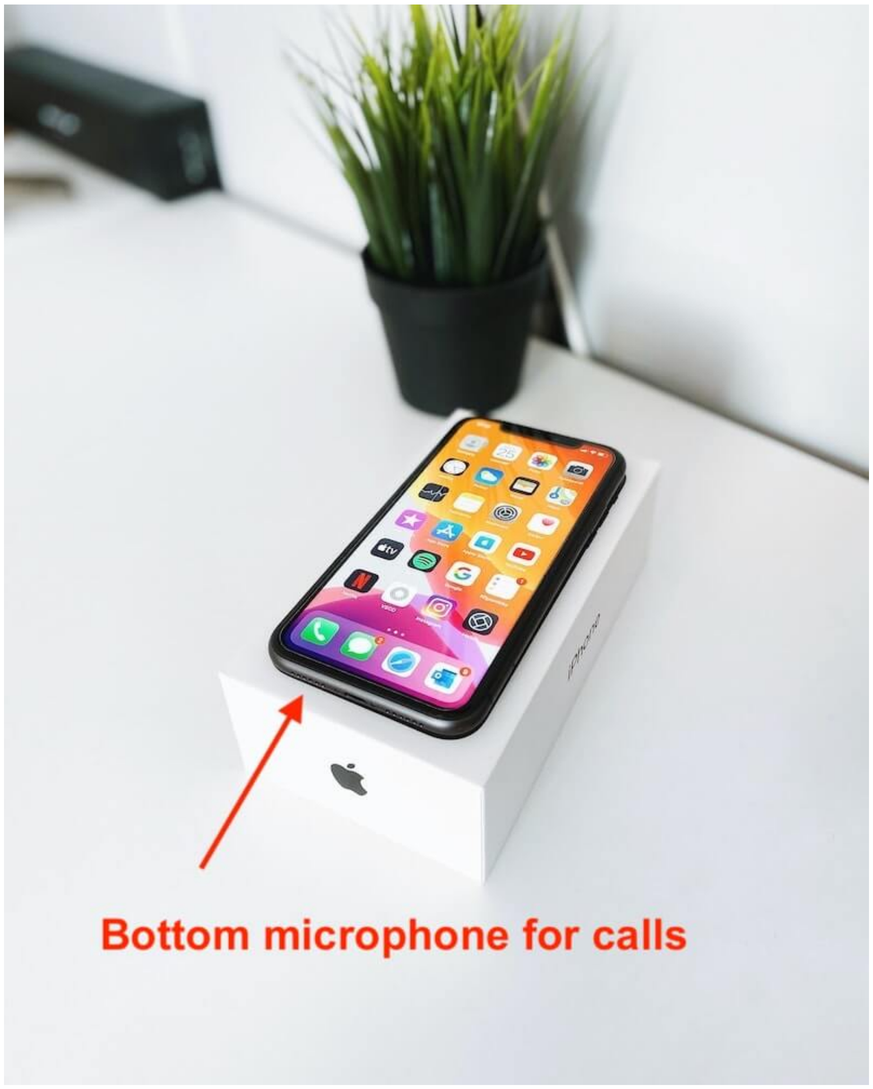


Figure 21: Bottom MEMS Microphone Location [38]

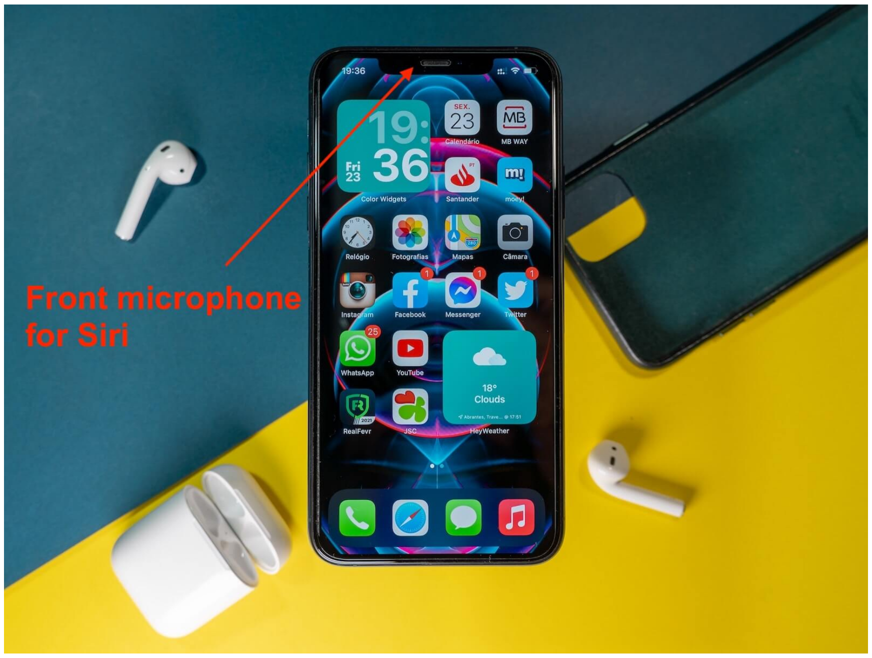


Figure 22: Front MEMS Microphone Location [38]

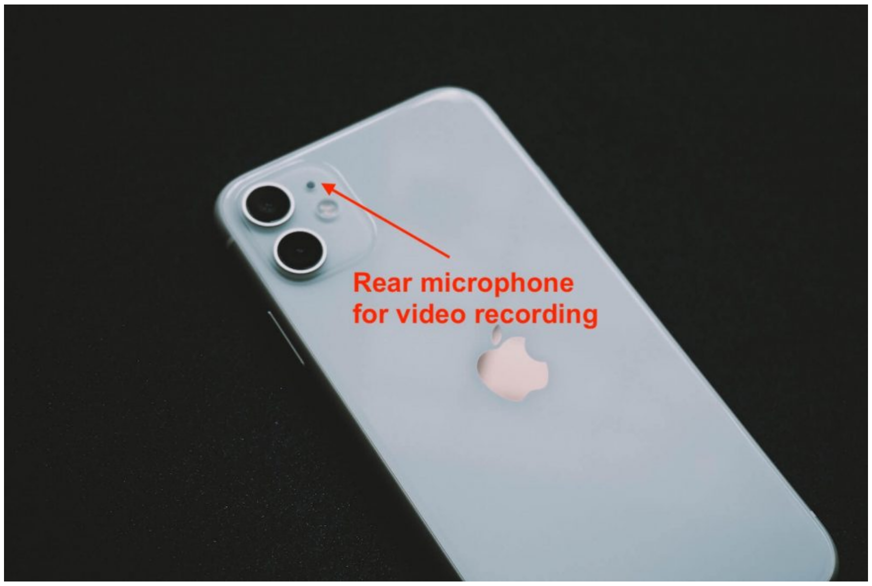


Figure 23: Rear MEMS Microphone Location [38]

In this research the GS8 and GP4 have two MEMS microhpones. On these phones, one MEMS microphone is at the front position as displayed in Figure 22, and the second MEMS microphon is at the bottom as displayed in Figure 21. The iP11P and GS20 have three MEMS microphones. On these two cell phones there is one MEMS microphone at each position shown in Figures 21, 22, and 23. For all cell phones, the main microphone is the one at the bottom. Tables 2 and 3 show hardware and software specifications for each cell phone in this research.

Table 2: Phone Specifications [39]

Phone	**Release Date**	**OS**	**System on Chip**
Samsung Galaxy S 8	21-Apr-17	Android 9	Samsung Exynos 9 Octa 8895
Google Pixel 4	22-Oct-19	Android 11	Qualcomm Snapsdragon 855
Iphone 11 pro	13-Sep-19	iOS 14.5.1	Apple A13 Bionic APL1W85
Samsung Galaxy S20	6-Mar-20	Android 11	Qualcomm Snapsdragon 865

Table 3: Phone Specifications [39]

Number of Microphones
2
2
3
3

3.4.2 Software

Figure 24 shows the Android software process for HurtzHunter.

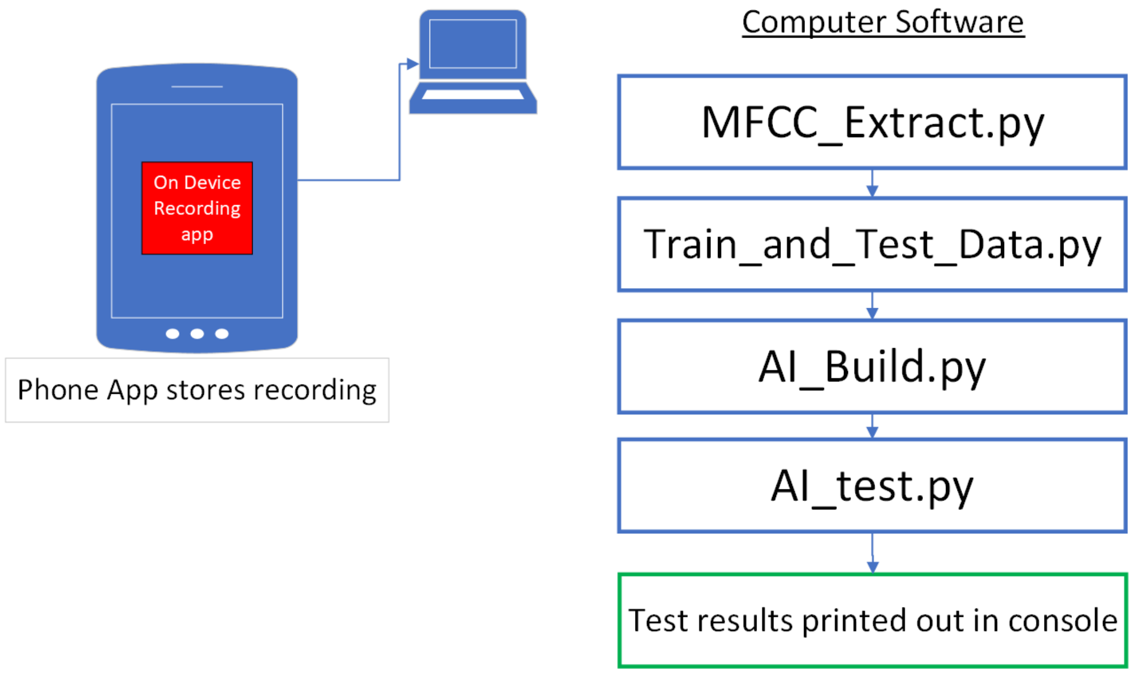


Figure 24: HurtzHunter Android Software Diagram

Each cell phone uses a recording application. The Androids use the SmartRecorder by SmartMob [40], and the iPhone uses Voice Memos [41]. SmartRecorder is in the GooglePlay store for download, and Voice Memos comes installed on every iPhone. Both apps have a start and stop recording button the researcher uses to manually start and stop the audio recordings. These apps store the recorded audio on the cell phones. The Androids store the audio as a .wave file, and the iPhone stores the audio as a .m4a files. These apps collect with a sampling rate of 44.1 kHz.

After moving the audio to the computer, the researcher manually runs each Python script for data processing. Section 3.6 discusses each Python script in detail. The computer uses jupyter notebook as the Integrated Development Environment (IDE) to run each script.

The first Python Script is MFC_Extract.py. This script divides every audio file into frames and extracts MFCCs from each frame. The MFCCs are the audio features used to train and test the AI. After all MFCCs are extracted, the script stores them in a .pkl file.

The next Python script is Train_and_Test_Data.py. This script divides the MFCCs into testing and training data. The system uses the training data to train the AI and the testing data to evaluate the AI.

Next, the script AI_Build.py creates and trains the AI. HurtzHunter uses an SVM as the ML. Based on their results, the researchers in [7] concluded the SVM is an appropriate model for acoustic payload detection. For this reason, this research chooses to use the SVM.

The last script is the AI_Test.py which uses the test data to evaluate the SVM built by AI_Build.py. This script prints the prediction results to the console.

The iPhone requires one more software step than the Androids. Figure 25 shows this additional step.

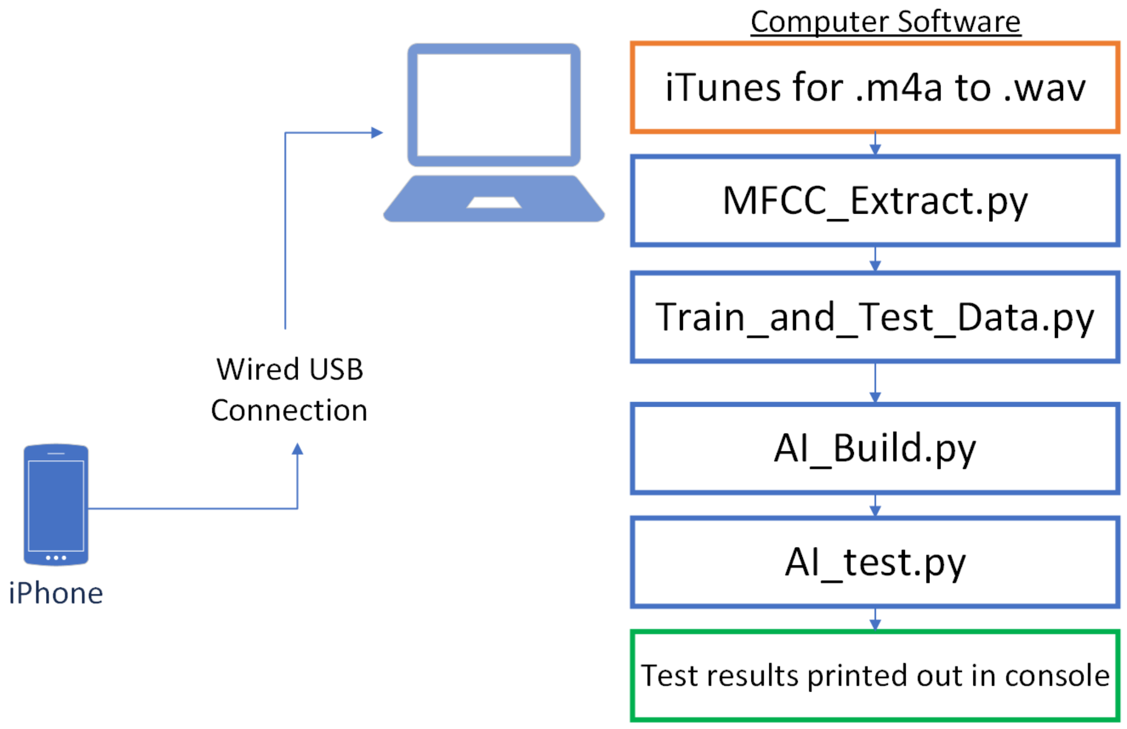


Figure 25: HurtzHunter iPhone Special Case Software Diagram

HurtzHunter requires iTunes [42] to download the audio files from the iPhone to the computer. To download, the user opens iTunes and plugs the iPhone into the computer via USB connection. iTunes syncs with the cell phone and the audio files are placed onto the computer. Then, the researcher uses iTunes to convert the audio files from .m4a to .wav. See Appendix A for directions on converting .m4a to .wav in iTunes. After conversion, the audio files exist in the computer's filing system under the iTunes folder in the .wav format. The files must be in a .wav format to be compatible with the Python scripts in HurtzHunter.

3.5 RodeVideoMicPro

Unlike the cellphones, the RVMP has one hardware configuration. This microphone does not have any on-device software, so it needs to be connected to a computer to gather recordings. However once recording is finished, the microphone does not need to be connected to the computer for further data processing because the audio files are stored on the computer during recording.

3.5.1 Hardware

Figure 26 shows the hardware configuration for the RVMP.

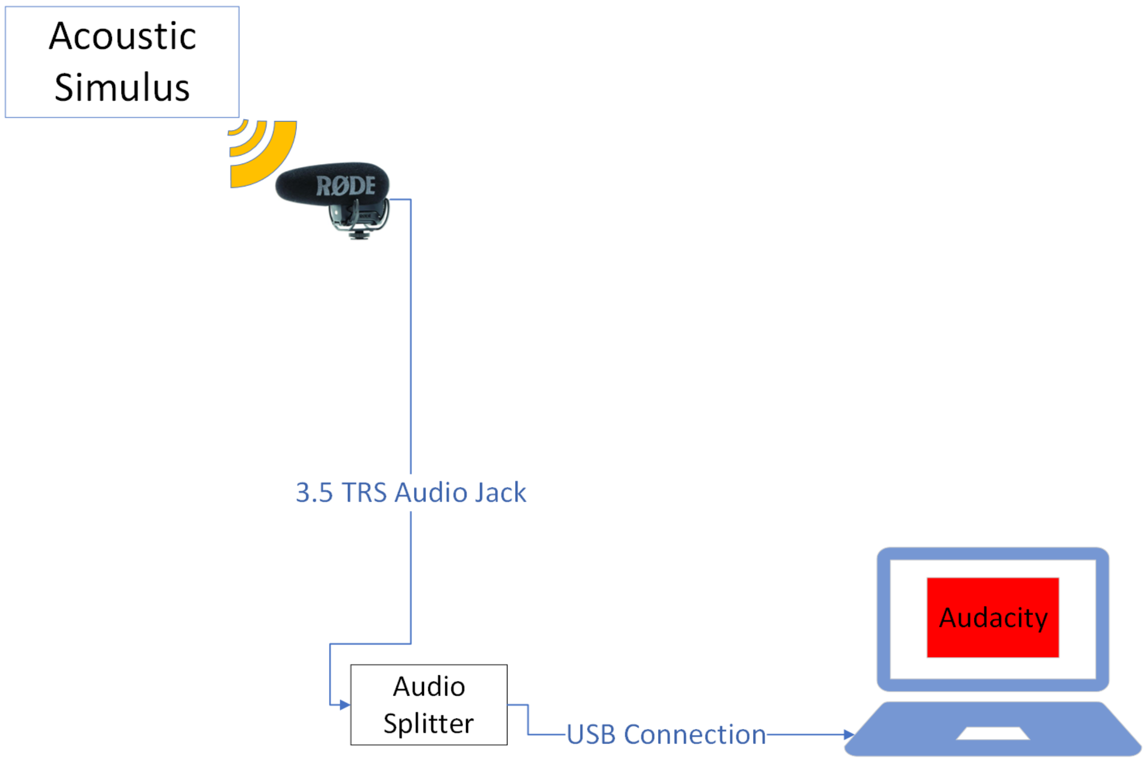


Figure 26: RodeVideoMicPro Hardware Configuration

The microphone connects to the TechRise USB External Stereo Sound Adapter across a TRS 3.5 mm audio chord. The sound adapter plugs into the USB of the computer. On the adapter, the sound control remains at full volume. The RVMP collects the audio with a sampling rate of 48 kHz.

3.5.2 Software

Figure 27 displays the Software configuration for the RVMP. Once the audio files are on the computer's filing system as a .wav, the process is the same as the phone process Section 3.4.2 discusses.

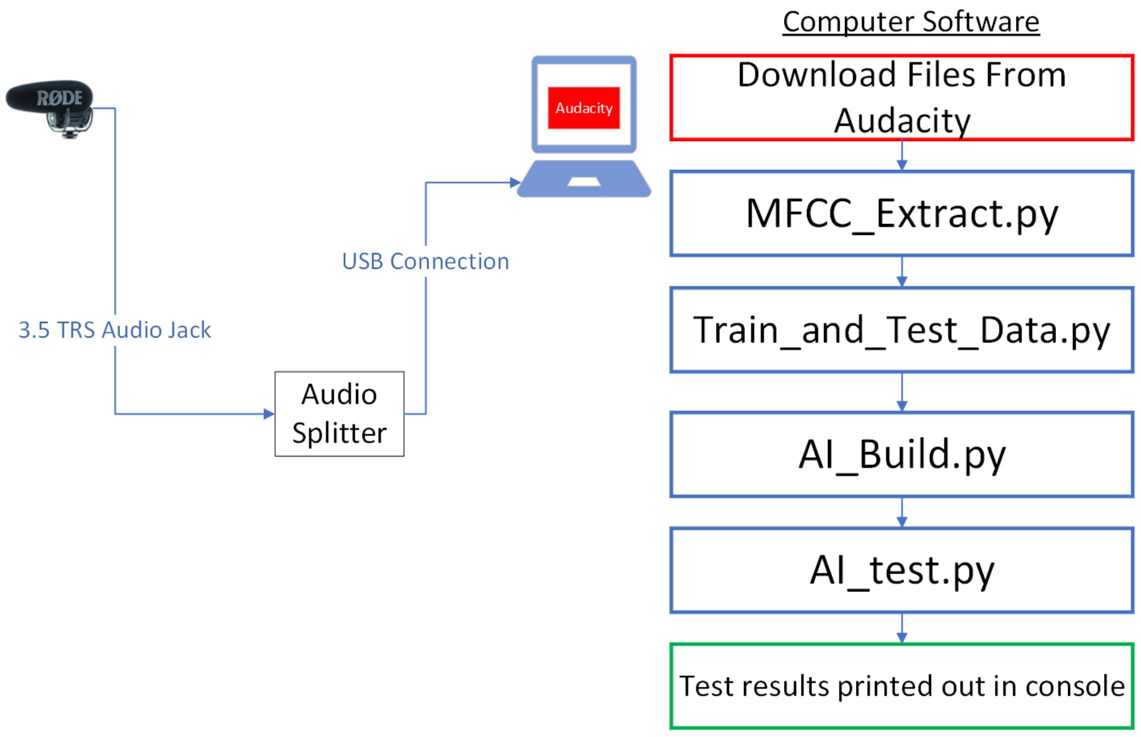


Figure 27: RodeVideoMicPro Software Diagram

HurtzHunter uses the Audacity recording software to control the microphone. The researcher manually creates audio files and presses start and stop recording for each data collect on the Audacity GUI. See Appendix B for detailed instructions on using Audacity. Audacity stores the audio collected by the RVMP on the computer. After recording, the researcher extracts the audio files from Audacity to the computer filing system as a .wav file. Once on the computer, the same Python scripts for the cell phones use the .wav files for data processing. The researcher manually runs the Python scripts inside Jupyter Notebook to build and test the drone detection AI.

3.6 HurtzHunter Python Scripts

This section discusses each Python script HurtzHunter uses for data processing. HurtzHunter uses Jupyter Notebook to run each Python script, and the researcher manually controls the process.

3.6.1 MFCC_Extract.py

MFCC_Extract.py extracts the MFCCs of each audio file. These MFCCs are the audio features HurtzHunter uses to train and test the AI. After training, this AI is able to identify drone payload weight by the MFCCs extracted from drone acoustic emissions. MFCC_Extract.py Section 1 shows the Python libraries in MFCC_Extract.py.

```
import wave
import pandas as pd
import librosa
import matplotlib.pyplot as plt
import librosa.display
import numpy as np
```

MFCC_Extract.py Section 1 Python Libraries used in MFCC_Extract.py

The command ”import” is the command in Python to add certain packages into the project. The first Python library, wave, includes the methods needed to open the .wav audio file into the Python script for data manipulation. pandas includes the data frame objects needed to create data frames of the MFCCs. Data frames are a Python object which holds information in the form of a table, similar to an excel sheet. This script extracts the 40 MFCCs for each frame and stores them in a data frame. librosa is the library which includes the audio processing methods essential for MFCC extraction. matplotlib.pyplot includes the methods for graphing the audio data and features. librosa.display includes the methods for graphing the audio spectrum. For instance, the script uses this library to display the mel-spectrogram of the audio data. The mel-spectrogram provides a visual representation of each audo file. The visual representation allows the researcher to visually check if the audio properly loads into the Python script. numpy is the package for creating arrays of the data and features.

After including the proper libraries, MFCC_Extract.py imports the audio wave files on the computer into the project. MFCC_Extract.py uses the commands in MFCC_Extract.py Section 2 shows to import these files.

```
Wave = wave.open('/Users/Documents/GalaxyS8/07.wav', 'rb')
sampleFrequency = Wave.getframerate()
sampleWidth = Wave.getsamplewidth()
frameNumber = Wave.getnframes()
```

MFCC_Extract.py Section 2 Importing Wave File To Python

The first command, Wave = wave.open('/Users/Documents/GalaxyS8 /07.wav', 'rb') imports the audio file stored at the location /Users/Documents/GalaxyS8 /07.wav'. The option 'rb' indicates the command to read the file into the object Wave. Wave is the Python object which stores the Wave audio data. The next commands in Section 2 use specific Wave object method to extract information from the Wave file. Wave.getframerate() stores the sampling frequency in the variable sampleFrequency. Wave.getsamplewidth() returns the number of bytes in each sample. Wave.getnframes() returns the number of samples in the audio files.

MFCC_Extract.py Section 3 displays the MFCC extraction settings. The script stores each setting in a variable and uses the variables to set the frame length, minimum frequency, maximum frequency, and number of MFCCs.

```
frame_stride = .04
frame_length = sample_rate*frame_stride
frame_length = int(frame_length)
win_len = None
hop_stride = frame_stride/2
hop_length = sample_rate*hop_stride
hop_length = int(hop_length)
n_mfcc = 40
n_mels = 40
```

MFCC_Extract.py Section 3 MFCC Extraction Settings

To extract MFCCs from each audio file, MFCC_Extract.py divides the audio file into 40 ms frames. HurtzHunter assumes 40 ms of recording equates to constant sound. In other words, the frequencies across 40 ms should not change. So, the system extracts MFCCs for each unit of constant sound. frame_stride sets the frame length to 40 ms. Then, the command frame_length = sample_rate*frame_stride determines the number of samples required to fill a 40 ms frame.

Hop stride refers to when the next audio frame begins. Hop stride is half of the frame stride. Therefore, the command hop_length = sample_rate*hop_stride determines the number of samples required to fill one hop. Each hop is 20 ms in time. Using half of the frame length as the hop length is common practice for smoothing audio results. In general, greater overlap between frames renders smoother Fourier Transforms. However, greater overlap leads to more audio frames which requires more computational resources. Using a 40 ms frame length and 20 ms hop stride results in 2999 frames for each audio recording because each recording is 60 s long. So, this script creates 2999 sets of MFCCs for each audio file. The calculation for 2999 frames is:

$$\text{number of frames} = \frac{\text{recording time}}{\text{hop length}} - 1 = \frac{60}{.04} - 1 = 2999 \tag{5}$$

The next variable, n_mfcc determines the number of MFCCs to extract for each frame. There are 40 MFCCs, so MFCC_Extract.py collects all 40 MFCCs. The final variable, n_mels sets the number of filter banks needed for MFCC extraction. As Chapter 2 discusses, mel filter banks place the spectrum in perceptually relevant frequency ranges. The number of mel filter banks to use is determined by the number of MFCCs. Since there are 40 MFCCs, this script uses 40 mel filter banks.

MFCC_Extract.py Section 4 shows the portion of code which builds the mel-spectrogram. As Chapter 2 discusses, a mel-spectrogram is a visual representation of the audio frequencies present in a signal across time. Mels is the unit on the y-axis, and time is on the x-axis.

```
df_final = pd.DataFrame()
audio_path = '/Users/Documents/GalaxyS8/07.wav'
audio_data = librosa.load(audio_path,sr =sample_rate, duration = 60)
audio_values_array = np.asarray(audio_data[0])
mel_spectrogram = librosa.feature.melspectrogram(y=audio_values_array,
    sr=sample_rate, S=None, n_fft=frame_length, hop_length=hop_length,
    win_length=None, window='hann', center=False, pad_mode='reflect',fmin
    = 0, fmax = None, power=2.0, n_mels = 128)
```

MFCC_Extract.py Section 4 Mel-Spectrogram Code

The first command df_final = pd.DataFrame() creates an empty data frame where the final MFCCs are stored. The command audio_path = '/Users/Documents/GalaxyS8/07.wav' saves the file path. The command librosa.load(audio_path, sr =sample_rate, duration = 60) uses this path to extract all the audio samples into a numpy array. The parameter duration = 60 loads only the first 60 seconds of audio data from the file. As Chapter 4 discusses, only the first minute of recording holds the important information. This array stores many characteristics of the audio data. The zero index holds the actual audio bits. The command audio_values_array = np.asarray(audio_data[0]) stores just the audio bits in the variable audio_values_array. Then, the command mel_spectrogram = librosa.feature.melspectrogram(y=audio_values _array, sr=sample_rate, S=None, n_fft =frame_length, hop_length =hop_length, win_length =None, window ='hann', center=False, pad_mode ='reflect',fmin = 0, fmax = None, power=2.0, n_mels = 128) creates a mel-spectrogram of the audio.

Lastly, MFCC_Extract.py uses the mel-spectrogram to extract the MFCCs. MFCC_Extract.py Section 5 shows the commands to extract the MFCCs.

```
1: phone_mfccs = librosa.feature.mfcc(S= librosa.power_to_db(mel_spectrogram),
   n_mfcc= n_mfcc)
2: df2 = pd.DataFrame('MFCC #1': phone_mfccs[0,:], 'MFCC #2':
   phone_mfccs[1,:], 'MFCC #3': phone_mfccs[2,:], 'MFCC #4': phone_mfccs[3,:],
   'MFCC #5': phone_mfccs[4,:], 'MFCC #6': phone_mfccs[5,:], 'MFCC #7':
   phone_mfccs[6,:], 'MFCC #8': phone_mfccs[7,:], 'MFCC #9': phone_mfccs[8,:],
   'MFCC #10': phone_mfccs[9,:], 'MFCC #11': phone_mfccs[10,:],
   'MFCC #12': phone_mfccs[11,:],'MFCC #13': phone_mfccs[12,:], 'MFCC
   #14': phone_mfccs[13,:], 'MFCC #14': phone_mfccs[13,:], 'MFCC
   #15': phone_mfccs[14,:], 'MFCC #16': phone_mfccs[15,:], 'MFCC
   #17': phone_mfccs[16,:], 'MFCC #18': phone_mfccs[17,:], 'MFCC
   #19': phone_mfccs[18,:], 'MFCC #20': phone_mfccs[19,:], 'MFCC
   #21': phone_mfccs[20,:], 'MFCC #22': phone_mfccs[21,:], 'MFCC
   #23': phone_mfccs[22,:], 'MFCC #24': phone_mfccs[23,:], 'MFCC
   #25': phone_mfccs[24,:], 'MFCC #26': phone_mfccs[25,:], 'MFCC
   #27': phone_mfccs[26,:], 'MFCC #28': phone_mfccs[27,:], 'MFCC
   #29': phone_mfccs[28,:], 'MFCC #30': phone_mfccs[29,:], 'MFCC
   #31': phone_mfccs[30,:], 'MFCC #32': phone_mfccs[31,:], 'MFCC
   #33': phone_mfccs[32,:], 'MFCC #34': phone_mfccs[33,:], 'MFCC
   #35': phone_mfccs[34,:], 'MFCC #36': phone_mfccs[35,:], 'MFCC
   #37': phone_mfccs[36,:], 'MFCC #38': phone_mfccs[37,:], 'MFCC #39':
   phone_mfccs[38,:], 'MFCC #40': phone_mfccs[39,:])
3: df_final = df2;
4: display(df2)
5: df_final.to_pickle('07_GalaxyS81-1_df.pkl')
```

MFCC_Extract.py Section 5 MFCC Extraction

The command in line 1 creates the 40 MFCCs from the mel-spectrogram. The reason the script uses the mel-spectrogram to extract MFCCs is because the mel-spectrogram input does not zero pad. Zero-padding occurs when there is not enough information inside a frame to fit certain requirements, so zeros are added to the beginning and end of an audio sample to meet the requirements. This library does not zero pad for the mel-spectrogram but does zero pad for other audio formats. So, this script uses the mel-spectrogram to ensure zero-padding does not occur.

After creating MFCCs, the command in line 2 stores the 40 MFCCs in the data frame df2. The librosa methods save the MFCCs with a zero index, while the data frame requires a one indexed sequence. So, the 'MFCC #NUMBER' is one higher than the index location in phone_mfccs[NUMBER,:]. Next, the next three lines of code store the data frame containing the MFCCs to df_final. Then, the script stores the df_final to a pickle file on the computer's file system. The purpose of a pickle file is to hold Python objects on the file system, so other scripts can load the object and use it. MFCC_Extract.py stores the MFCC data frame as a pickle file, so the other Python scripts can import the data frame for their use.

3.6.2 Train_and_Test_Data.py

Train_and_Test_Data.py imports the MFCCs generated by MFCC_Extract.py and forms train and test arrays to build the AIs. HurtzHunter uses 80% of the data to train the ML and 20% of the data to test the data. The reason for using these portions to test and train the ML is to achieve 5-cross validation, which is a common technique used to assess ML learning algorithms. The technique prevents accidentally choosing really good or really bad testing and training data, and in so doing, provides a robust assessment of the ML algorithm [43]. Figure 28 shows the k-cross validation process.

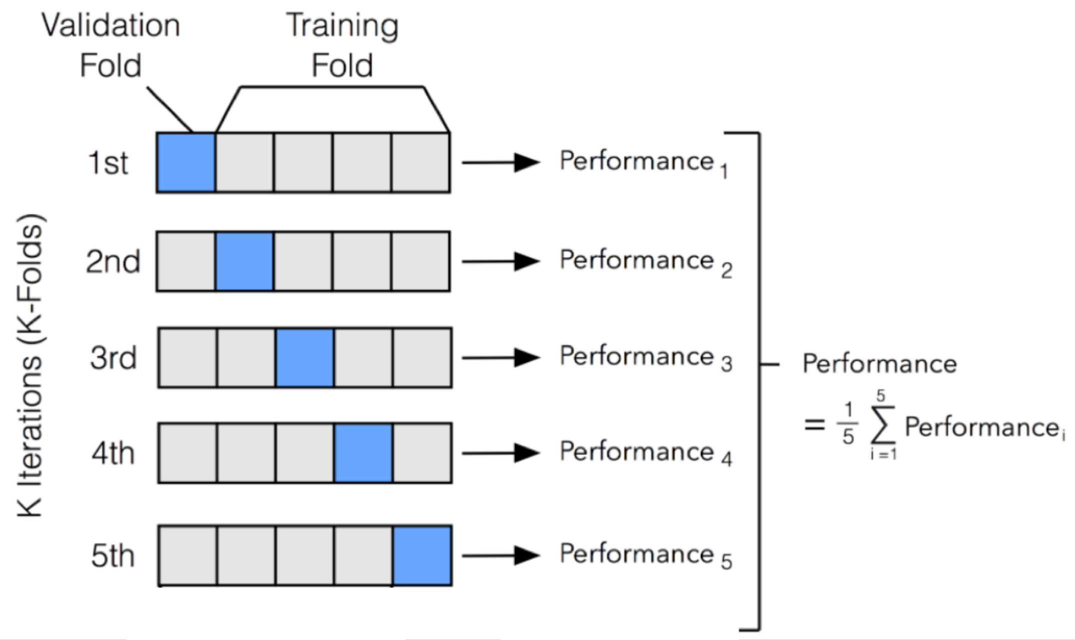


Figure 28: k-Cross Validation Diagram When K = 5 [44]

5-cross validation uses 5 different data configurations to train a ML algorithm and averages all 5 performance results together. As Figure 28 shows, the data is split into 5 folds. For each iteration, the validation data, also called the testing data, is from a different fold than the previous iteration. After all k-iterations are used to assess the ML, the performance metrics from each iteration are averaged. This process ensures randomly good or bad data is not used to assess the performance of an ML. By using all parts of the data for testing and training, the performance metric is a robust measurement.

The value of k determines the number of folds and iterations. 5-cross means 80% of the data is used for training, and 20% for testing. HurtzHunter uses 5-cross validation.

Train_and_Test_Data.py Section 1 shows the Python libraries used in this script.

```
import pandas as pd
import numpy as np
import pickle
```

Train_and_Test_Data.py Section 1 Python Libraries Imported

The first library, pandas is the Python library needed to create and manipulate data frames. numpy is the Python libary needed to use numpy arrays. The last library, pickle, is the library needed to pickle the training and testing objects to a file on the computer.

The next section of Train_and_Test_Data.py loads the pickled MFCC data frames into the Python project. Train_and_Test_Data.py Section 2 shows this the portion of this script which loads the pickled MFCC data frames created by MFCC_Extract.py.

```
1: with open('0100_GalaxyS81-1_df.pkl', 'rb') as f:
2:       Gram0_df = pickle.load(f)
```

Train_and_Test_Data.py Section 2 Load MFCC Pickled Files

The pickled MFCC data frame exists at the file location '0100_GalaxyS81-1_df.pkl'. Line 1 of Section 2 opens this file location as f, and line 2 loads the MFCC data frame from the pickle file into the object Gram0_df. Gram0_df holds the MFCC data frame created in the script MFCC_Extract.py.

Train_and_Test_Data.py Section 3 separates the MFCCs into 5 data frames. The script uses these 5 data frames as the 5 folds for 5-cross validation.

```
1: MFCC = ['MFCC #' + str(i) for i in range(1,41)]
2: section1_data_0100 = pd.DataFrame(columns=[*MFCC])
3: section2_data_0100 = pd.DataFrame(columns=[*MFCC])
4: section3_data_0100 = pd.DataFrame(columns=[*MFCC])
5: section4_data_0100 = pd.DataFrame(columns=[*MFCC])
6: section5_data_0100 = pd.DataFrame(columns=[*MFCC])
7: Gram0DataSize= Gram0_df.index.size
8: sectionDataSize = int((Gram0DataSize)/5)
```

Train_and_Test_Data.py Section 3 Test and Train Data Frames

In Section 3, line 1 creates an array of MFCC labels for the columns of each data frame. Lines 2-6 create the five data frames. Each data frame holds one fifth of the total MFCCs. In line 7, the variable Gram0DataSize holds the total number of MFCC arrays. In line 8, sectionDataSize holds the total number of MFCC arrays divided by 5. The script uses sectionDataSize to put the right amount of MFCC arrays in each data frame.

Train_and_Test_Data.py Section 4 shows the for loop used to build each data frame. While section 4 only shows two iterations of the for loop, there are 5 in total, one for each fold.

```
1: index = 0
2: for i in range(0,Gram0DataSize): do
3:     if (i < sectionDataSize): then
4:         new_df = pd.DataFrame(columns=[*MFCC], index = [index])
5:         new_df.loc[index, 'MFCC #1':'MFCC #40'] = Gram0_df.iloc[i]
6:         section1_data_0100 = pd.concat([section1_data_0100, new_df])
7:         index = index +1
8:     else if ((i <= sectionDataSize) and (i <(sectionDataSize + sectionData-
   Size))): then
9:         new_df = pd.DataFrame(columns=[*MFCC], index = [index])
10:        new_df.loc[index, 'MFCC #1':'MFCC #40'] = Gram0_df.iloc[i]
11:        section2_data_0100 = pd.concat([section2_data_0100, new_df])
12:        index = index +1
13:    end if
14: end for
```

Train_and_Test_Data.py Section 4 Test and Train Data Frames

The variable sectionDataSize is the size each data frame fold is once all the data is divided among the 5 data frame folds. The script uses if statements to decide which section of the data goes in each fold. Inside each if statement, the script places an array of MFCCs in a new data frame, and the new data frame is concatenated onto the proper fold.

Train_and_Test_Data.py Section 5 displays the commands used to create each iteration of the 5-cross validation.

```
1: train1_data_0100 = pd.DataFrame(columns=[*MFCC])
2: train1_data_0100 = pd.concat([train1_data_0100, section2_data_0100, section3_data_0100, section4_data_0100, section5_data_0100])
3: test1_data_0100 = pd.DataFrame(columns=[*MFCC])
4: test1_data_0100 = pd.concat([test1_data_0100, section1_data_0100])
5: train2_data_0100 = pd.DataFrame(columns=[*MFCC])
6: train2_data_0100 = pd.concat([train2_data_0100, section1_data_0100, section3_data_0100, section4_data_0100, section5_data_0100])
7: test2_data_0100 = pd.DataFrame(columns=[*MFCC])
8: test2_data_0100 = pd.concat([test2_data_0100, section2_data_0100])
```

Train_and_Test_Data.py Section 5 Test and Train Data For Cross Validation Iterations

In Section 5, lines 1 and 2 create the train data for iteration one of the 5-cross validation. Lines 3 and 4 create the test data for iteration one. This process repeats in lines 4-8, but for iteration two. The folds of data used for training and testing in each iteration follow Figure 28 where iteration one is the first row in the figure, and iteration two is the second row. Section 5 shows the code for creating the first two iterations. This process repeats for iterations three, four, and five.

Train_and_Test_Data.py Section 6 shows the code for storing all the testing and training data in a pickle file.

```
1: train1_data_0100.to_pickle('GalaxyS81-1_0g_100m_train1_data')
2: test1_data_0100.to_pickle('GalaxyS81-1_0g_100m_test1_data')
3: train2_data_0100.to_pickle('GalaxyS81-1_0g_100m_train2_data')
4: test2_data_0100.to_pickle('GalaxyS81-1_0g_100m_test2_data')
```

Train_and_Test_Data.py Section 6 Train and Test Pickle Files

Section 6 shows the commands for pickling the first two iterations of data. This process repeats for each iteration. Lines 1 and 2 pickle the test and train data for iteration one, and lines 3 and 4 pickle the data for iteration two. The script AI_Build.py uses these pickle files to build the AIs.

The process in Train_and_Test_Data.py Sections 2-6 repeats for each audio recording.

3.6.3 AI_Build.py

The Python script AI_Build.py creates the AI capable of UAV payload detection. The AI is a Support Vector Machine (SVM). The reason HurtzHunter uses the SVM is because past researchers concluded that the SVM is an appropriate ML for acoustic payload detection [7]. AI_Build.py Section 1 shows the libraries needed to create the SVM.

```
from sklearn import svm
import numpy as np
import pandas as pd
```

AI_Build.py Section 1 Libraries Used To Build AI

The first library, sklearn, includes the packages needed to build an SVM object [45]. numpy includes the methods for using numpy arrays. pandas includes the methods for using data frames.

AI_Build.py Section 2 shows the commands for importing the testing and training data created by Train_and_Test_Data.py.

```
1: with open('GalaxyS81-1_0g_7m_train1_data', 'rb') as g:
2:       Gram0_7m_df_train1 = pickle.load(g)
```

AI_Build.py Section 2 Import Test and Train Pickle Files

Line 1 opens the file at GalaxyS81-1_0g_7m_train1_data as the variable g with read permissions. Line 2 loads the object in the file to the variable Gram0_7m_df_train1. This process repeats for all training data.

AI_Build.py Section 3 creates the class identity which correspond to each array of MFCCs. To train the SVM, HurtzHunter submits an array of MFCCs with the corresponding payload weight. The payload weight is the class identity. Section 3 shows the code for creating the array of corresponding payload weights for 0 grams.

```
1: y0 = np.array([])
2: for i in range(0,size):
3:       y0= np.append(y0,[0])
```

AI_Build.py Section 3 Create Training Values For Zero Gram

Line 1 creates an empty array which holds the 0 gram values. Lines 2 and 3 create an array of zeros to correspond with each set of 0 gram MFCCs. This process repeats for each payload weight recorded by the microphones.

After importing all the MFCC data and creating the arrays of weight values in Section 3, AI_Build.py concatenates all training data into train1X and train1y in Section 4. train1x holds the MFCC values and train1y holds the corresponding weight values in grams. Then, AI_Build.py builds the SVMs. AI_Build.py Section 4 shows the commands for building the SVM.

```
1: lsvm1 = svm.SVC(kernel = 'linear')
2: lsvm1.fit(train1X,train1y)
```

AI_Build.py Section 4 Support Vector Machine Creation

Line 1 creates an untrained SVM with a linear kernel called lsvm1. Line 2 trains this SVM with the training data for iteration one. This process repeats for each iteration of 5-cross validation. After training, the script pickles the SVM to a file on the computer. The script AI_Test.py uses the AI in this file to evaluate the AI's performance.

3.6.4 AI_Test.py

AI_Test.py evaluates the performance of the SVM created by AI_Build.py. This script prints the prediction accuracy from 5-cross validation to the console. AI_Test.py uses the same Python libraries as shown in AI_Build.py Section 1 to test the SVM. AI_Test.py loads the pickled testing data files from Train_and_Test_Data.py with the same commands AI_Build.py Section 2 uses. This script also loads the SVMs created by AI_Build.py using the same commands AI_Build.py Section 2 uses to load pickle files.

AI_Test.py uses for loops to iterate through all testing data. AI_Test.py Section 1 shows the testing process.

```
for i in range(0,Gram0_7m_df_test1.index.size): do
    test1 = np.array(test1_data0_7m_array[i])
    result = lsvm1.predict([test1])
    if result == 0: then
        correct = correct +1
    else
        incorrect = incorrect +1
        df1 = pd.DataFrame([[0, result]], columns = (['0g actual', '0g pred']),
        index = [i])
        incorrect_df = incorrect_df.append(df1, ignore_index = True)
    end if
end for
```

AI_Test.py Section 1 Testing Loop

In Section 1, the for loop iterates through all testing samples in the data frame Gram0_7m_df. In each iteration, the script gives an array of MFCCs to the AI which makes a weight prediction in grams, and the script stores the weight prediction in result. The loop then compares the prediction to the correct payload value. If the prediction is correct, the variable correct is incremented. If not, lines 7-10 increment the variable incorrect and store the incorrect prediction in the data frame incorrect_df. The loop iterates until the script uses all testing data. Then, the script uses the variables incorrect and correct to calculate prediction accuracy.

AI_Test.py Section 2 shows the code for calculating the prediction accuracy.

```
1: accuracy_07_test1 = (correct/(correct+incorrect))
2: print('0g 7m k-cross test1 accuracy: ', accuracy_07_test1)
```

AI_Test.py Section 2 Prediction Accuracy

The command in line 1 divides the number of correct predictions by the total number of predictions. The command in line 2 prints the accuracy to the console. AI_Test.py repeats this process for all testing data.

Because HurtzHunter uses a 40 ms frame length and a 20 ms hope stride, the number of MFCC arrays for each 60 s audio recording is 2999. Using 5-cross validation for 2999 samples results in 599 full MFCC arrays for testing. This calculation is:

$$\frac{2999}{5} = 599.8 \tag{6}$$

The HurtzHunter design requires full sets of data, so 599.8 is floored to 599. Therefore, the accuracy of each iteration is:

$$\frac{\#\text{ of correct predictions}}{599} = \text{AI Accuracy} \tag{7}$$

After assessing all testing data, AI_Test.py averages all iterations to achieve 5-cross validation. AI_Test.py Section 3 shows the code for 5-cross validation.

```
1: totalAccuracy_07 = (accuracy_07_test1 + accuracy_07_test2 + accuracy_07_test3
   + accuracy_07_test4+ accuracy_07_test5)/5
2: print('0g 7m total accuracy:', totalAccuracy_07)
```

AI_Test.py Section 3 5-Cross Validation Results

Line 1 averages all iterations from cross validation and line 2 prints the results to the console. The commands in AI_Test.py Section 1-3 repeats for all test data. Once this script is complete, the researcher manually records the accuracy results from the console for presentation.

3.7 HurtzHunter Execution Process

This section describes the process to use HurtzHunter.

1. Manually set up all devices for recording.
2. Open Voice Memos on the iPhone and SmartRecorder on the androids.
3. Connect the RVMP to the laptop. Connect the 3.5 mm TRS cable to the RVMP and to the audio splitter. Then, connect the audio splitter to the laptop.
4. Open Audacity on the laptop. See Appendix B for Audacity use instructions.
5. Attach the proper payload weight to the drone. Instruct the pilot to fly the drone to the proper distance and height.
6. Manually begin recording by pressing record on all recording software: Voice Memos, SmartRecorder, and Audacity.

7. Wait desired recording time. This research uses HurtzHunter for 60 s of recording.
8. Manually stop recording. Press end recording on all recording software.
9. Stop the flight. Instruct the pilot to end the flight.
10. Repeat steps 5-9 until are recordings are captured.
11. Save the Audacity project
12. At this point, recording is no longer needed, so the set up may be taken down.
13. Download all recordings to the computer,
14. Extract recordings from Audacity (see Appedix B) to the computer's file system.
15. Use iTunes to convert the .m4a files from the iPhone to .wav files. See Appendix A for instructions.
16. Manually run all Python scripts.
17. Record the results printed to the console.

3.8 Design Summary

This chapter discusses the HurtzHunter prototype this research develops. Section 3.2 provides a summary of the HurtzHunter design and its components. Section 3.3 describes the design goals of HurtzHunter. Section 3.4 discusses the hardware and software components of the cell phones in the design. Section 3.5 discusses the hardware and software components of the high-quality microphone used in this design. Section 3.6 discusses the Python scripts HurtzHunter uses to process the collected audio data. Finally, Section 3.7 provides the steps to use the prototype design.

IV. Methodology

4.1 Overview and Objectives

This research extends the field of study by focusing on the following questions:

- Can cell phones detect UAV payload weight with sound?
- What is the maximum range of acoustic UAV payload detection?
- Does the cell phone model affect detection accuracy?
- Does a high-quality microphone perform better than a cell phone for acoustic payload detection?
- Does the manufacturing process for a specific recording device affect detection accuracy?

Section 4.2 explains the System Under Test (SUT). Sections 4.3, 4.4, 4.5, 4.6 discuss the factors, metrics, constant parameters and uncontrollable variables of the system. Section 4.7 shows the experiment design. Section 4.8 discusses the case studies in this research. Section 4.9 discusses the timer used for tracking runtime in the experiment. Section 4.10 discusses the wind noise measuring technique. Section 4.11 discusses the statistical analysis.

4.2 System Under Test

The HurtzHunter Prototype which Chapter 3 describes is the System Under Test (SUT) designed to answer the questions listed in Section 4.1. This system explores acoustic payload detection capabilities. Section 3.7 describes the execution process for using the HurtzHunter system. Figure 29 shows the SUT.

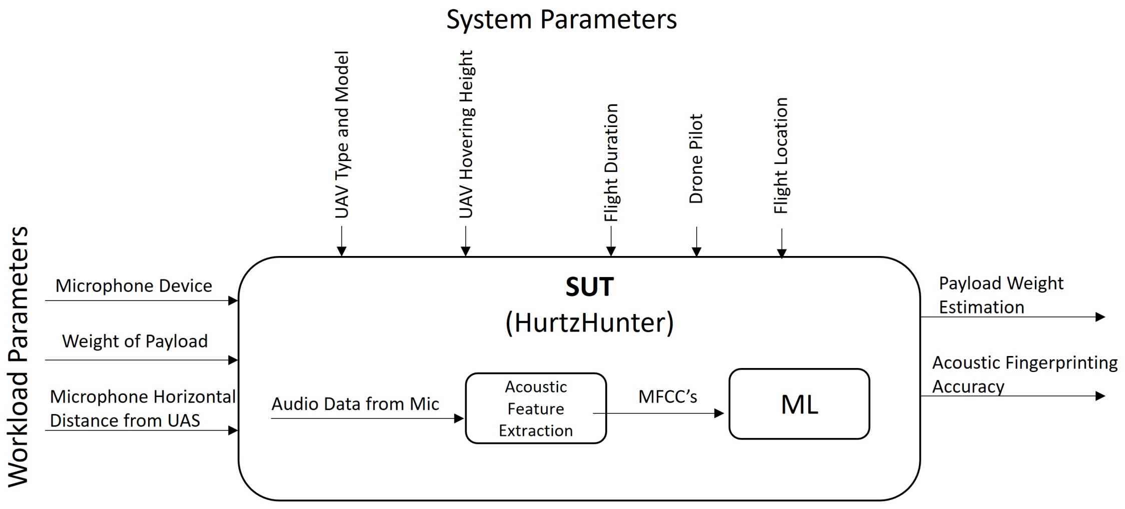


Figure 29: System Under Test

4.3 Factors

The factors in this experiment are microphone device, payload weight, and ground distance:

- Microphone Device: The device which collects the acoustic emissions from the drone. HurtzHunter uses the following recording device: one Samsung Galaxy S20 (GS20), three Samsung Galaxy S8s (GS8), one Google Pixel 4 (GP4), one iPhone 11 Pro (iP11P), and one RodeVideoMicPro (RVMP).
- Payload Weight: The drone payload weight varies across experiments. The different weights are 0 g, 56 g, 112 g, 168 g, 224 g, 280 g.
- Ground Distance: The ground distance between the drone and the microphones varies throughout experimentation. The ground distances are 7 m, 10 m, 20 m, 30 m, 40 m, 50 m, 75 m, 100 m.

Past research in [7] used payload weights in 50 g increments. This research follows this methodology to provide experiment validation. The reason the weights are in 56 g increments rather than 50 g is because the sponsor of this research only supplied weights in 1 oz increments. So, the closest weight to 50 g using 1 oz increments is 56 g. To provide context for these types of payload weights, Table 4 shows real-life comparable payloads which could be carried by the DJI Mavic Air 2 [46]. This research uses the Mavic Air 2 for all experiments. The DoD is particular interested in payload detection for the Mavic Air 2 because DJI is a Chinese drone company [47].

Table 4: Real Life Payload Weights

Device	**Weight**	**Blast Radius**
v40 Grenade [48]	136 g	5 m
Stick of Dynamite [49]	190 g	4.3 m
Go Pro Camera [50]	117 g	N/A

This drone is capable of carrying all the weights listed in Table 4. The maximum payload weight for this UAV is 300 g [46]. These examples show how weight identification provides important information to aid in detecting hazardous drones.

4.4 Metrics

Table 5 shows the metrics in the SUT. This research uses these metrics to infer the success of the system. The last Python script in HurtzHunter, AI_Test.py, creates these metrics.

- Payload Weight Prediction: This metric represents the payload weight the AI believes the drone is carrying based on the MFCC features fed to the algorithm. The AI predicts one of the six payloads flown by the drown in the experiment: 0 g, 56 g, 168 g, 224 g, 280 g.

- ML Prediction Accuracy: This metric conveys the accuracy of the system. For each trial, the total number of predictions divides the total number of correct predictions to produce the accuracy of the ML. Equation 8 shows this calculation.

$$\frac{\text{Total Correct Predictions}}{\text{Total Predictions}} * 100 = \text{Percent Accuracy} \tag{8}$$

Table 5: Metrics

Metric	**Unit**	**Expected Range**
Payload Weight Prediction	gram (g)	0 g, 56 g, 112 g, 168 g, 224 g, 280 g
ML Prediction Accuracy	percent	0-100%

4.5 Constant Parameters

The following parameters remain constant during experimentation. These parameters are constant to support the reliability of the results. This practice prevents an impact on results from the environment or other factors not listed in the SUT. Table 6 shows the constant parameters.

Table 6: Constant Parameters

Parameters	**Proposed Values**	**Controlled By**
UAV Model	DJI Mavic Air 2	Experiment Design
Hovering Height	2-3 m	Experiment Design
Flight Duration	60 s	Experiment Design
Drone Pilot	AFRL/RYAR Pilot	Experiment Design
Flight Location	Avon Park, FL	Experiment Design

- UAV Model: The DJI Mavic Air 2. DJI is a Chinese company, and the Department of Defense is particularly interested in counter drone research focused on Chinese drones. The maximum payload weight for the DJI Mavic Air 2 is 300 g.
- Hovering Height: The experiments record acoustics while the drone hovers at 2-3 m Above Ground Level (AGL).
- Flight Duration: Each experiment records in-flight acoustics for 60 seconds.
- Drone Pilot: AFRL/RYAR has pilots certified to fly Chinese drones for DoD research. This lab supplied a pilot to fly all experiments.
- Flight Location. AFRL/RYAR is certified to fly the Mavic Air 2 at the Air Force Bombing range in Avon Park, FL. All tests are conducted at the bombing range.

4.6 Uncontrollable Variables

Wind and time of day are the uncontrollable variables in this experiment. These variables likely affect the results of this research, and future work is needed to fully understand the impact of wind and time of day on acoustic payload detection.

Due to the nature of this research, there is only one repetition of the experiment. There was not enough flight time in Avon Park, FL to complete more than on repetition. Additionally, battery power limits the experiment. There was not enough battery power to accomplish all data collects at the same time of day. Weather such as wind varied through out the day. So, some audio recording have much greater wind noise than others. The presence or absence of wind noise does affect the AIs prediction accuracy. Chapter 5 discuses this impact. In short, wind and time of day are two uncontrollable factors which have a negative impact on the experiment, and because only one repetition of the experiment could be accomplished, little can be done to mitigate the effect of these uncontrollable variables.

4.7 Experiment Design

This section describes the experiments in this research. The experiments seek to answer the research questions listed in Section 4.1:

- Can cell phones detect UAV payload weight with sound?
- What is the maximum range of acoustic UAV payload detection?
- Does the cell phone model affect detection accuracy?
- Does a high-quality microphone perform better than a cell phone for acoustic payload detection?
- Does the manufacturing process for a specific recording device affect detection accuracy?

This experiments use the HurtzHunter design Chapter 3 discuses to answer these questions. Figure 30 shows the experiment diagram.

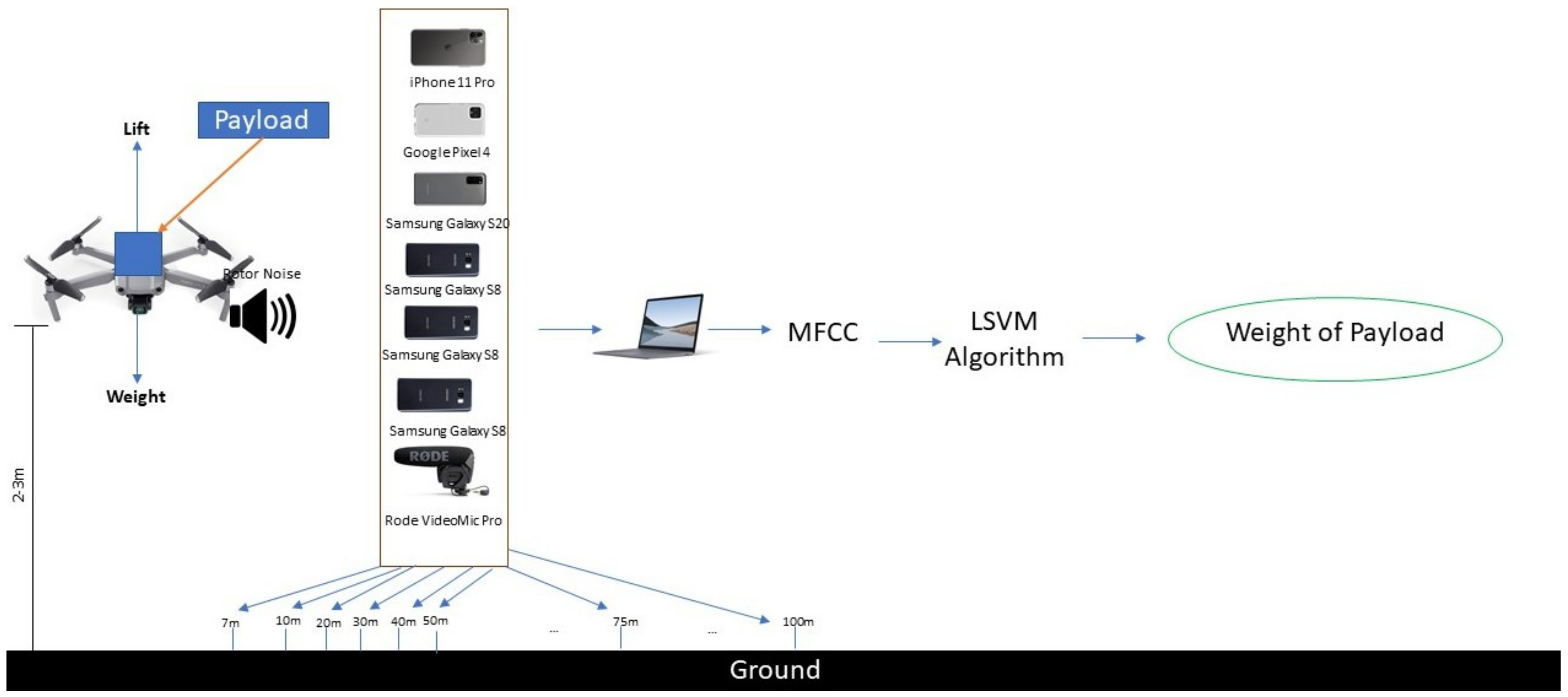


Figure 30: Experiment Diagram

1. The microphones are 1 m above the ground, horizontally oriented, pointing towards where the drone flies. Figure 31 and 32 show this setup. This research assumes the bottom microphone on each cell phone is the optimal MEMS microphone for recording, so this microphone points towards the acoustic source.

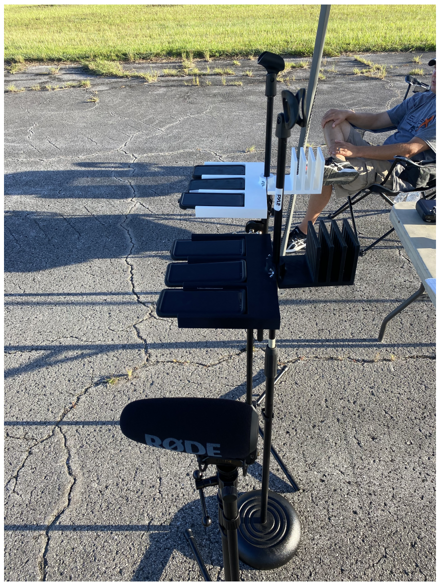


Figure 31: Microphone Setup

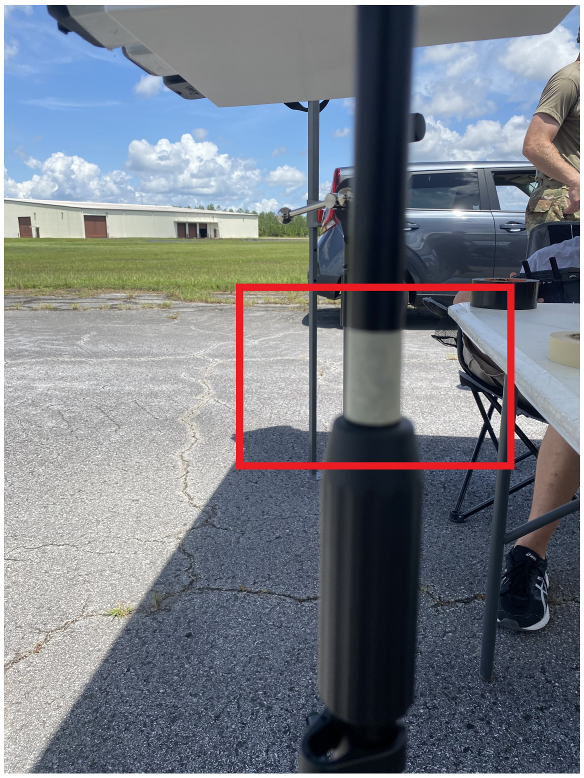


Figure 32: Tape Used To Set 1 m Height

2. The laptop running Audacity is on the table behind the microphones. The RVMP connects to the laptop. Figure 33 shows this step.

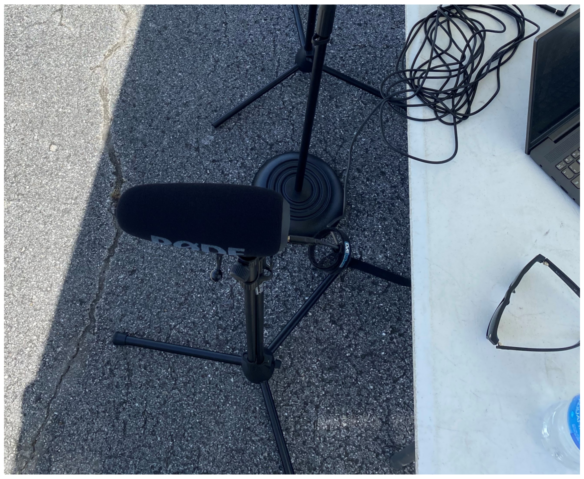


Figure 33: RVMP Setup

3. The UAV has the payload attached. Figure 34 shows this configuration. Figure 35 shows the payload.

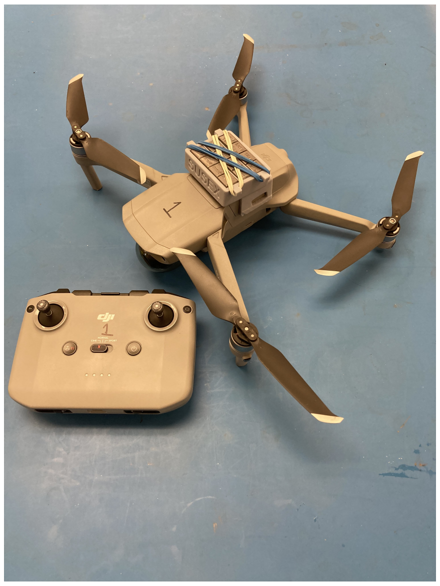


Figure 34: Drone Mounted With Payload Basket

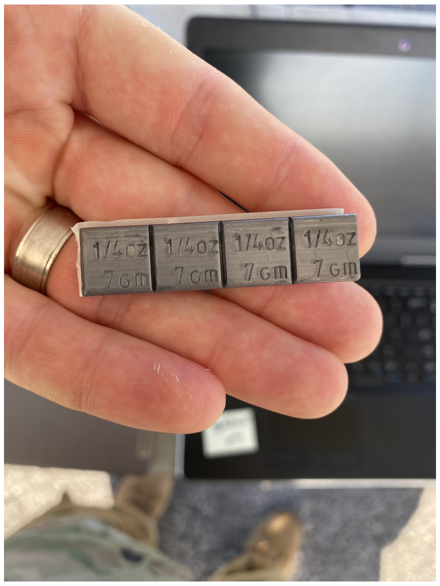


Figure 35: Payload Weights

4. The pilot flies the drone to 7 m. The drone hovers at 2-3 m for 60 seconds of audio recording. The researcher manually starts and stops recording. Figure

36 and 37 show this step.

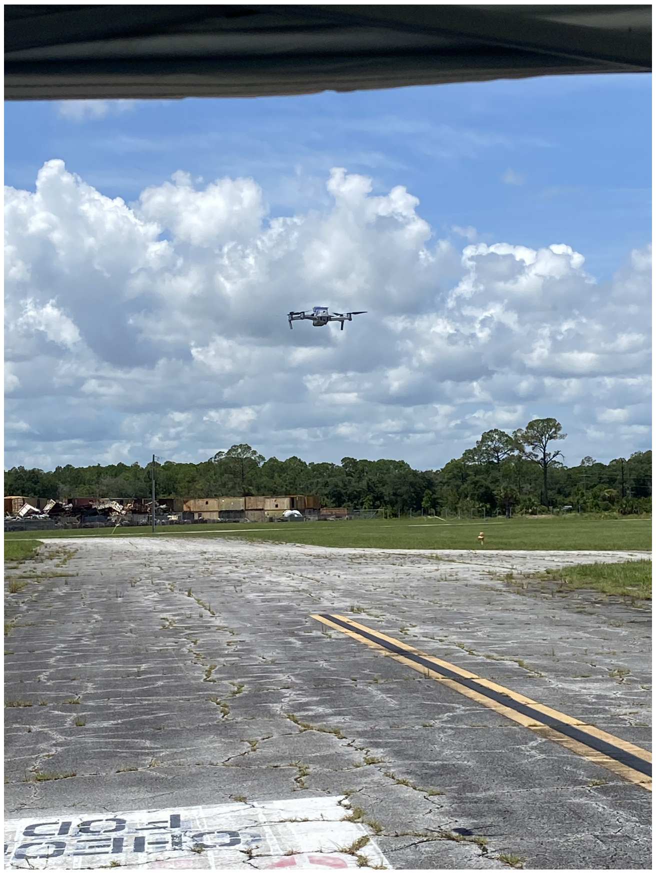


Figure 36: Drone In Flight

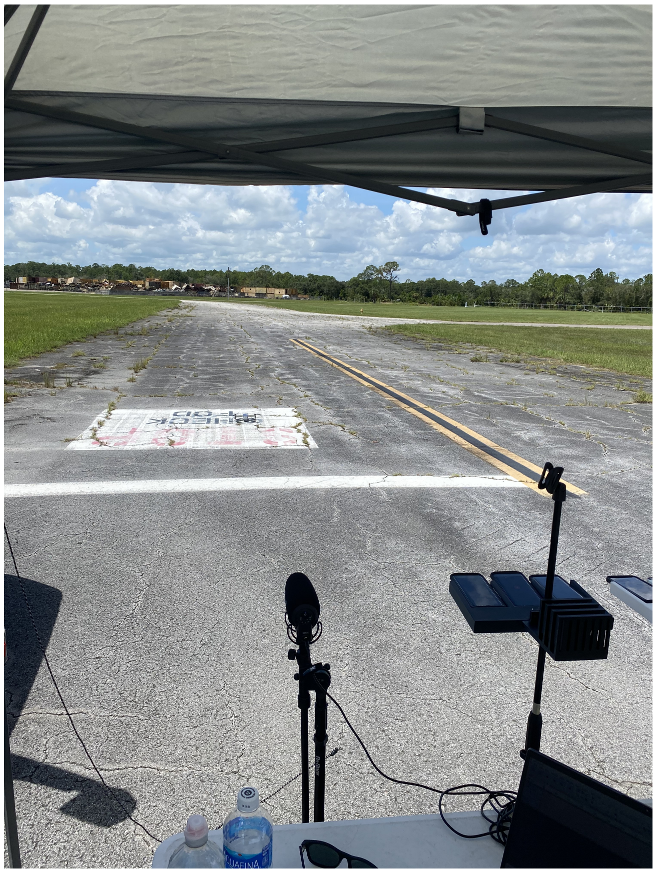


Figure 37: Microphones and Flight-line

5. The pilot conducts fights at 10 m , 20 m, 30 m, 40 m, 50 m, 75 m, and 100 m for all 6 payloads. Once all audio data is gathered, the researcher follows the final HurtzHunter steps for audio processing Section 3.7 describes.

4.7.1 Experiment Limitations

The experiment limitations are time on the flight-line, and battery power. All flights could not be conducted one after another because the batteries for the Mavic Air 2 would die after ∼12 minutes of flight time. The batteries require 2 hours to charge, and only one charger was available. Therefore, flights took place at all times of the day over the course of three days. Only three days were allocated to run the tests in Avon Park, FL. So, once all experiments were done. There was no time for replication.

4.8 Case Studies

Four case studies use the data from this experiment to test the HurtzHunter design. Each case study targets specific aspects of the research hypothesis from Section 1.4 which is: This research hypothesizes that cell phones are able to determine the weight of a payload carried by a UAV from acoustic emissions at up to 100 m away ground distance from the UAV.

4.8.1 Case Study 1

Case study 1 focuses on repeating experiments in past research with the addition of a cell phone recording device. In past research, acoustic emissions were collected using a RVMP at 7 m ground distance from the drone. Case Study 1 uses the acoustic emissions collected by the RVMP and GS8 at only 7 m away. The HurtzHunter design trains a linear, quadratic, and cubic SVM with this data, and the design tests the AI prediction accuracy. This study uses the 5-cross validation process to train and test the AI. The results of this study provide experiment validation as well as prediction accuracy comparison between the RVMP and GS8.

4.8.2 Case Study 2

Case study 2 expands case study 1 by using the acoustic data from all ranges, 7-100 m. This study expands the research field by using the HurtzHunter design to create a linear, quadratic, and cubic SVM capable of detecting payload at up to 100 m away. This study uses the GS8 and RVMP as the recording devices and uses the HurtzHunter Python scripts to train and test the AIs through 5-cross validation. This study provides insight into the maximum range for acoustic payload detection, and the difference in prediction accuracy for the GS8 and RVMP.

4.8.3 Case Study 3

Case study 3 expands case study 2 by using all the cell phones for acoustic payload detection. This study uses the HurtzHunter design to build a cubic SVM for the GP4, iP11P, and GS20 using data from all ranges (7-100 m). This study creates results using 5-cross validation and compares the prediction quality of the RVMP, GS8, GP4, iP11P, and GS20. The results provide evidence to show the effect of cell phone model on detection accuracy.

4.8.4 Case Study 4

This cause study uses the 3 GS8s to assess the effect of manufacturing on prediction accuracy. Acoustic data from two GS8s trains a cubic SVM, and data from the third GS8 tests the cubic SVM. The results indicate if data from every recording device is needed to properly train an AI, or if as long as the same model is used, the accuracy remains the same.

4.9 Runtime

The experiments track the time it takes to build the AIs in each case study. The experiment uses the Python library time [51] to track the runtimes. This model uses an epoch for time reference to determine all runtimes. This research is done on a Windows computer and the epoch for this library on a Windows computer is midnight January 1, 1601 [52][53]. The following algorithms displays the commands for capturing runtimes using the time library:

```
1: start_time1 = time.time()
2: ”Do some code”
3: endtime1 = time.time() - start_time1
4: print(’time 1:’, endtime1)
```

Time Library Capturing Commands

Line 1 captures the current time relative to the epoch with the command time.time() and stores it in the variable start_time1. Then, line 2 represents some code execution which needs to be timed. Then, line 3 shows the command to store the runtime by subtracting the current time relative to the epoch from the starting time. The stop time is recorded in endtime1. Then, line 4 prints the runtime to the console. The precision of the epoch based timer on a Microsoft Windows machine is 100 ns [53].

4.10 Wind Noise

The experiments use the iOS National Institute of Occupational Safety and Health (NIOSH) Sound Meter app to measure ambient noise [54]. The Center for Disease Control (CDC) developed this app to measure ambient noise in dB [55]. Right before each flight, the researcher uses this app on the iP11P to measure the ambient noise in dB. Chapter 5 discusses the impact of wind noise on the system. In each run, the largest contributor to ambient noise is wind. When wind is not present, the NIOSH meter reads very low ambient noise.

4.11 Statistical Analysis

This research uses the prediction accuracy of the AI for statistical analysis. Equation 8 in Section 4.4 shows the prediction accuracy calculation. The accuracy shows how well a recording configuration performs, and this research uses the accuracy to determine the best drone detection configuration. Due to the limitations of this research, further statistical analysis is not achievable. There was only enough time on the flight line to run one repetition of the experiment this chapter describes. With only one repetition, there is zero degrees of freedom for statistical analysis. Statistical analysis, such as a difference in means test, needs at least one degree of freedom for proper analysis. Thus, conducting such a test with zero degrees of freedom would be misleading. Chapter 5 discusses the statistical analysis techniques which would be used, should there be more degrees of freedom. Specifically, a difference in means t-test would be used to show statistical differences in prediction performance between different recording devices and ranges. However, given the limitations of this research, future work needs to accomplish such tests.

4.12 Summary

This chapter describes the SUT and experiment design. Section 4.2 describes the SUT. Section 4.3 discusses the factors. Section 4.4 conveys the metrics. Section 4.5 discusses the constant parameters. Section 4.6 describes the uncontrollable variables. Section 4.7 conveys the experiment design. Section 4.8 explains the case studies. Section 4.9 discusses the timer used for tracking runtime in the experiment. Section 4.10 discusses the sound meter app used for measuring wind noise. Section 4.11 discusses the statistical analysis.

V. Results and Analysis

5.1 Overview

This chapter discusses the results from the experiments Chapter 4 discuses. Section 5.2 reviews the findings from case study 1. Section 5.3 reviews the results from case study 2. Section 5.4 reviews the results from case study 3. Finally, Section 5.5 reviews the results from case study 4. Each case study uses the same data from the experiments Chapter 4 describes. The studies differ in the way they use the data to train and test the Support Vector Machines (SVMs). Previous research, [7] successfully used acoustic emissions to classify UAV payload weight. These researchers achieved 98.5 - 98.9 % accuracy at 7 meters from the acoustic source using SVMs. The recording device used in this research is the RodeVideoMicPro (RVMP). Case study 1 repeats this experiment in [7] as well as expanding the field of study by using the Galaxy S8 (GS8) for an additional recording device. Case study 2 builds upon case study 1 by expanding the recording range from 7 m to 100 m. Case study 3 further expands the research by evaluating prediction accuracy among 4 different cell phone devices and the RVMP at 7 m to 100 m range. Finally, case study 4 evaluates the effect of manufacturing on the prediction accuracy by using three GS8s to train and test a cubic SVM (CSVM). These case studies answer the questions in Section 4.1 which are as follows.

- Can cell phones detect UAV payload weight with sound?
- What is the maximum range of acoustic UAV payload detection?
- Does the cell phone model affect detection accuracy?
- Does a high-quality microphone perform better than a cell phone for acoustic payload detection?

- Does the manufacturing process for a specific recording device affect detection accuracy?

Each study targets specific questions, and the result aggregate provides evidence to support the Hypothesis of this research.

5.2 Case Study 1

The purpose of this case study is to provide experiment validation by repeating experiments in past research [7], and expand the research field by using a cell phone as the recording device. The experiments in [7], used the RVMP as the recording device and collected acoustic emissions from a hovering drone at 7 m ground distance from the drone. This study repeats the experiment with the RVMP and GS8.

This case study compares the results from the RVMP, GS8, and past research. This scenario assumes the oldest model cell phone performs the worst of the ones listed in Chapter 4. The oldest model is the GS8. By comparing the cell phone expected to perform the worst (the GS8) to the RVMP, this case study seeks to set a base performance comparison between a cell phone device and the RVMP.

This case study provides evidence to answer the following questions: "Does a high-quality microphone perform better than a cell phone for acoustic payload detection?" and, "Can cell phones detect UAV payload weight with sound?".

This case study uses three SVMs, one with a linear, a quadratic, and a cubic kernel. The reason this study uses these SVM kernels is because the experiments in [7] used these SVM kernels. By using the same SVM configurations, this case study is able to compare its results with past research. In [7], the researchers achieved prediction accuracy ranging from 98.5 - 98.9 % with their configurations.

5.2.1 RodeVideoMicPro

Table 7 shows the prediction results from a linear SVM (LSVM) trained and tested with the data collected by the RVMP at 7 m away.

Table 7: Case Study 1 RodeVideoMicPro linear SVM Results

RVMP LSVM Case Study 1 (7 m)	
Weight (g)	Prediction Accuracy by weight
0	100.00%
56	99.80%
112	100.00%
168	100.00%
224	98.66%
280	99.80%
Total Accuracy	**99.71%**

In Table 7, the total prediction accuracy is 99.71 %. The total accuracy is the average prediction accuracy for each weight shown in Table 7. This result is consistent with past research which achieved a 98.9 % prediction accuracy with the RVMP at 7 m from the drone [7]. These results indicate the weight carried by a drone affects the acoustics emitted by the drone. And, the effect each weight has on the acoustics is unique enough for accurate payload classification. Specifically, MFCCs can capture the acoustic variation between payload weights, and an LSVM is able to use the MFCCs to accurately predict payload weight. Lastly, the acoustic signal collected at 7 m from the drone is strong enough to classify payload weight.

Table 8 shows the prediction results from the quadratic SVM (QSVM) using the RVMP at 7 m ground distance.

Table 8: Case Study 1 RVMP Quadratic SVM Results

RVMP QSVM Case Study 1 (7 m)	
Weight (g)	Prediction Accuracy by weight
0	100.00%
56	99.43%
112	100.00%
168	99.97%
224	95.76%
280	98.70%
Total Accuracy	**98.98%**

The results in Table 8 reflect the same trends present in the results in Table 7. So, the results from the QSVM behave the same as the results from the LSVM. The total accuracy using the QSVM is 98.98 %. This result is consistent with past research which observed a prediction accuracy of 98.9 % using a QSVM and the RVMP [7]. The results further show that acoustic emissions can be used to identify payload weight carried by a drone.

Table 9 shows the prediction results form the cubic SVM (CSVM) using the RVMP at 7 m ground distance.

Table 9: Case Study 1 RVMP Cubic SVM Results

RVMP CSVM Case Study 1 (7 m)	
Weight (g)	Prediction Accuracy by weight
0	100.00%
56	99.53%
112	100.00%
168	99.97%
224	96.09%
280	98.76%
Total Accuracy	**99.06%**

The results in Table 9 reflect the same trend observed by the results in Table 7 and 8. The total accuracy using the CSVM is 99.06 %. This result is consistent with past research which observed a 98.5 % prediction accuracy from the CSVM [7]. Overall, the results show the CSVM is capable of payload classification using acoustic emissions.

The results in this case study using the RVMP are consistent with past research. This provides experiment validation, meaning there is evidence to support this experiment provides accurate results consistent with its intent.

5.2.2 Samsung Galaxy S8

This case study uses the GS8 to expand the field of research. The results from the GS8 show the capability of using cell phones to detect UAV payload weight from acoustic emissions. The cell phone uses microphones found on many IoT devices, so this study provides evidence for the ability to use IoT devices in general for payload detection. By using the GS8, this case study compares the results of the RVMP and the GS8. The comparison shows performance differences between a high-quality microphone and a small cell phone microphone. The GS8 records audio at 7 m from the UAV. A linear, quadratic, and cubic SVM use the audio data for payload classification. Table 10 show the results from the LSVM.

Table 10: Case Study 1 Galaxy S8 Linear SVM Results

GS8 LSVM Case Study 1 (7 m)	
Weight (g)	Prediction Accuracy by weight
0	99.97%
56	99.70%
112	100.00%
168	99.97%
224	98.06%
280	100.00%
Total Accuracy	**99.62%**

In Table 10, the total prediction accuracy is 99.62 %. The total accuracy is the average prediction accuracy of all the weights in Table 10. This result shows the prediction scheme using the GS8 is just as accurate as the scheme using the RVMP. In other words, the performance of a cell phone device is nearly the same as a high-quality microphone.

Table 11 shows the prediction results from the GS8 using a QSVM.

Table 11: Case Study 1 Galaxy S8 Quadratic SVM Results

GS8 QSVM Case Study 1 (7 m)	
Weight (g)	Prediction Accuracy by weight
0	99.93%
56	99.77%
112	100.00%
168	99.93%
224	98.76%
280	99.93%
Total Accuracy	**99.72%**

The total accuracy in Table 11 from using a QSVM is 99.72 %. These observations are consistent with the observations from the LSVM. The CSVM has similar results. Table 12 shows summarizes the CSVM results.

Table 12: Case Study 1 Galaxy S8 Cubic SVM Results

GS8 CSVM Case Study 1 (7 m)	
Weight (g)	Prediction Accuracy by weight
0	99.66%
56	100.00%
112	100.00%
168	100.00%
224	100.00%
280	99.93%
Total Accuracy	**99.93%**

The results show the GS8 performed just as well if not better as the RVMP in this configuration. Thus at 7 m from the drone, the GS8 is able to classify payload weight with the same accuracy as the RVMP.

5.2.3 Graphical Comparison

Figure 38 summarizes the prediction accuracy in this case study.

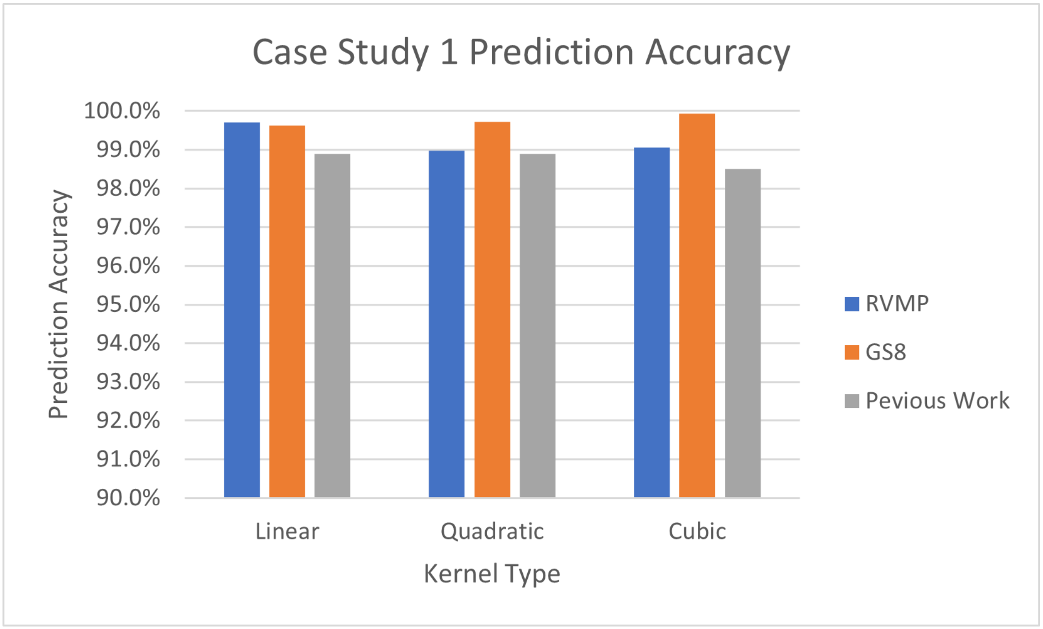


Figure 38: Case Study 1 Prediction Accuracy Comparison

Both the RVMP and GS8 consistently perform with nearly the same accuracy as past research. These results indicate the experiment behaves consistently with past research providing evidence for experiment validation. In addition, the results show that the GS8 performs at the same level as the RVMP. So, there is evidence to suggest a cell phone device achieves the same performance as a high-quality microphone for acoustic payload detection at 7 m. Lastly, the three SVM configurations (linear, quadratic, and cubic kernel) performs nearly the same for the GS8, RVMP, and prior research [7].

Table 13 shows the total runtimes for each AI

Table 13: Case Study 1 Runtimes

		1 AI (s)	**5 AI (s)**
RVMP	LSVM	0.14	0.98
	QSVM	0.52	2.92
	CSVM	0.4	2.43
GS8	LSVM	0.13	0.72
	QSVM	0.38	2.54
	CSVM	0.31	1.69

Table 13 shows the runtime of 1 AI and 5 AIs for each configuration. The reason this study shows 5 AIs is because 5-cross validation requires 5 AIs for implementation. As Chapter 3 discusses, HurtzHunter uses 5-cross validation for prediction results. So, the runtime to build all 5 AIs conveys the real runtime to build the AI for the HurtzHunter design. The runtime for the LSVM is shorter than the QSVM and CSVM for both the GS8 and RVMP. However, the longest runtime is 2.4 s which is very short, and does not affect the goals of this research.

5.2.4 Statistical Analysis

This research runs one repetition of the experiment. There was not enough time on the flight line to run additional repetitions while on TDY in Florida. Therefore, the data has zero degrees of freedom which limits the statistical analysis of this research. If more repetitions could be accomplished, then a difference in means test would be done on the prediction accuracy to determine statistical significance between configurations. Specifically, a two-sample t-test would be used to compare the accuracy between the SVMs and recording devices. For example, the difference between the

RVMP and GS8 is small. The two-sample t-test would show whether this difference in accuracy is significant, and if the recording device has a statistically noticeable impact on the system. Also, with more degrees of freedom, the two-sample t-test would be used to compare statistical differences in accuracy between configurations in case studies 2-4. However, conducting a test with zero degrees of freedom in any case study would be misleading, so future work needs to provide statistical evidence.

5.3 Case Study 2

Case study 2 builds upon case study 1 by increasing the distance between the microphone and drone from 7 m to 100 m. Previous work gathered acoustic emissions at 7 m away from the drone. This research expands the field of study by gathering acoustic emissions at 7 m to 100 m away. This study uses the GS8 and RVMP to collect acoustic data at 7 m, 10 m, 20 m, 30 m, 40 m, 50 m, 75 m, and 100 m ground distance from the UAV. Additionally this case study builds an LSVM, QSVM, and CSVM for each recording capable of acoustic payload prediction using MFCCs. The results in this case study provide answers to the questions "What is the maximum range of acoustic UAV payload detection?" and, "Does a high-quality microphone perform better than a cell phone for acoustic payload detection?"

5.3.1 RodeVideoMicPro

Table 14 summarizes the LSVM results from the RVMP. The data recorded by the RVMP trains and tests the LSVM AI.

Table 14: Case Study 2 RVMP LSVM

	RVMP LSVM Prediction Results 7 - 100 m								
	Distance (meters)								
Weight (g)	7	10	20	30	40	50	75	100	**Avg by weight**
0	99.60%	99.73%	98.16%	95.46%	99.50%	99.90%	99.97%	98.63%	98.87%
56	92.92%	98.83%	92.99%	94.82%	98.43%	91.52%	45.58%	38.36%	81.68%
112	99.27%	96.89%	94.59%	87.71%	96.93%	94.29%	93.12%	12.65%	84.43%
168	98.46%	96.99%	98.60%	96.06%	97.60%	90.45%	97.06%	95.99%	96.40%
224	94.62%	92.12%	95.46%	96.83%	85.64%	79.47%	92.19%	61.44%	87.22%
280	97.90%	99.33%	99.57%	99.47%	99.27%	99.93%	98.20%	98.53%	99.03%
Distance Avg	97.13%	97.32%	96.56%	95.06%	96.23%	92.59%	87.69%	67.60%	
									Total Device Avg
									91.27%

The total accuracy in Table 14 is 91.27%. The total accuracy stands for the average accuracy of predicting all payload weights at all distances. The results show a loss in accuracy as distance increases. Case study 1 shows a total accuracy of 99.71% for the LSVM. When the distance expands to include 100 m, the average accuracy decreases to 91.27%. This decrease is expected due to the loss of signal strength as distance increases. A sound is fainter at longer distances and stronger at shorter distances, and fainter sounds are more difficult to capture with a microphone. Therefore, a weaker signal should lead to a decrease in prediction accuracy because it carries less information. The results in Table 14 support this phenomenon as the percent accuracy for each distance, in the row "Distance Avg" shows, decreases as distance increases. Specifically, the main decrease is from 40 m to 100 m.

Another cause for the decrease in accuracy is the increase in training data. Because the intensity of sound decreases with distance, 0 g at 7 m away sounds different than 0 g at 100 m. The frequencies are the same, but the magnitude of these frequencies changes. The AI may be classifying based on the change in loudness rather a change in frequency. Therefore, the AI may think two signals of the same loudness are the same weight when in reality they are not. In a perfect scenario, the AI would be able to discard changes in loudness and only classify changes in frequency, but the methodology in this research does not provide for a way to control the difference between loudness and frequency. Even so at closer range (below 40 m), the results indicate prediction accuracy at $\sim$96%.

In addition to loudness, nuisance factors such as wind and time of flight affect the AI's classification accuracy. During testing, the wind varied throughout the day. Wind intensity increased as the day went on. Wind noise greatly impacts prediction accuracy. For example, the audio recordings collected for 112 g at 100 m experienced the a high level of winds, and Table 14 shows the prediction accuracy greatly decreased to 12.65% for this data point. These nuisance factors do not play a major role when data is collected at 7 m, but when the range increases their impact becomes great. Section 5.6 discusses these nuisance factors in greater details.

Another change in case study 2 is the AI build time. The build time in case study 1 was negligible. But in case study 2 the runtime increases significantly. Table 15 sows the runtime for the LSVM in case study 2.

Table 15: Case Study 2 RVMP LSVM Runtime

	AI Train Time (s)	**min**	**hr**
1 AI	3781.8	63.0	1.1
all 5	19971.6	332.9	5.5

Table 15 shows two runtimes. The first in row ‘1 AI’ is the time it takes to build one AI. The second in row ‘all 5’ is the time it takes to build all 5 AIs needed to accomplish 5-cross validation. The total runtime to accomplish 5-cross validation for the LSVM is 5.5 hours. This runtime drastically increases from case study 1 to case study 2. The reason for this change is because the AI in this study uses data from all ranges (7 m to 100 m) which is seven times the data case study 1 uses. In addition to the amount of data, the nature of the data in this case study is more difficult to classify than the data in case study 1. Due to factors such as range, loudness, and wind noise, the SVM needs more hyperplanes to fully classify the data. In short, this case study uses more data which takes the SVM more time to classify.

Table 16 summarizes the QSVM results from the RVMP. The QSVM uses the data from the RVMP at all ranges.

Table 16: Case Study 2 RVMP QSVM Results

	RVMP QSVM Prediction Results 7 - 100 m								
	Distance (meters)								
Weight (g)	7	10	20	30	40	50	75	100	**Avg by weight**
0	99.77%	99.73%	98.03%	96.63%	99.67%	99.73%	99.93%	98.10%	98.95%
56	91.69%	99.27%	86.61%	92.75%	98.26%	95.39%	41.10%	9.12%	76.77%
112	100.00%	99.03%	93.79%	86.84%	96.79%	96.09%	93.99%	42.04%	88.57%
168	98.23%	97.03%	97.90%	95.93%	97.53%	89.45%	96.16%	93.66%	95.74%
224	89.58%	85.04%	95.59%	84.94%	82.74%	77.40%	88.58%	50.18%	81.76%
280	96.46%	97.86%	99.83%	99.73%	99.67%	100.00%	96.53%	97.93%	98.50%
Distance Avg	95.96%	96.33%	95.29%	92.80%	95.78%	93.01%	86.05%	65.17%	
									Total Device Avg
									90.05%

The total prediction accuracy of the QSVM is 90.05 %. This accuracy is slightly less than the LSVM but nearly the same. The same trends observed by the LSVM are present in the QSVM. As distance increases, the accuracy decreases. Also, the varying loudness and nuisance factors, such as wind, brings the accuracy down. The results show the largest decrease in accuracy from 40 m to 100 m. Contrary to the LSVM, the runtime for AI training is significantly less for the QSVM. Table 17 show the runtime for the QSVM.

Table 17: Case Study 2 RVMP QSVM Runtime

	AI Train Time (s)	**min**
1 AI	224.6	3.7
all 5	1086.5	18.1

The runtime for one AI is 3.7 min, and the runtime for all 5 AIs needed for 5-cross validation is 18.1 min. This runtime is far less than the runtime for the LSVM. The reason for this decrease is because the quadratic nature of the hyper plane in a QSVM allows the SVM to curve the boundary between classes. With curved boundaries, the SVM can fully classify the data with less hyperplanes, and the less hyperplanes there are, the less time the AI needs to train.

Although curved hyperplanes improve runtime, they increase the risk of over fitting the data. Overfitting occurs when the SVM classifies the data based on nuisance factors present in the data. For example, if one recording is very windy, the SVM may classify that recording by the wind noise rather than the actual MFCC values corresponding to the weight class. The reason overfitting is more common in higher degree hyperplanes and not linear hyperplanes is because nuisance factors generally follow non-normal trends in classification, and a linear hyper plane can only follow normal trends. In other words, a curved hyper plane can classify non-normal data and a linear hyper plane cannot.

However, nuisance factors are not the only non-normal characteristics. Often, the features needed to train an SVM are not normal, and an AI needs more complex hyperplanes for class differentiation. Because of this need for complex planes, the best way to prevent overfitted data is to increase the amount of repetitions, and use test data from different environments. Ideally, the experiments in this research would be repeated on multiple days, and in multiple environments. This repetition would provide the diverse data needed to detect overfitting and prevent an AI from training on nuisance factors. However, the nature of this research does not allow enough time to perform experiment repetitions in diverse environments. As Chapter 4 discusses, this research was accomplished over three days at the Air Force Bombing range in Avon Park, FL. Only one experiment repetition was accomplished in this time frame. There is no way to know whether overfitting occurred in the QSVM Table 16 summarizes. This research assumes it does not occur.

Table 18 shows the CSVM results from the RVMP.

Table 18: Case Study 2 RVMP CSVM Results

	RVMP CSVM Prediction Results 7 - 100 m								
	Distance (meters)								
Weight (g)	7	10	20	30	40	50	75	100	**Avg by weight**
0	99.97%	99.73%	97.96%	97.03%	99.67%	99.83%	99.87%	97.16%	98.90%
56	91.35%	99.37%	86.44%	93.59%	98.40%	95.26%	40.83%	10.35%	76.95%
112	100.00%	99.10%	92.59%	87.28%	96.89%	95.76%	92.89%	43.71%	88.53%
168	98.26%	98.00%	97.73%	96.06%	98.13%	89.85%	95.89%	92.62%	95.82%
224	89.35%	85.54%	95.83%	83.54%	84.47%	77.43%	86.84%	51.62%	81.83%
280	96.86%	97.86%	99.80%	99.57%	99.33%	100.00%	95.49%	97.30%	98.28%
Distance Avg	95.97%	96.60%	95.06%	92.85%	96.15%	93.02%	85.30%	65.46%	

Total Device Avg
90.05%

The total average for the CSVM is 90.05%. This average is the same as the QSVM. The CSVM performs nearly the same as the QSVM at each distance, and the average accuracy is the same as the QSVM. The same trends observed for the QSVM are present for the CSVM. Factors such as distance and wind bring the accuracy down, and the most noticeable decrease is between 40 m and 100 m. Furthermore, a CSVM runs the risk of overfitting, but due to the limits on this research there is no way to identify overfitting in the results. Also, the CSVM renders a much less runtime than the LSVM. Table 19 shows the runtime for the CSVM.

Table 19: Case Study 2 RVMP CSVM Runtime

	AI Train Time (s)	**min**
1 AI	205.6	3.4
all 5	1061.6	17.7

The runtime for one AI is 3.4 min, and the runtime for all 5 AIs needed for 5-cross validation is 17.7 min. These runtimes are less than the LSVM and QSVM. It is significantly less than the LSVM but marginally less than the QSVM. The significant decrease from the LSVM is due to the greater freedom a cubic hyper plane has to divide classes than a linear hyper plane. A CSVM also has greater flexibility to divide classes than a QSVM, and this flexibility could be the cause of the decrease in runtime.

However other factors, such as additional processes running on the computer could be the cause of the slight difference difference between the CSVM and QSVM runtime. This research does not run each AI build in a contained environment where it is the only process running. Therefore, there is no way to know if small changes in runtime could be due to hyper plane complexity or the process load on the CPU during runtime.

Figure 39 shows the accuracy of all three SVMs over distance.

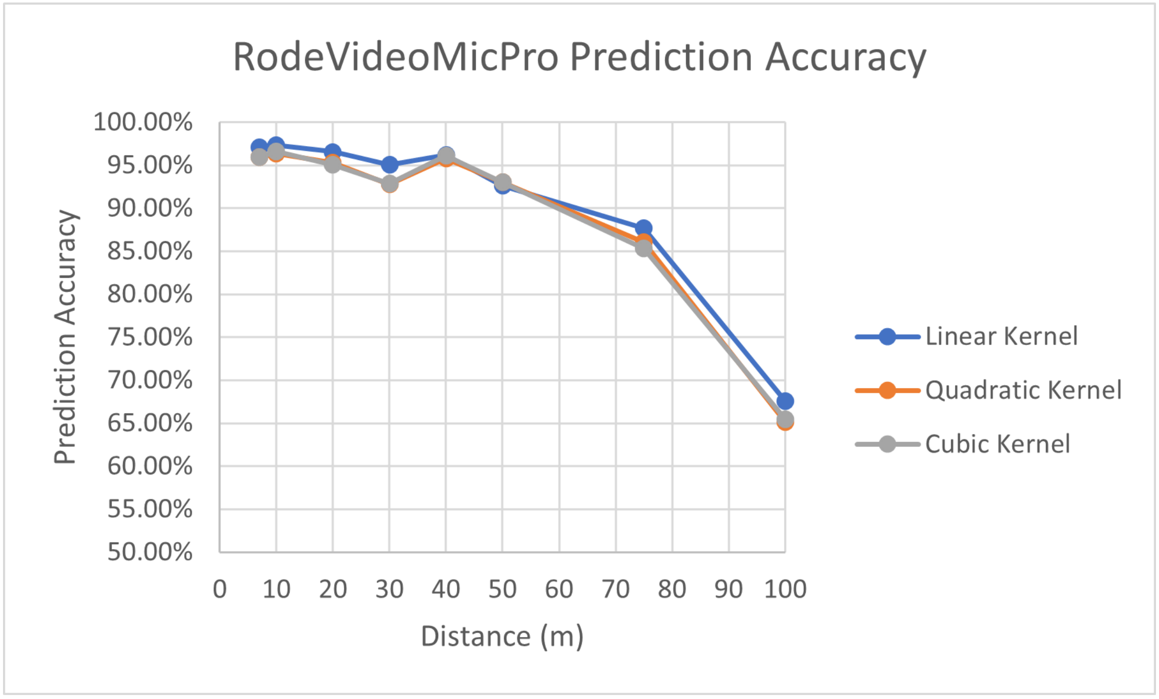


Figure 39: Case Study 2 RVMP Three SVMs

In each SVM, the same trends are present. From 7 m to 40 m the accuracy slightly fluctuates, then at 40 m the accuracy declines steadily. The slight difference from 7 m to 40 m is likely due to nuisance factors such as wind or other random sounds present from flight to flight. More testing should be done to verify the cause of this slight fluctuation. However, the results indicate distance does not significantly impact the accuracy of the system until after 40 m.

5.3.2 Samsung Galaxy S8

This subsection of case study 2 shows the results for the LSVM, QSVM, and CSVM using the GS8 data. Table 20 summarizes the results from the LSVM.

Table 20: Case Study 2 GS8 LSVM Results

	GS8 LSVM Prediction Results 7 - 100 m								
	Distance (meters)								
Weight (g)	7	10	20	30	40	50	75	100	**Avg by weight**
0	99.17%	98.73%	97.83%	98.53%	99.90%	99.63%	99.37%	96.13%	98.66%
56	94.19%	98.93%	87.55%	97.96%	98.70%	91.42%	52.65%	13.09%	79.31%
112	98.50%	92.69%	77.90%	90.38%	92.99%	82.27%	75.46%	50.82%	82.63%
168	94.82%	94.79%	95.19%	87.61%	97.40%	90.92%	94.56%	95.06%	93.79%
224	96.39%	88.81%	90.95%	93.52%	87.01%	71.02%	35.39%	78.53%	80.20%
280	99.67%	99.63%	99.30%	97.50%	98.90%	99.33%	99.37%	93.99%	98.46%
Distance Avg	97.12%	95.60%	91.45%	94.25%	95.82%	89.10%	76.13%	71.27%	
									Total Device Avg
									88.84%

Table 20 shows the total prediction accuracy to be 88.84%. This accuracy is significantly less than the prediction accuracy case study 1 shows for the GS8. The decrease in accuracy is due to the same trends this study observes for the RVMP. The increased range lead to a decrease in accuracy bringing the average prediction accuracy down. Additionally, the difference in signal loudness and nuisance factors present such as wind bring the average down. Similar to the RVMP LSVM, this LSVM requires significantly more runtime to fully classify the data. Table 21 shows the runtime required by this LSVM.

Table 21: Case Study 2 GS8 CSVM Runtime

	AI Train Time (s)	**min**	**hr**
1 AI	7831.9	130.5	2.2
all 5	34541.9	575.7	9.6

The runtime required for this LSVM is 2.2 hours for one AI and 9.6 hours for all 5 AIs needed for 5-cross validation. This large runtime is similar to the runtime for the LSVM from the RVMP. Using data from 7 m to 100 m rather than just 7 m greatly increases the complexity of the data, so the LSVM needs more hyperplanes to fully classify the data. Therefore, the runtime for a LSVM in this case study is significantly greater than the runtime in case study 1. The great increase in runtime indicates a linear hyper plane is not be sufficient to classify the data, and a hyper plane with greater complexity such as a quadratic or cubic curve is needed. However, runtime may not be a limitation in certain scenarios, so depending on the application a linear hyper plane may be sufficient. This research assumes runtime is a priority, and the shorter the runtime the better

Table 22 summarizes the results for the QSVM.

Table 22: Case Study 2 GS8 QSVM Results

	GS8 QSVM Prediction Results 7 - 100 m								
	Distance (meters)								
Weight (g)	7	10	20	30	40	50	75	100	**Avg by weight**
0	99.87%	97.96%	97.23%	98.63%	99.73%	99.50%	99.33%	94.96%	98.40%
56	93.69%	99.73%	86.61%	97.96%	97.96%	89.81%	41.47%	5.64%	76.61%
112	99.43%	93.02%	83.24%	96.76%	93.56%	86.54%	71.75%	51.52%	84.48%
168	95.63%	96.16%	94.46%	83.11%	97.43%	83.11%	94.29%	92.42%	92.08%
224	97.40%	86.88%	91.42%	94.62%	86.68%	70.88%	33.72%	75.89%	79.69%
280	99.97%	99.93%	99.57%	98.16%	99.00%	99.50%	99.57%	90.78%	98.31%
Distance Avg	97.67%	95.61%	92.09%	94.87%	95.73%	88.22%	73.36%	68.54%	
									Total Device Avg
									88.26%

The total prediction accuracy for the QSVM is 88.26 %. The QSVM encompasses the same trends present in the LSVM. The QSVM accuracy is slightly less than the LSVM, but the runtime greatly improves. The decrease in accuracy was slight and may not be significant. Table 23 shows the time of the QSVM.

Table 23: Case Study 2 GS8 QSVM Runtime

	AI Train Time (s)	**min**
1 AI	257.6	4.3
all 5	1357.5	22.6

The total runtime for one AI is 4.2 min and the runtime for all 5 AIs needed for 5-cross validation is 22.6 min. As the SVMs for the RVMP show, the runtime for the GS8 QSVM is significantly less than the LSVM due to the complexity of the quadratic hyper plane.

Table 24 summarizes the results for the CSVM.

Table 24: Case Study 2 GS8 CSVM Results

	GS8 CSVM Prediction Results 7 - 100 m								
	Distance (meters)								
Weight (g)	7	10	20	30	40	50	75	100	**Avg by weight**
0	99.90%	98.13%	97.22%	98.73%	99.67%	99.47%	99.33%	94.57%	98.38%
56	93.09%	99.73%	85.14%	97.76%	97.93%	88.25%	44.91%	9.92%	77.09%
112	99.47%	93.49%	83.01%	94.89%	92.75%	85.04%	69.88%	48.68%	83.40%
168	95.16%	95.49%	94.69%	83.81%	97.30%	83.84%	94.29%	91.95%	92.07%
224	98.06%	85.91%	91.85%	94.56%	87.25%	72.52%	38.36%	75.69%	80.53%
280	99.87%	100.00%	99.47%	98.16%	99.47%	99.53%	99.33%	90.95%	98.35%
Distance Avg	97.59%	95.46%	91.90%	94.65%	95.73%	88.11%	74.35%	68.63%	
									Total Device Avg
									88.30%

The total prediction accuracy is 88.30% for the CSVM. The CSVM encompasses the same trends present in the LSVM and QSVM. The CSVM accuracy is slightly less than the LSVM and slightly greater than the QSVM. This difference in accuracy was slight and may not be significant. Table 25 shows the runtime for the CSVM.

Table 25: Case Study 2 GS8 CSVM Runtime

	AI Train Time (s)	**min**
1 AI	264.5	4.4
all 5	1324.1	22.1

The runtime for one AI is 4.4 min and for all 5 AIs is 22.1 min. This runtime is significantly less than the LSVM and slightly less than the QSVM. The significant drop in runtime between the LSVM and CSVM is due to the complex shape of the cubic hyperplanes. The slight drop in runtime between the QSVM and CSVM may be due to other factors such as CPU load during runtime or the increased complexity of the cubic kernel.

Figure 40 shows the accuracy of all three SVMs over distance.

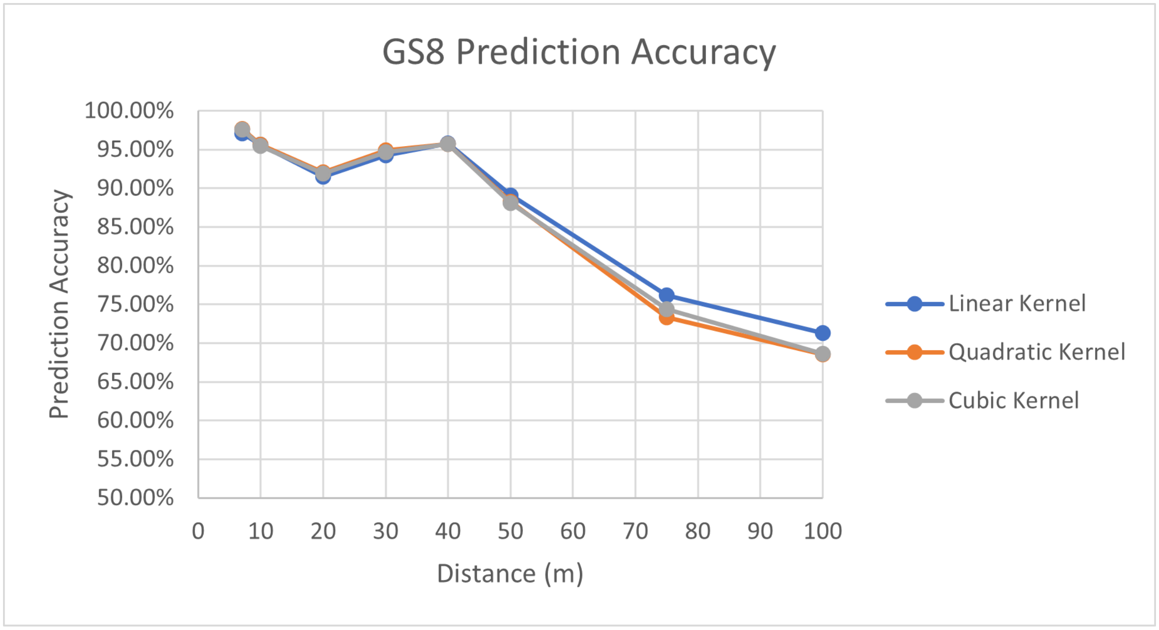


Figure 40: Case Study 2 GS8 Three SVM Comparison

The same trends the RVMP shows across all three SVMs are present here for the GS8. Accuracy slightly fluctuates between 7 m to 40 m. And there is a steady decline from 40 m to 100 m.

5.3.3 Graphical Comparison

Because all three kernel configurations for the SVM follow the same results, this research concludes the CSVM is the optimal configuration due to its high accuracy and short runtime. Chapter 6 further discusses this choice. Because of this choice, this section compares the CSVM for the RVMP and GS8. Also, case study 3 and 4 focus solely on the CSVM due to this optimization choice. Figure 41 shows the CSVM for the RVMP and GS8.

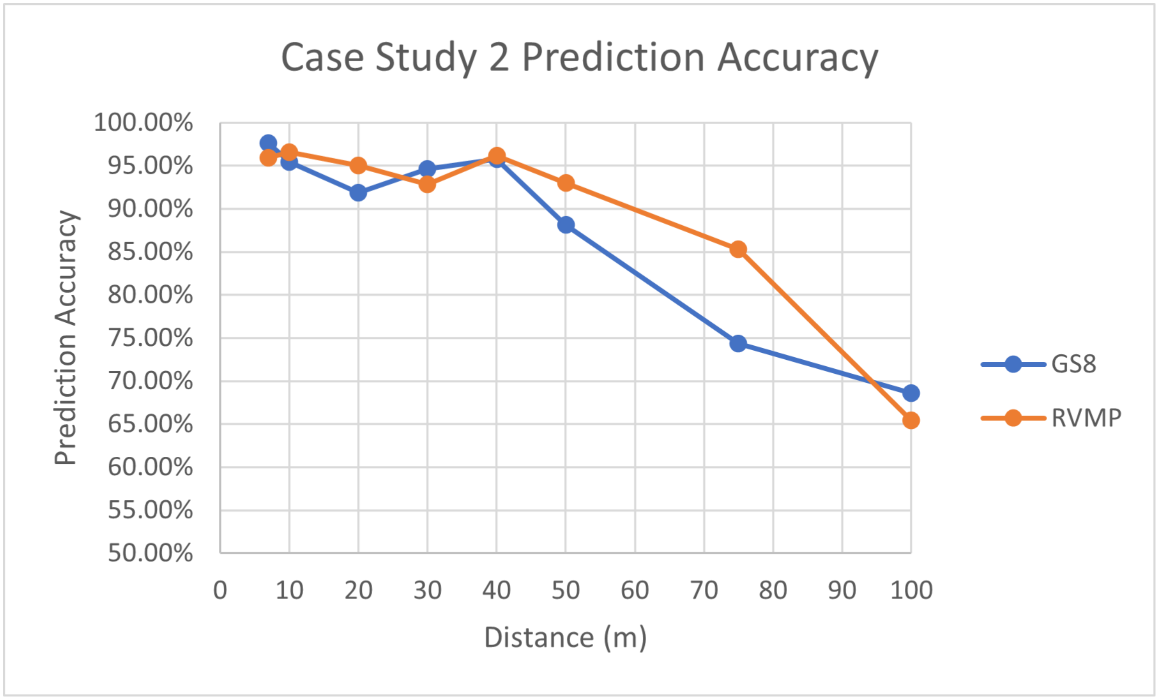


Figure 41: Case Study 2 CSVM Comparison

The graph shows that there is not an optimal device for all distances. The RVMP and GS8 jockey for position from 7 m to 40 m, then the RVMP is optimal from 40 m until 100 m where the GS8 is optimal. This result shows that the same trend is among both recording devices. Also, it shows that the cell phone performs the same as a high-quality microphone. Additionally, both devices observe a drop-off in prediction quality after 40 m.

Figure 42 and 43 show the prediction results for the GS8 and the RVMP with their quadratic trendlines using the CSVM.

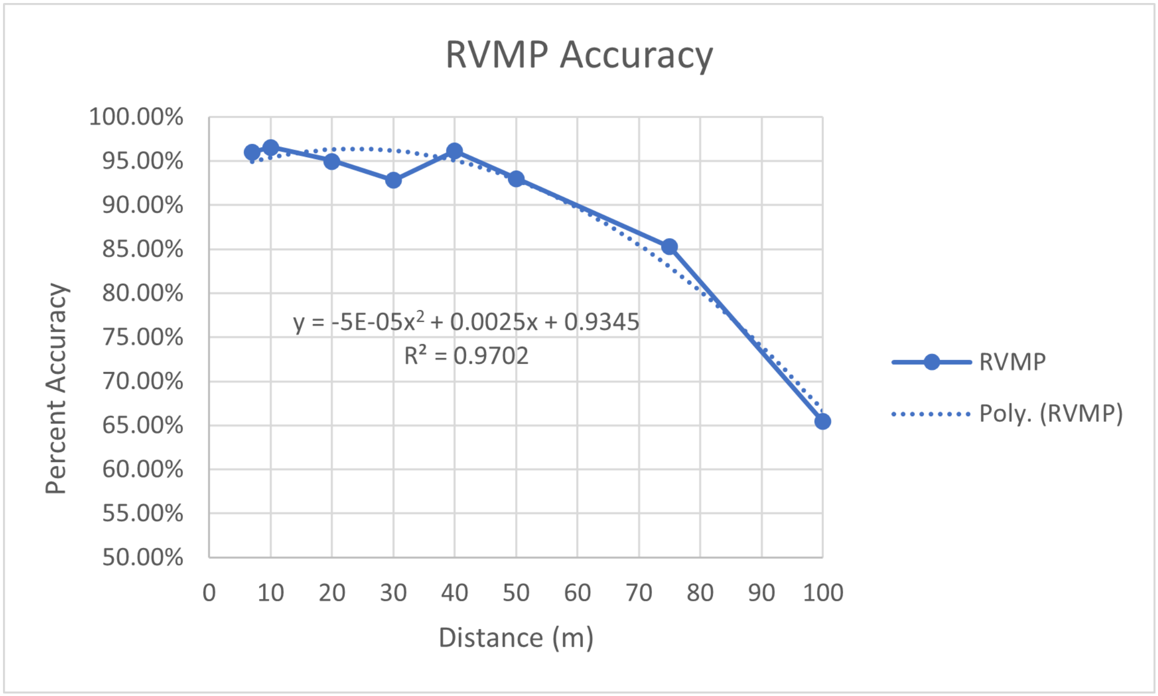


Figure 42: Case Study 2 RVMP CSVM With Trendline

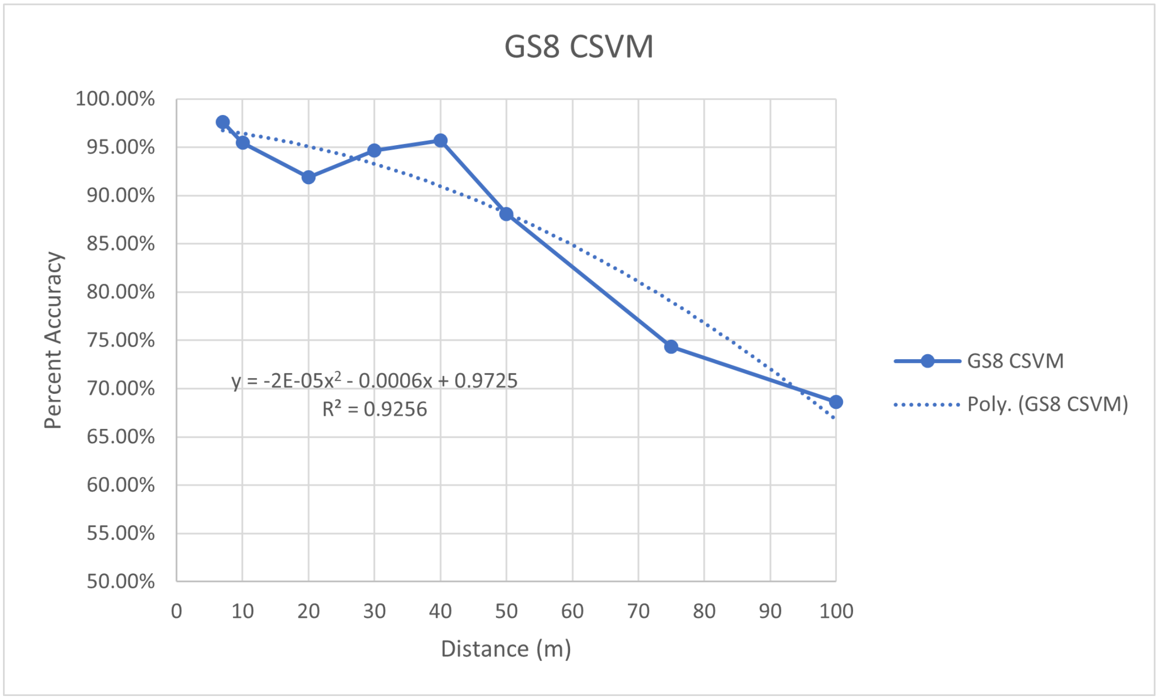


Figure 43: Case Study 2 GS8 CSVM With Trendline

Based on these figures, a negative second degree polynomial can model prediction accuracy. Specifically, the trendline for the RVMP is:

$$y = -5 * 10^{-5} * x^2 + .0025 * x + 0.9345 \tag{9}$$

with an $R^2 = 0.9702$, and the trendline for the GS8 is

$$y = -2 * 10^{-5} * x^2 - .0006 * x + 0.9725 \tag{10}$$

with an $R^2 = 0.9256$ 0.92. The R^2 value indicates how well the trendline fits the data. An $R^2 = 1$ means the trendline perfectly fits the data. Therefore, trendlines with a R^2 value closest to 1 are the best fit equation for the data. The fit of the trendline is tighter for the RVMP than the GS8 because the R^2 value for the RVMP is closer to 1 than the R^2 value for the GS8. But, both the RVMP and GS8 prediction results follow the curve of the second degree polynomial. This trendline provides possible prediction results for distances not recorded in this research. For example at 125 m, the trendlines indicate the RVMP accuracy would be 46.48% and the GS8 accuracy would be 58.5%. This prediction indicates that the GS8 accuracy declines at a slower rate than the RVMP after 100 m.

5.4 Case Study 3

This case study builds upon case study 2 to include more cell phones as recording devices. Due to the results in case study 2, this case study only uses the cubic SVM. This study uses the CSVM because case study 2 shows that the CSVM requires the least amount of runtime, yet achieves nearly the same accuracy as the LSVM and QSVM. The long runtime required for the LSVM makes it non-ideal for AI training and tuning. The runtime for the QSVM is similar to the CSVM, but the CSVM runtime is still less.

The additional recording devices this case study uses are the Google Pixel 4 (GP4), iPhone 11 Pro (iP11P), and Galaxy S 20 (GS20). This study uses the results for the RVMP and GS8 from case study 2 to compare performance across all devices. This study uses data from 7 m to 100 m from each device.

This study expands the research field by using additional cell phone devices, varying in make and model, for acoustic payload. The results provide evidence to answer the questions:

- ”Can cell phones detect UAV payload weight with sound?”
- ”What is the maximum range of acoustic UAV payload detection?”
- ”Does the cell phone model affect detection accuracy?”
- ”Does a high-quality microphone perform better than a cell phone for acoustic payload detection?”

5.4.1 Google Pixel 4

Table 26 shows the results for the GP4.

Table 26: Case Study 3 GP4 CSVM Results

	GP4 CSVM Prediction Results 7 - 100 m								
	Distance (meters)								
Weight (g)	7	10	20	30	40	50	75	100	**Avg by weight**
0	98.26%	96.86%	92.75%	97.30%	99.80%	99.50%	98.60%	92.85%	96.99%
56	63.64%	98.53%	85.18%	92.99%	92.89%	80.90%	42.90%	3.87%	70.11%
112	99.93%	86.71%	63.91%	88.25%	90.45%	93.96%	72.39%	48.01%	80.45%
168	91.72%	96.69%	93.16%	88.85%	94.62%	83.64%	89.55%	90.28%	91.06%
224	94.69%	83.04%	94.66%	88.41%	84.81%	56.43%	72.39%	59.27%	79.21%
280	98.80%	96.99%	99.20%	98.76%	94.56%	98.70%	94.02%	91.32%	96.54%
Distance Avg	91.17%	93.14%	88.14%	92.43%	92.86%	85.52%	78.31%	64.27%	
									Total Device Avg
									85.73%

The total accuracy of the device is 85.75%. The same trends case study 2 finds are present here. The accuracy remains relatively the same from 7 m to 40 m, then sees a steady decline from 40 m to 100 m. The total accuracy is 3-5% less than the RVMP and the GS8. However, some distances show nearly the same accuracy as the GS8 and RVMP. So, the GP4 overall is not the best solution, but at certain distances it may be no different than the RVMP and the GS8. The GP4 may have a lower accuracy because of the phone hardware, software, design structure, or position during the experiment. In the array of cell phones the RVMP and GS8 are on the right side of the array and the GP4 is on the left, so the setup may have affected accuracy. However, there is not clear observables in this research to indicate the exact reason for the lower accuracy from the GP4. Still, the results show the GP4 is less effective than the RVMP and GS8.

Furthermore, the results for the GP4 refute the assumption made about device age and accuracy. It was assumed that newer cell phones would perform better than older ones. But, the GP4, released in 2019, performed worse than the GP8, released in 2017 [39]. It is difficult to determine the cause of this observation without deep inspection of the GP4 design and the GP8. Possibly, the newer phones focus less on recording qualities than in the past. Or, Google phones may configure their device in a way less optimal for this use case than the GS8. Future work needs to determine the exact cause. Having said, the GP4 accuracy is greater than 90% at 7 m to 40 m, performing at around the same accuracy as the GS8 and RVMP. Therefore, the type of recording device may not matter at close range. Statistical analysis should be used to show in which cases there is a statistical difference between devices, but given this experiment has zero degrees of freedom, such analysis would be misleading in this research.

Table 27 shows the runtime for the GP4.

Table 27: Case Study 3 GP4 Runtime

	AI Train Time (s)	**min**
1 AI	216.8	3.6
all 5	1135.1	18.9

The build results in 3.6 min for 1 AI and 18.9 min for all 5 AIs needed for 5-cross validation. This runtime is slightly less than the GS8 and RVMP but in the ball park. This result makes sense because the cubic kernel used for the SVM should result in about the same runtime on all devices. As in case study 2, the runtime calculation for this study is not made in a contained environment. Other processes in the back ground could contribute in the slight variations of runtime among the CSVMs.

Figure 44 shows the graph and trendline of the GP4 data.

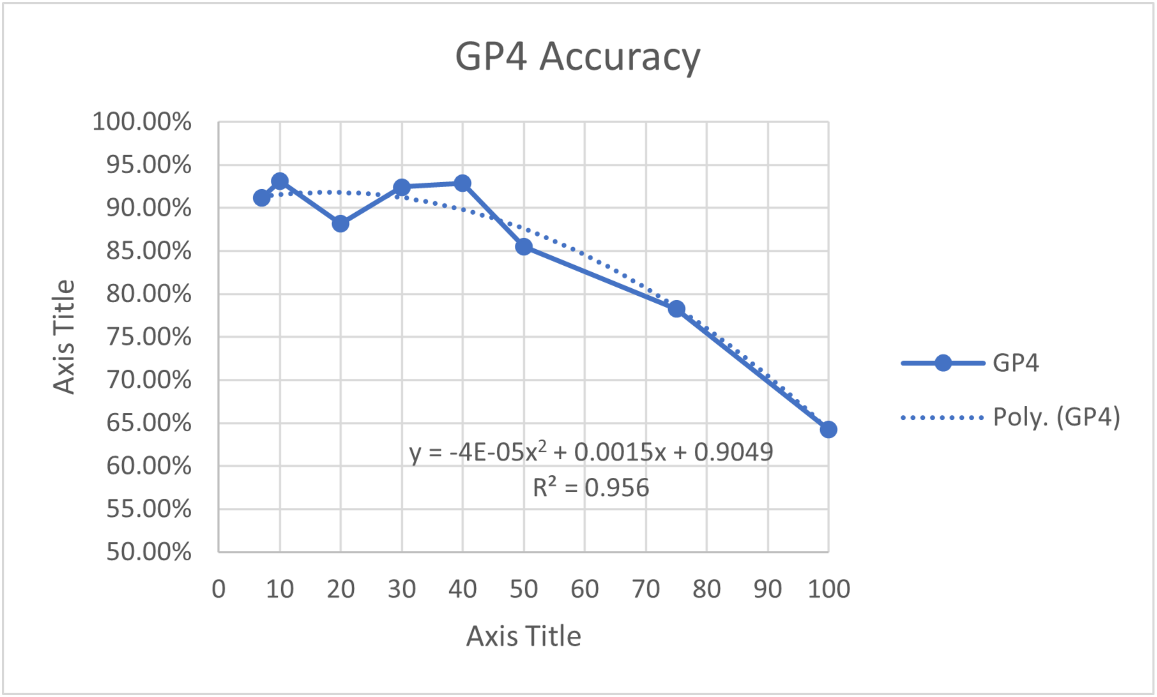


Figure 44: Case Study 3 GP4 CSVM With Trendline

The trendline is:

$$y = -4 * 10^{-5} * x^2 + .0015x + .9049 \tag{11}$$

with an $R^2 = 0.956$. In line with the results of case study 2, the prediction results follow a second degree polynomial curve as distance increases. Therefore, prediction accuracy for distances not collected can be predicted. For example, the prediction for 125 m is 46.74%.

5.4.2 iPhone 11 Pro

Table 28 summarizes the results for the (iP11P).

Table 28: Case Study 3 iP11P CSVM Results

	iP11P CSVM Prediction Results 7 - 100 m								
	Distance (meters)								
Weight (g)	7	10	20	30	40	50	75	100	**Avg by weight**
0	99.93%	98.33%	96.49%	97.93%	99.47%	99.43%	99.87%	97.10%	98.57%
56	77.36%	98.63%	86.98%	88.48%	91.62%	85.74%	48.68%	16.66%	74.27%
112	99.63%	89.62%	60.67%	95.76%	89.95%	82.04%	71.35%	37.60%	78.33%
168	97.93%	97.30%	97.76%	89.95%	99.40%	85.34%	94.72%	95.13%	94.69%
224	98.83%	81.97%	87.97%	96.09%	87.55%	65.31%	66.81%	58.23%	80.35%
280	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	99.97%	100.00%
Distance Avg	95.61%	94.31%	88.31%	94.70%	94.67%	86.31%	80.24%	67.45%	
									Total Device Avg
									87.70%

The total accuracy is 87.70%. The same trends and causes for the results in case study 2 are present here. Overall, the iP11P performs better than the GP4 but worse than the RVMP and GS8. Whether there is a statistical difference in accuracy needs to be determined. These results also refute the assumption that prediction quality increases with newer phones. The iP11P release date is 2019, yet it performs worse than the GS8 (2017) [39].

Table 29 shows the runtime for the iP11P.

Table 29: Case Study 3 iP11P Runtime

	AI Train Time (s)	**min**
1 AI	248.4	4.1
all 5	1213.4	20.2

The CSVM runtime for 1 AI is 4.1 min and 20.2 min for all 5 AI's required for 5-cross validation. The results are nearly the same as the runtime for the RVMP, GS8, and GP4. Indicating the CSVM renders the same result regardless of device.

Figure 45 shows the graph and trendline of the iP11P data.

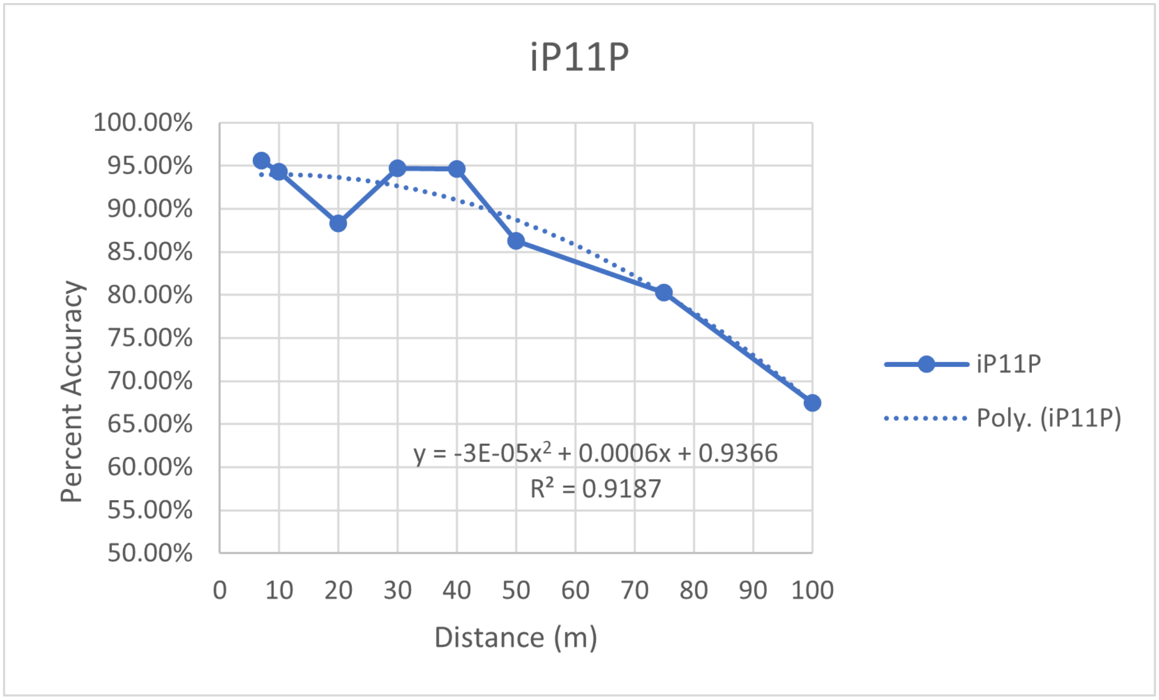


Figure 45: Case Study 3 iP11P CSVM With Trendline

The trendline is:

$$y = -3 * 10^{-5} * x^2 + .0006x + .9366 \tag{12}$$

with an $R^2 = 0.9187$. The prediction results follow a second degree polynomial curve as distance increases. Therefore, prediction accuracy for distances not collected can be predicted. For example, the prediction for 125 m is 54.29%.

5.4.3 Samsung Galaxy S 20

Table 30 summarizes the results for the (GS20).

Table 30: Case Study 3 GS20 CSVM Results

	GS20 CSVM Prediction Results 7 - 100 m								
	Distance (meters)								
Weight (g)	7	10	20	30	40	50	75	100	**Avg by weight**
0	99.17%	98.46%	98.60%	96.66%	97.36%	97.66%	99.23%	98.63%	98.22%
56	77.13%	98.36%	86.04%	98.50%	92.69%	81.77%	57.00%	22.87%	76.80%
112	99.47%	90.65%	82.90%	88.75%	90.65%	77.06%	66.58%	42.50%	79.82%
168	95.79%	94.39%	95.82%	90.18%	97.90%	86.61%	97.50%	95.06%	94.16%
224	92.55%	89.58%	92.29%	91.35%	84.47%	73.46%	53.29%	64.67%	80.21%
280	97.60%	96.09%	93.19%	94.26%	88.01%	80.50%	88.48%	91.72%	91.23%
Distance Avg	93.62%	94.59%	91.47%	93.28%	91.85%	82.84%	77.01%	69.24%	
									Total Device Avg
									86.74%

The total accuracy is 86.74%. The same trends and causes for the results in case study 2 are present here. Overall, the GS20 performs better than the GP4 but worse than the iP11P, RVMP and GS8. Whether there is a statistical difference in accuracy needs to be determined. These results also refute the assumption that prediction quality increases with newer phones. The iP11P release date is 2020, yet it performs worse than the GS8 (2017) [39].

Table 31 shows the runtime for the GS20.

Table 31: Case Study 3 GS20 Runtime

	AI Train Time (s)	**min**
1 AI	259.5	4.3
all 5	1381.0	23.0

The CSVM runtime or 1 AI is 4.3 min and 23.0 min for all 5 AIs required for 5-cross validation. The results are nearly the same as the runtime the iP11P, RVMP, GS8, and GP4. Indicating the CSVM renders the same result regardless of device.

Figure 46 shows the graph and trendline of the GS20 data.

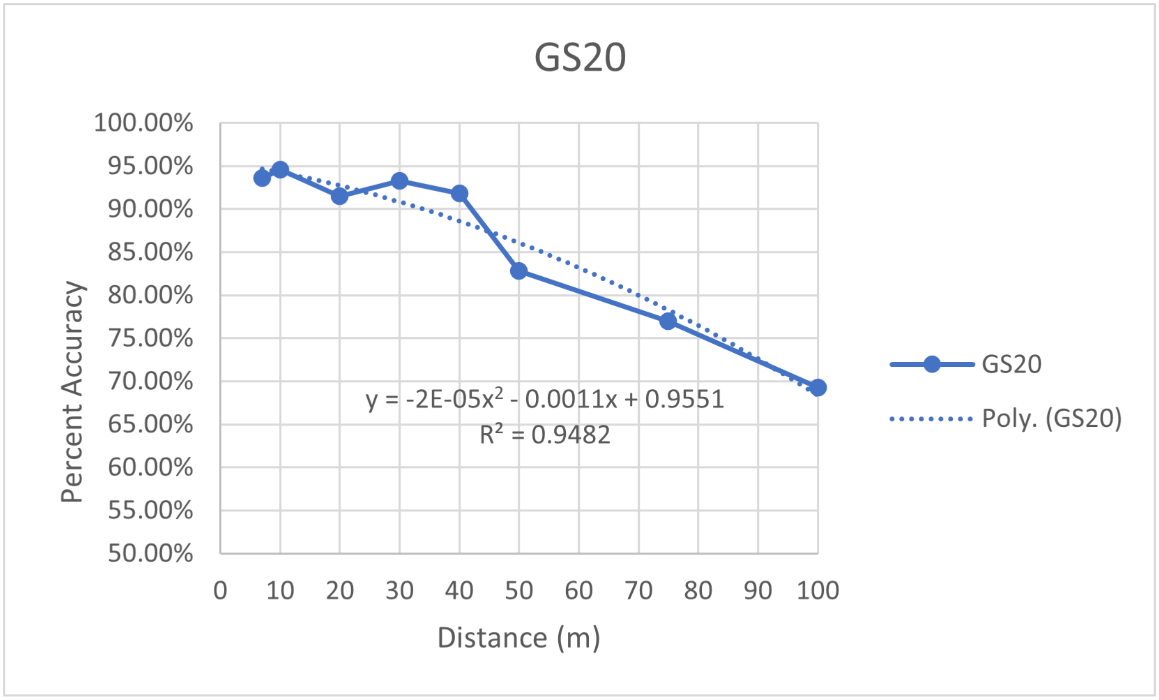


Figure 46: Case Study 3 GS20 CSVM With Trendline

The trendline is:

$$y = -2 * 10^{-5} * x^2 - .0011x + .9551 \tag{13}$$

with an $R^2 = 0.9482$. The prediction results follow a second degree polynomial curve as distance increases. Therefore, prediction accuracy for distances not collected can be predicted. For example, the prediction for 125 m is 50.51%.

5.4.4 Data Comparison

Figure 47 includes all prediction results for all 5 recording devices.

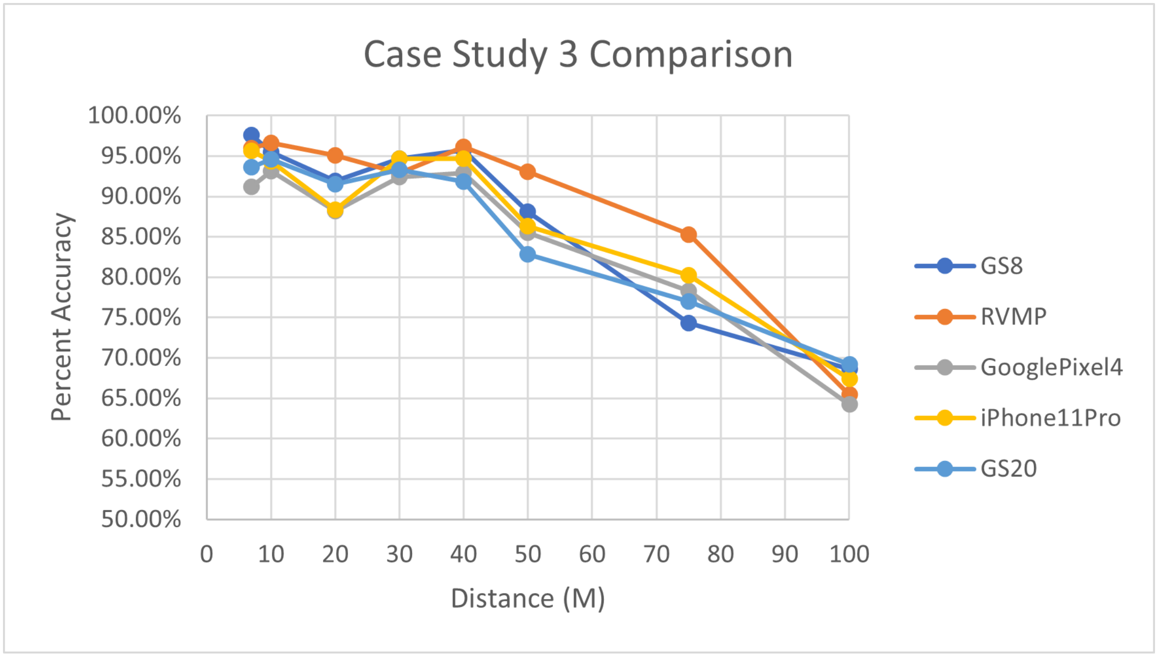


Figure 47: Case Study 3 CSVM Comparison

The graph indicates the same prediction behavior for all recording devices. From 7 m to 100 m the optimal recording device fluctuates among the five. Then, the RVMP remains optimal from 40 m to just before 100 m. Table 32 summarizes the results for all recording devices.

Table 32: Case Study 3 Result Comparison

Device	**Total Accuracy**	**trendline (y)**	**R^2**	**y(125)**	**runtime (min)**
RVMP	90.05%	-5*10^(-5)*x^2 + 0.0025*x + 0.9345	0.9702	46.48%	17.7
GS8	88.30%	-2*10^(-5)*x^2 - .0006*x+0.9725	0.9256	58.50%	22.1
GP4	85.73%	-4*10^(-5)*x^2 + .0015x+.9049	0.9560	46.74%	18.9
iP11P	87.70%	-3*10^(-5)*x^2 + .0006x+.9366	0.9187	54.29%	20.2
GS20	86.74%	-2*10^(-5)*x^2 - .0011x+.9551	0.9482	50.51%	23.0

The RVMP has the highest accuracy at 90.05%. Of the cell phones, the GS8 performs the best at 88.30%. The accuracy margin between the RVMP and GS8 is small, and the GS8 out performs the RVMP at certain distances. So, there is little difference between a high-quality microphone and a cell phone.

Among the cell phones, the results indicate there is a difference between cell phone models. The older model, the GS8, performs better than the newer models. This difference could be due to design choices for newer cell phones. Recording quality may not be a design goal for newer phones. Or, the physical design of the phone may be impacting the recording quality at farther ranges. More research should be done to determine the exact reason for this trend, but it is clear the older model cell phone (GS8) outperforms the newer models.

Additionally, the runtime for each AI is relatively the same. The differences are most likely due back ground processes running on the CPU during the build. Therefore, the data indicates a similar runtime for an SVM with a cubic kernel.

As distance increased, accuracy decreased for each device. The decrease in accuracy for each device follows a second degree polynomial. Figure 48 shows the second degree trendline for each device.

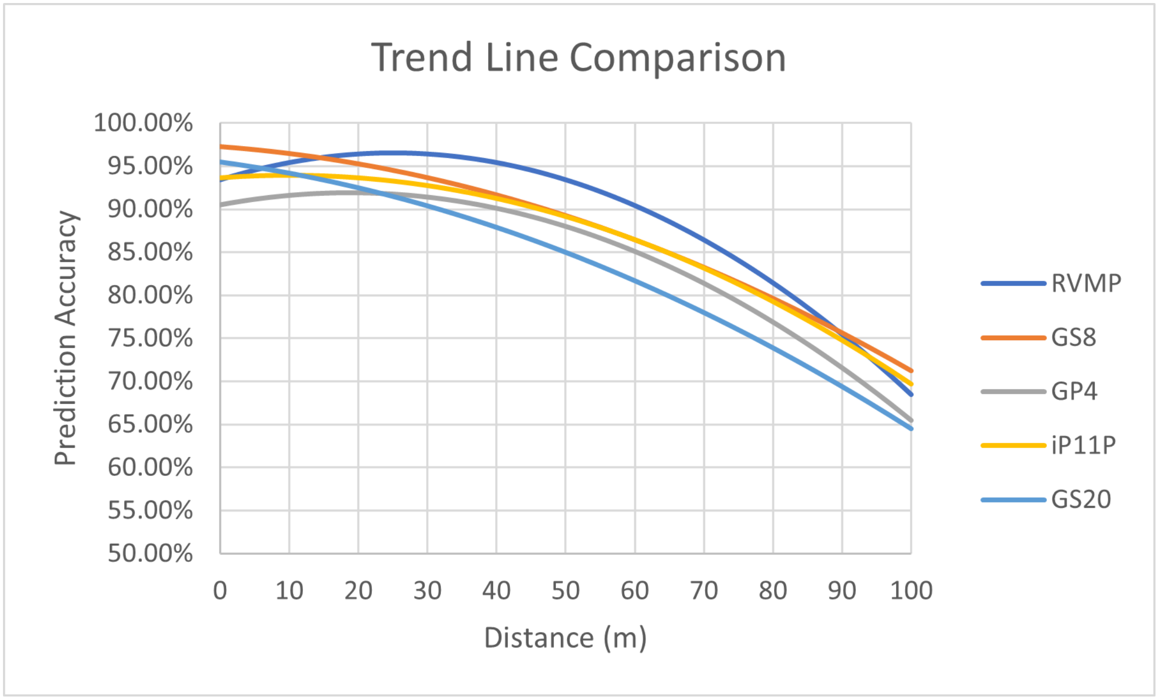


Figure 48: Case Study 3 Trendline Comparison

The observation that each recording device follows a similar second degree polynomial suggests there may be common function of prediction accuracy over distance for all recording devices. In other words, a general second degree polynomial may be able to describe the effect weight has on acoustic emissions of a UAV. Further work should be done to support this claim.

5.5 Case Study 4

Case study 4 answers the question "Does the manufacturing process for a specific recording device affect detection accuracy?". This study uses three GS8s to collect acoustic emissions during the experiment in Chapter 4. The data from two of the GS8s trains a CSVM, and data from the third GS8 tests the CSVM. In theory, the same accuracy trends for the GS8 in case study 2 should be observed here unless there is a manufacturing impact between GS8s. If there is a manufacturing impact, the AI needs data from every single recording device for adequate acoustic payload detection, and this need would be unfortunate.

The reason for this study is because AFRL/RYAA (the sponsor of this research) is interested in deploying a network of cell phone like recording devices for UAV payload detection. This study indicates whether data from every recording device is needed for a useful detection system, or if data only needs to be collected by one device if the same model is used throughout the network. In other words, if 10 microphones are deployed for acoustic detection, does the SVM need training data from all 10 devices or just one.

Table 33 summarizes the results for this case study.

Table 33: Case Study 4 CSVM Results

	CSVM Prediction Results 7 - 100 m								
	Distance (meters)								
Weight (g)	7	10	20	30	40	50	75	100	**Avg by weight**
0	99.80%	97.30%	97.80%	99.13%	99.70%	99.50%	99.17%	97.37%	98.72%
56	95.30%	99.87%	86.26%	97.50%	96.97%	84.46%	42.81%	7.07%	76.28%
112	99.23%	97.63%	79.86%	89.76%	87.93%	73.69%	50.85%	32.14%	76.39%
168	98.27%	96.07%	65.09%	71.02%	93.70%	65.37%	75.83%	59.05%	78.05%
224	81.03%	85.93%	93.90%	85.83%	82.76%	78.89%	62.15%	76.56%	80.88%
280	95.97%	98.17%	91.63%	74.59%	67.62%	91.30%	89.16%	83.76%	86.53%
Distance Avg	94.93%	95.83%	85.76%	86.31%	88.11%	82.20%	70.00%	59.33%	
									Total Device Avg
									82.81%

The total prediction average is 82.81%. This is significantly lower than the total average for the GS8 in case study 2. In case study 2, the average is 88.30%. This results shows that the AIs performance decreases when the test data and training data are not from the same device. Therefore, there is evidence to suggest the manufacturing process for a specific recording device does affect accuracy.

Other factors may be causing this result such as CPU activity on the cell phones. During recording, it is likely that the CPUs on each GS8 are preforming tasks other than recording and that these tasks are not the same for all three devices. Therefore, these processes may be causing the decline in performance present in this case study. However further research should be done to support this claim. Either way, there is a decline in accuracy when training data and testing data for the AI are not from the same device.

Although the accuracy for case study 4 is different than the accuracy in case study 2, the behavior of the accuracy over distance is the same. Figure 49 shows the accuracy over distance for case study 2 and 4.

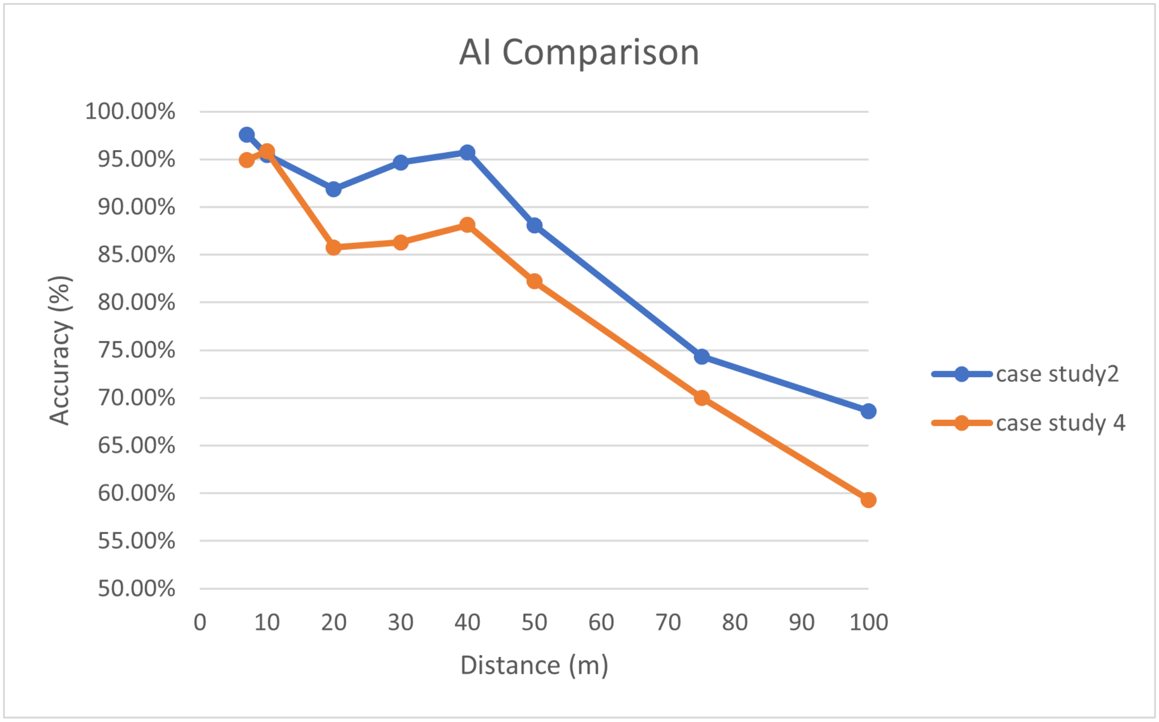


Figure 49: Case Study 2 And Case Study 4 Comparison

The prediction accuracy follows the same pattern as the accuracy in case study 2. Although the accuracy is less, it follows the same trend, and the difference in accuracy for all distances remains nearly the same. Because the same trend is observed, the model for case study 4 is reliable. Future work needs statistical analysis to determine if the difference in accuracy is significant, and if so at which distances is it significant. Given the accuracy exhibits the same behavior as previous case study, the accuracy follows a second degree polynomial. Figure 50 shows the second degree polynomial.

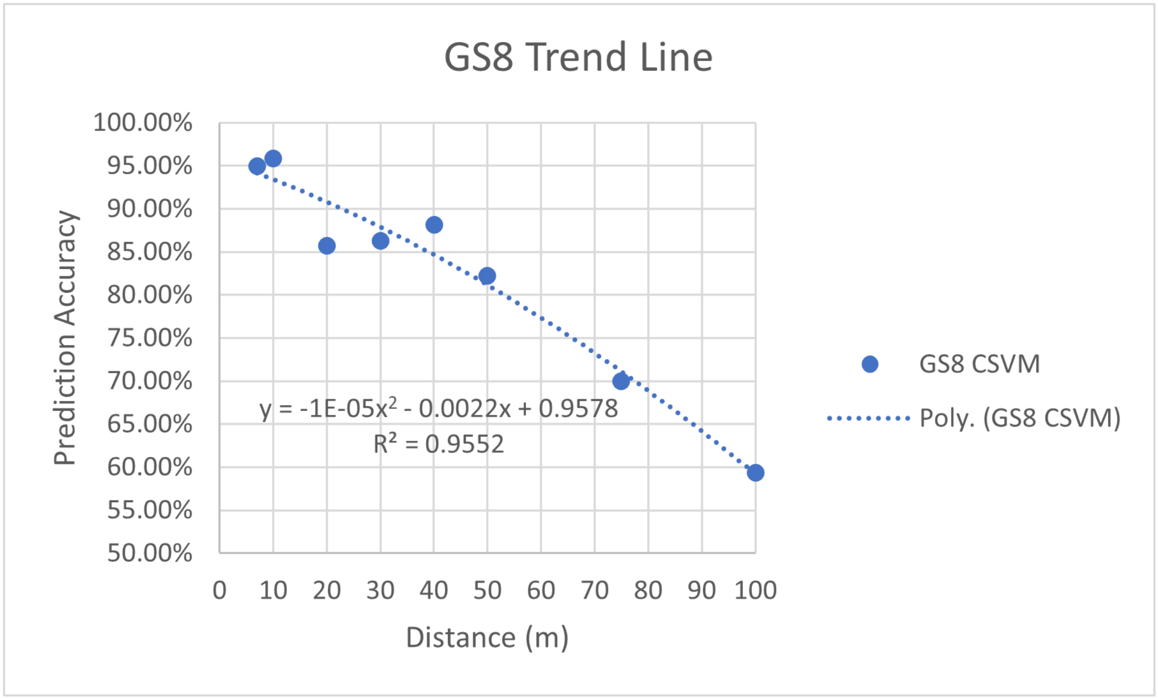


Figure 50: Case Study 4 GS8 Trendline

The results show the prediction results follow the curve:

$$y = -1 * 10^{-5} * x^2 - .0022x + 0.9578 \tag{14}$$

with an $R^2 = 0.9552$. This prediction curve follows the same behavior Case Studies 2 and 3 present. Therefore, it is possible to predict accuracy for distances not recorded even when different devices provide training and testing data. Given a second degree polynomial provides an adequate trendline for all scenarios, there may be a second degree polynomial which models the behavior of acoustic payload prediction across all devices.

Table 34 shows the runtime to build the CSVM for case study 4.

Table 34: Case Study 4 AI Runtime

	AI Train Time (s)	**min**
1 AI	959.34	15.989

The runtime to build the AI in this case study is 15.99 min. Because the entire data set from two phones provides the training data, there is no risk of accidentally choosing really good or bad data. Therefore, case study 4 does not need to perform k-cross validation. Also, this study uses over double the training data than case study 2 and 3, and five times the testing data. So, it is difficult to compare the runtimes between this case study and the others. Still the cubic SVM needs about the same time as the other case studies. This is due to the same kernel and using the same device for processing. The result indicates that with the hardware in this research, a cubic SVM needs about 15 min to 20 min to build.

5.6 Nuisance Factors

This section discusses the nuisance factors present in the experiment. The three largest factors are wind, time of day, and battery life. Wind and time of day affect the prediction results. These factors are uncontrollable in the experiment configuration Chapter 4 describes. Additionally, only two drone batteries were available and one battery charger available on the flight line. Each battery provides ∼12 min of flight time, and it takes 2 hours to charge a battery. So, the recordings were spread out over the days. The different times of flight affect the experiment results. This is because the winds varied throughout the day, with the highest winds present in the early afternoon. Recordings done during high winds show lower prediction accuracy than those during low winds. Figure 51 shows the prediction accuracy over time of day. The figure shows results from the GS8 in case study 2. All phones show this same trend.

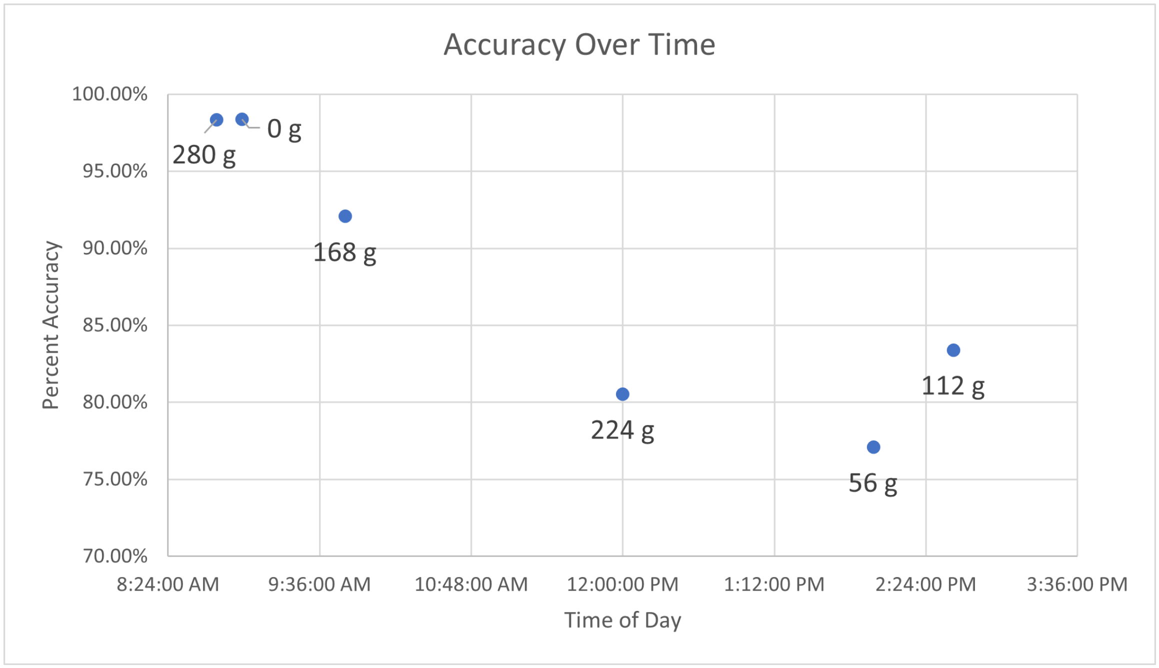


Figure 51: Prediction Accuracy Over Time Of Day

Because of limited battery life, the recordings for each weight were done at all distances in one flight. If battery life is plentiful, the recording order should be completely randomized, but this was not possible under the constraints for this experiment. However, the order of weight classes was randomized by writing each weight on its own piece of paper and picking them out of a hat. The order is as follows: 280 g, 0 g, 168 g, 224 g, 56 g, 112 g. Therefore 280 g and 0 g were ran in the morning, 224 g was ran around noon, 56 g was ran in the early afternoon, and 112 g was ran in the late afternoon.

As the day went on, there was no wind in the early morning, the winds started around mid morning, it reached its peak in the early afternoon, and started to trail off in the mid afternoon. Figure 51 shows how accuracy changed throughout the day, and that this changes is directly correlated to wind noise. Table 35 shows the wind noise readings for each weight.

Table 35: Wind Noise

Weight	**Wind Noise (dB)**
280	44.0
0	45.0
168	49.7
224	55.4
56	62.5
112	54.7

The results shows wind noise has a clear effect on the system. When the noise level is below 50 dB, the system remains fairly accurate, but at above 54 dB there is a significant decrease in accuracy. The wind noise begins to drown out the drone acoustic emissions somewhere between 50 dB and 54 dB. Further research needs to find the exact threshold.

5.7 Summary

This section discusses the results from the experiments in Chapter 5. Overall, the results show a decrease in prediction accuracy as distance between the drone and microphone increases. There is a steady decline in accuracy after 40 m ground distance, and the prediction for each device follows a second degree polynomial. The type of recording device does affect the prediction results. This research determines the optimal configuration is using the GS8 with a CSVM.

VI. Conclusion

6.1 Overview

This chapter summarizes the entirety of the research. Section 6.2 reviews the problem this research seeks to answer. Section 6.3 concludes the findings of this research and how it relates to the problem at hand. Section 6.4 discusses the contributions this research makes to the overall body of literature in this field. Section 6.5 discusses the limitations of this research. Section 6.6 recommends future work to expand this research.

6.2 Problem Review

The growing research of UAVs brings new security threats. Whether in the civilian world, corporate world, or the military front, drone detection systems are imperative to handle these threats. Many platforms use technology such as RF, RADAR, and image processing to detect drones, but few use acoustic emissions and nearly none detect UAV payload.

Payloads are a necessary attribute to detect for a UAV due to the growing presence of drone delivery systems and drone IEDs. The ability to detect and classify drone payload weight would help mitigate these threats. Acoustics are a unique attribute emitted by all drones which can not only detect payload weight, but do not require line of sight for detection. Acoustics are also more difficult to spoof than technologies such as RADAR or RF. Therefore, there is a need for a drone payload detection system based on acoustic emissions.

6.3 Research Conclusions

The goals of this research were to demonstrate an acoustic payload detection method using cell phone devices and determine the maximum range of acoustic payload detection. To fulfill these goals, an acoustic payload detection system called HurtzHunter was built to demonstrate and test acoustic payload detection with cell phone devices and a high-quality microphone. Past research successfully demonstrated acoustic payload detection at close range (7 m) and with a high-quality microphone (the RVMP). To expand the field of study, this research hypothesizes that cell phones are able to determine the weight of a payload carried by a UAV from acoustic emissions at up to 100 m away ground distance from the UAV.

To achieve these goals, this research conducts an experiment with the HurtzHunter design. This experiment, summarized in Chapter 4, implements the System Under Test (Figure 29), and Figure 30 shows the experiment design. In the experiment, the HurtzHunter prototype uses 7 recording devices and 1 quad-copter drone. The recording devices are the three Samasung Galaxy S8 (GS8), one Samsung Galaxy S20 (GS20), one Google Pixel 4 (GP4), one iPhone 11 Pro (iP11P), and one RodeVideoMicPro (RVMP). The drone is the Mavic Air 2. The microphones collect acoustic emissions while the drone hovers for 1 min. The microphones collect acoustics for 48 flights in total. Flights are conducted at 7 m, 10 m, 20 m, 30 m, 40 m, 50 m, 75 m, and 100 m away from the microphones, and at each distance the drone performs 6 flights, each with a different payload weight. The payload weights are 56 g, 112 g, 168 g, 224 g, and 280 g. HurtzHunter uses the sound data from the microphones to train and test an SVM.

Four case studies use the data from this experiment to answer the following questions:

- Can cell phones detect UAV payload weight with sound?

- What is the maximum range of acoustic UAV payload detection?
- Does the cell phone model affect detection accuracy?
- Does a high-quality microphone perform better than a cell phone for acoustic payload detection?
- Does the manufacturing process for a specific recording device affect detection accuracy?

Case study 1 conducts similar methodology in past research with the addition of a cell phone device. Only using the acoustic emissions from 7 m, a linear, cubic, and quadratic SVM is built and tested for both the RVMP and the GS8. All SVMs for the RVMP and GS8 resulted in $98\% - 99\%$ payload prediction accuracy. This study finds little to no difference in payload prediction accuracy between the GS8 and RVMP for the linear, quadratic, and cubic SVMs. This means at close range (7 m) there is no difference between a high-quality microphone and a cell phone device. Additionally, the results are consistent with past research which achieved $\sim 98\%$ accuracy with the RVMP at 7 m. This consistency with past research provides experiment validation.

Case study 2 builds on case study 1 by expanding the detection range from 7 m to 100 m. The results show little difference in prediction accuracy between the GS8 and RVMP for all distances. The results also show a significant difference in AI build time between the linear, quadratic, and cubic SVMs. The linear SVMs for both recording devices needed 6-9 hrs of build time, while the quadratic and cubic SVMs needed 16-20 min. The CSVM is the optimal choice because it has the lowest build time and the same accuracy as the linear and quadratic SVMs. With the CSVM, the RVMP achieves a 90.05% overall accuracy, and the GS8 achieves a 88.30% overall accuracy. From 7 m - 40 m the prediction accuracy remains greater than 91% for both devices. After 40 m, there was a significant drop in accuracy observed for both devices.

The correlation between distance and accuracy follows a quadratic curve for both devices, and case study 2 provides a prediction trendline which can be used for distance accuracy not measured in this research. The results of this study show the GS8 performs the same as the RVMP from 7 m to 40 m. From 40 m to 100 m the RVMP performs better, but at 100 m the GS8 performs better. This results means that at distances less the 40 m, a cell phone device performs the same if not better than a high-quality microphone. Also, the results indicate adequate payload detection is achievable at up to 40 m away ground distance with a cell phone device.

Case study 3 builds upon case study 2 by expanding the research to include 3 additional cell phone devices for recording. This study builds and tests a CSVM for the GP4, iP11P, and GS20 using data from 7 m to 100 m. The GP4 overall accuracy is 85.73%, the iP11P overall accuracy is 87.70%, and the GS20 overall accuracy is 86.74%. After comparing the prediction results from these devices with the GS8 from case study 2, the GS8 has the best overall prediction results with an 88.30% overall accuracy. This result is unexpected as the GS8 is the oldest model cell phone. The exact reason for this phenomenon is unknown, but possible solutions could be that new cell phone technology prioritizes performance for things other than recording quality. Therefore, there is a difference in payload prediction accuracy across cell phone models, and the GS8 is the optimal choice for acoustic payload detection.

Case study 4 explores the effect of manufacturing on prediction quality. In other words, if the exact same recording device is used for payload detection, how does the accuracy change from device to device. It is not practical to train an AI with data from every single recording device in a real world scenario. So, if a detection system was built using the GS8, would the user be able to take any GS8, load the system on the device, and use it for payload detection? In this study, acoustic emissions from two GS8s is used to train a CSVM, and acoustic emissions from a third GS8 is used to test the CSVM. The results from this case study show the prediction accuracy is 82.81%. This accuracy is a significant decrease from case study 2 where the accuracy for the GS8 is 88.30%. Therefore, the manufacturing from device to device does have an effect, and the optimal solution would need training data for the AI from every single device. However, from 7 m - 40 m the accuracy remains roughly around 88%, so at close range the manufacturing impact on the device may not be detrimental.

Over all, this research finds that the GS8 is the optimal cell phone device for acoustic payload detection, and the maximum range for accurate acoustic detection using cell phone devices is 40 m. This finding proves the hypothesis to be true at up to 40 m rather than 100 m. To restate, cell phones are able to determine the weight of a payload carried by a UAV from acoustic emissions at up to 40 m away ground distance from the UAV.

There is little difference in accuracy between the GS8 and the RVMP for acoustic payload detection from 7 m to 100 m. Additionally, the best configuration for the HurtzHunter design is to use the GS8 with a CSVM. The CSVM has the fastest runtime and the highest prediction accuracy across many scenarios. When using this platform, the optimal range is 7 m to 40 m. There is a significant drop in accuracy after 40 m which follows a quadratic curve. However, a 40 m range would prove useful as the hazardous payloads in Table 4 have a blast radius of 4 m to 5 m, making the accurate 40 m range for acoustic payload detection a safe distance. In other words, the HurtzHunter design is capable of detecting a hazardous UAV before it becomes lethal.

6.4 Research Contributions

Past research accomplished acoustic payload detection at close range (7 m). Their methodology used a high-quality microphone and a MacBook Pro as the recording device [7][8]. This research expands the field by testing acoustic detection from 7 m to 100 m, and on many difference cell phone devices. To test these capabilities, this research creates the HurtzHunter Prototype design. This design deploys 6 cell phone devices and one high-quality microphone to collect acoustic emissions. The prototype processes the acoustic data to build an AI capable of acoustic payload detection and test the AI's performance. The methodology in this research uses the HurtzHunter prototype to test different scenarios, and the results show the HurtzHunter prototype achieves 82.81 - 99.93% prediction accuracy based on the configuration. In short, this research provides novel insight into the maximum range for UAV payload detection using acoustic emissions and provides insight into the ability to use cell phone devices or similar IoT devices for detection.

6.5 Limitations of This Research

The experiments in this research are conducted on the flight line at the Air Force Bombing Range in Avon Park, FL. The limited access to this range contributes to many limitations in this research. One limitation is that this research accomplishes only one repetition of the experiments. The statistical analysis in this research is limited by one repetition because there is zero degrees of freedom. Furthermore, this research is limited to data collected in the environment of the bombing range. Ideally, there would be acoustic data collected at different times from different environments to prevent overfitting the data in the AI, but this is not possible under the experimental limits of this research.

Also, this research only uses one drone for acoustic emissions. There was not enough time to fly multiple drones during experimentation. So, there is no away of knowing if the AIs built in this research only work for the specific drone used in this research or for every drone.

Wind and time of day are two major nuisance factors in this experiment. The winds vary throughout the day in Avon Park, FL. Recordings made in the afternoon include higher wind noise than those in the morning. Recordings with high wind noise greatly decrease the prediction quality. In addition to wind, the data was collected in august when the temperature and humidity are at its highest in Florida. The hot humid climate may affect the amount of lift needed to fly a payload. If so, the acoustic signatures may be different based on the climate, and a detection system built in one environment may not work in another. This research is not able to assess the affect of climate on the prediction accuracy because all the experiments were conducted in one place.

6.6 Recommendations for Future Work

The experiment in this research needs to be conducted for multiple repetitions. With more degrees of freedom, the researcher can determine at what distances or with what recording devices there is a significant statistical difference between prediction accuracy.

Future research needs to use the HurtzHunter design with multiple drones and compare the results. Different drones may emit different acoustic signatures even though they carry the same payload.

Acoustic emissions need to be collected in multiple different climates, at different times of the day, and at different times of the year. Increasing the diversity of the training and testing data for the AI in the HurtzHunter design provides evidence for overfitting. If overfitting is present, new methodology is needed for an adequate acoustic detection system.

Wind noise and time of day greatly decrease the acoustic payload prediction accuracy. Future work needs to study the exact cause for this decrease and how to mitigate it.

This research finds the GS8 renders the best acoustic prediction accuracy. The exact cause for this phenomenon is unknown, and this research makes the general conclusion that older model cell phones perform better than new ones. Future work is needed to verify this conclusion and determine the exact cause for better prediction results from the GS8.

6.7 Summary

This research demonstrates the ability to use UAV acoustic emissions to identify payload at up to 100 m away from the drone. This research develops the HurtzHunter design to show the ability to use cell phone devices for payload detection. This design is deployable on any small computing devices with recording capabilities similar to the cell phone. Based on the configuration, the procedures demonstrated in this research are capable of achieving a payload prediction accuracy between 82.81 - 99.93%.

Appendix A. iTunes Conversion Instructions

To convert the .m4a to .wav file using iTunes follow these steps:

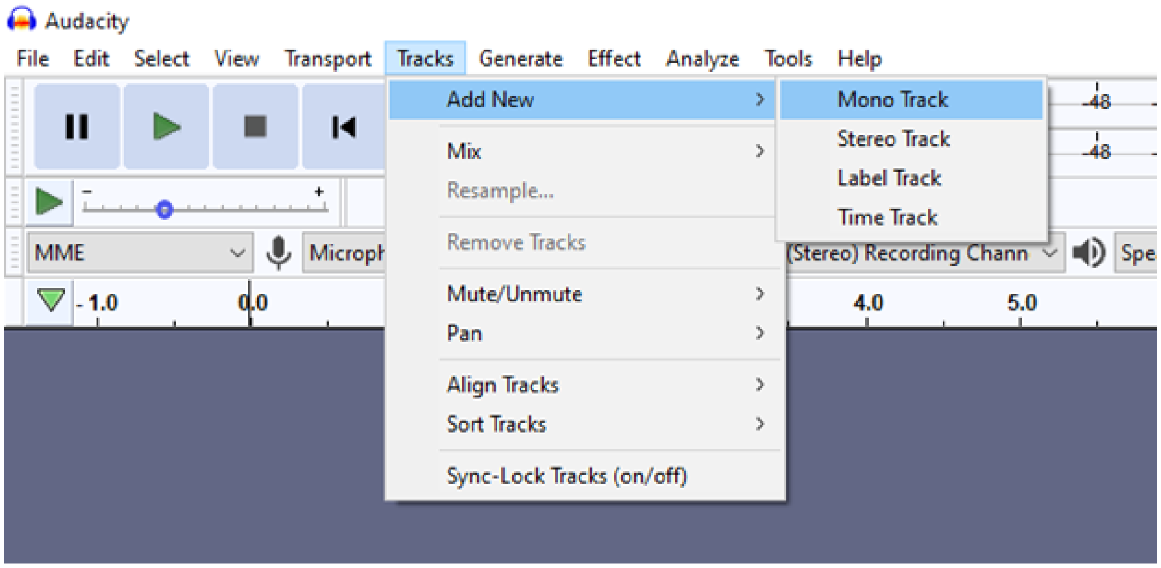
1. Open iTunes
2. Click ”edit” at the top left in the tool bar
3. Select Preferences
4. Select Import Settings...
5. In the Import Using Selection choose ”WAV Encoder”
6. Select ok
7. Select ok
8. Highlight the audio track to convert
9. Select File at top left in the tool bar
10. Select convert
11. Select Create Wave Version
12. After clicking ”Create Wave Version” a new version of the audio file is created in the .wav format and exists in the computer’s file system where the iTunes files are stored.

Appendix B. Audacity Use Instructions

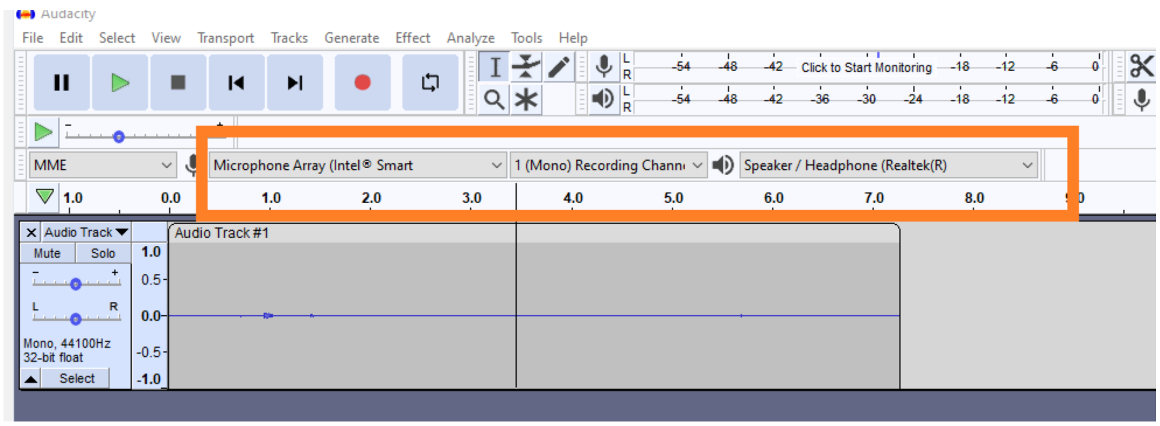
Use the following steps to create an audio recording and store the recording on the computer:

1. Open Audacity

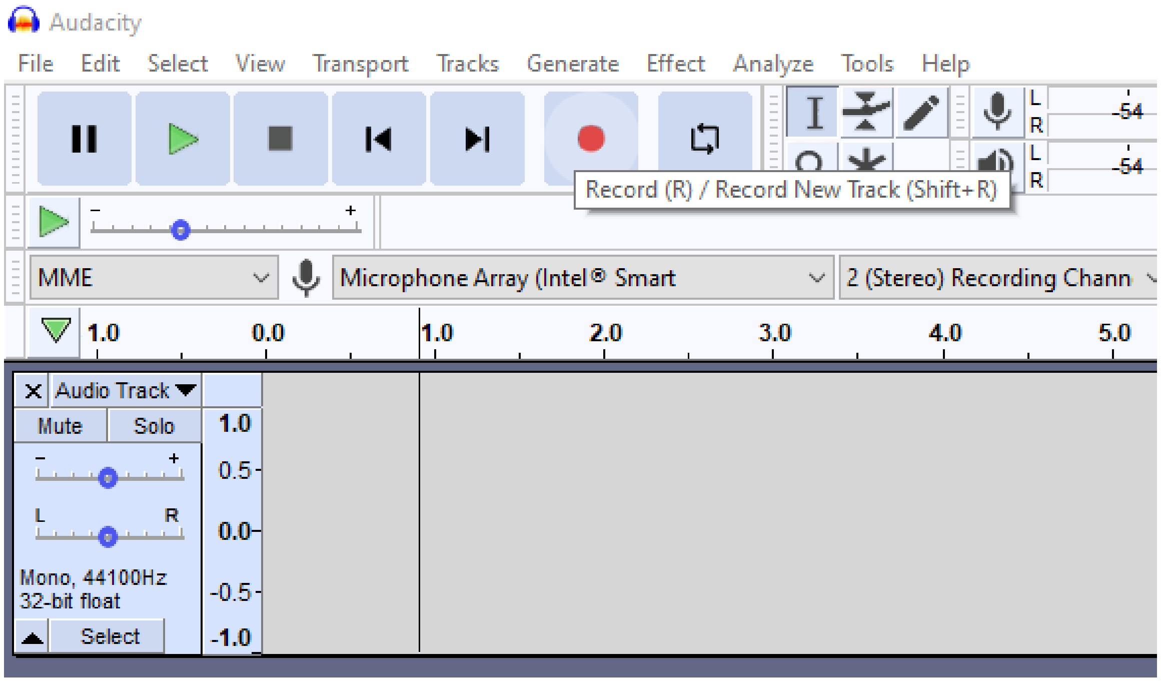
2. Create a new track: select "Tracks", then select "Add New", then select "Mono Track"



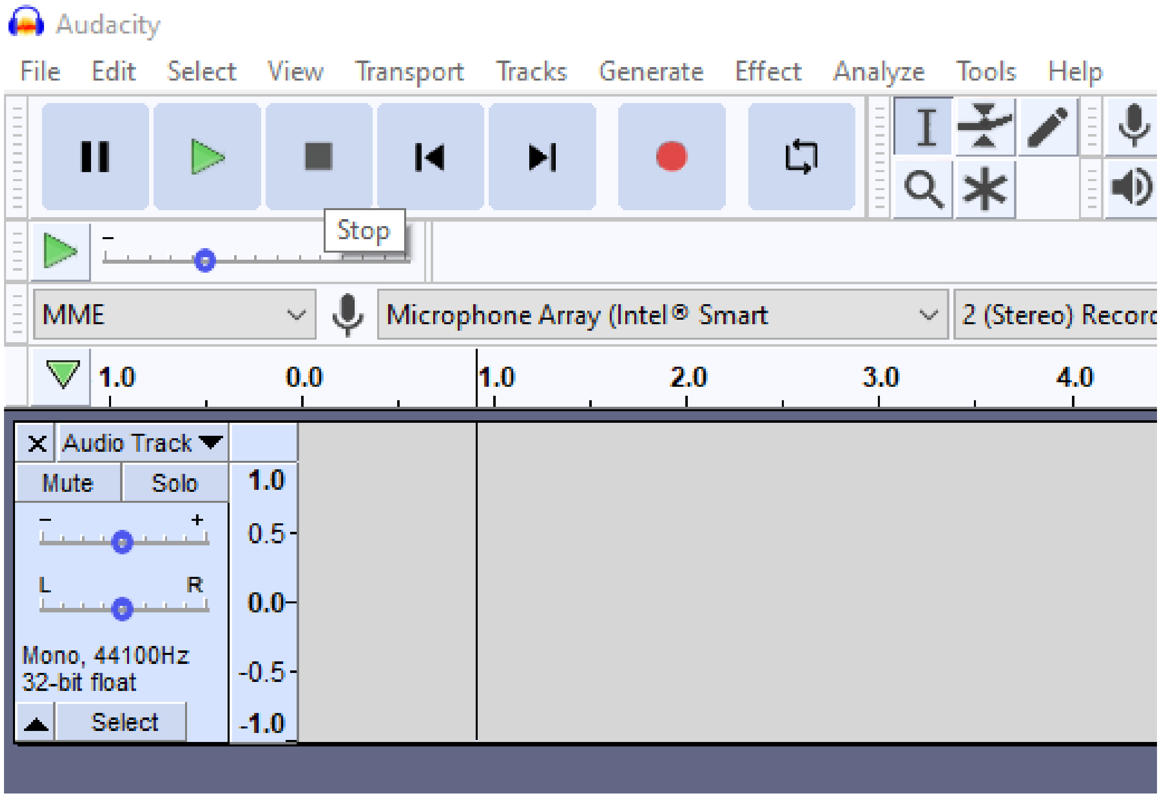
3 Set the Recording channel to mono, and ensure the microphone and speakers are set the desired devices.


Audacity
File Edit Select View Transport Tracks Generate Effect Analyze Tools Help
Click to Start Monitoring
MME
Microphone Array (Intel® Smart
1 (Mono) Recording Chann
Speaker / Headphone (Realtek(R)
Audio Track
Mute Solo
Mono, 44100Hz
32-bit float
Select
Audio Track #1

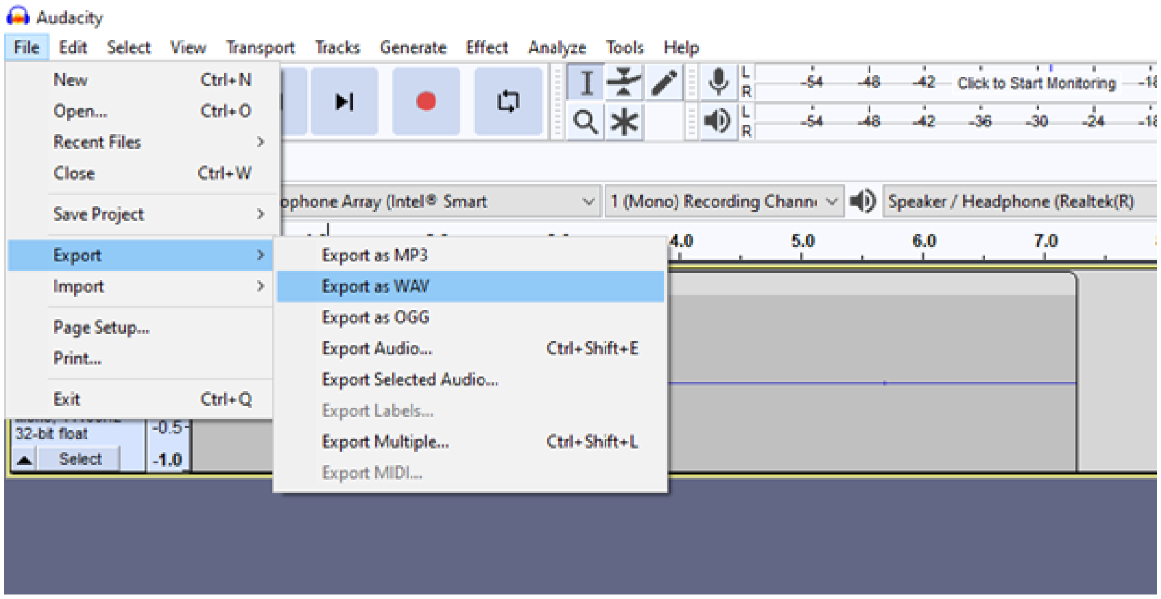
4. To start recording select the red circle at the top left.


Audacity
File Edit Select View Transport Tracks Generate Effect Analyze Tools Help
Record (R) / Record New Track (Shift+R)
MME
Microphone Array (Intel® Smart
2 (Stereo) Recording Chann
Audio Track
Mute Solo
Mono, 44100Hz
32-bit float
Select

5. To stop recording press the square block at the top left of the screen to the right of the green arrow.


Audacity
File Edit Select View Transport Tracks Generate Effect Analyze Tools Help
Stop
MME
Microphone Array (Intel® Smart
2 (Stereo) Record
Audio Track
Mute Solo
Mono, 44100Hz
32-bit float
Select

6. Select the track, then select file, Export, Export as WAV



After selecting Export as WAV, the computer's file system will come up, save the audio file where the user wants.

Bibliography

1. The Editors of Encyclopaedia Britannica. Joseph-Michel and Jacques-Etienne Montgolfier, 2018. Accessed: 10 December 2021 [Online]. Available: https://www.britannica.com/biography/Montgolfier-brothers.

2. Federal Aviation Administration. FAA Aerospace Forecast 2019-2039, 2019. Accessed: 4 January 2022 [Online]. Available: https://www.faa.gov/data_research/aviation/aerospace_forecasts/media/FY2019-39_FAA_Aerospace_Forecast.pdf.

3. Navigating Drone Prices: A Guide to What to Expect at Different Price Points When Buying a Drone, 2019. Accessed: 03 June 2021 [Online]. Available: https://uavcoach.com/drone-prices/.

4. Harini Kolamunna, Thilini Dahanayaka, Junye Li, Suranga Seneviratne, Kanchana Thilakaratne, Albert Y. Zomaya, and Aruna Seneviratne. DronePrint: Acoustic Signatures for Open-set Drone Detection and Identification with Online Data. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 5(1), 2021. pp. 1-31. Accessed: 21 January 2022 [Online]. Available: https://dl.acm.org/doi/abs/10.1145/3448115/.

5. S. Snow. Syrian fighters in Raqqa uncover ISIS drone factory, 2017. Accessed: 10 December 2021 [Online]. Available: https://www.militarytimes.com/flashpoints/2017/07/26/syrian-fighters-in-raqqa-uncover-isis-drone-factory/.

6. T. Wright. How many drones are smuggling drugs across the US southern border? Accessed: 10 December 2021 [Online]. Available: https://www.airspacemag.com/flight-today/narcodrones-180974934/.

7. Omar Adel Ibrahim, Savio Sciancalepore, and Roberto Di Pietro. Noise2Weight: On Detecting Payload Weight from Drones Acoustic Emissions. pages 1–14, 2020. Accessed: 14 January 2022 [Online]. Available: https://arxiv.org/abs/2005.01347.

8. Ahmad Traboulsi and Michel Barbeau. Identification of Drone Payload Using Mel-Frequency Cepstral Coefficients and LSTM Neural Networks. 1288, 2021. pp. 402-412, Accessed: 21 January 2022 [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-030-63128-4_30.

9. Amazon. Prime Air, 2016. Accessed: 13 December 2021 [Online]. Available: https://www.amazon.com/Amazon-Prime-Air/b?ie=UTF8&node=8037720011.

10. C. Huang Shen, F. Y.C. Albert, C. K. Ang, Dwee Jin Teck, and K. P. Chan. Theoretical development and study of takeoff constraint thrust equation for a drone. *IEEE 15th Student Conference on Research and Development (SCOReD)*, 2018. pp. 18-22, Accessed: 21 January 2022 December 2021 [Online]. Available: `https://ieeexplore.ieee.org/abstract/document/8305428`.

11. BestBuy. DJI Mavic Air 2 drone with Remote Controller Black CP.MA.00000176.03. Accessed: 4 January 2022 [Online]. Available: `https://www.bestbuy.com/site/dji-mavic-air-2-drone-with-remote-controller-black/6407737.p?skuId=6407737`.

12. Daniel Bonet-Solà and Rosa Ma Alsina-Pagès. A comparative survey of feature extraction and machine learning methods in diverse acoustic environments. *Sensors (Switzerland)*, 21(4), 2021. pp. 1-21, Accessed: 21 January 2022 [Online]. Available: `https://www.mdpi.com/1424-8220/21/4/1274`.

13. S. B. Davis and P. Mermelstein. Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences. *Transactions on Acoustics, Speech, and Signal Processing*, 28(4), 1980. pp. 357-366, Accessed: 21 january 2022 [Online]. Available: `https://ieeexplore.ieee.org/abstract/document/1163420`.

14. Emmanuel Ifeachor Jervis and Berrie W. Jervis. *Digital Signal Processing: A Practical Approach.* Pearson, second edition, 2002. Accessed: 21 January 2022 [Online]. Available: `https://www.pearson.com/us/higher-education/program/Ifeachor-Digital-Signal-Processing-A-Practical-Approach-2nd-Edition/PGM16482.html`.

15. Paul Davidovits. Waves and Sound, 2019. Accessed: 18 January 2022 [Online]. Available: `https://www.sciencedirect.com/topics/mathematics/middle-c`.

16. L. R. Rabiner Schafer and R. W. *Theory and applications of digital speech processing.* Pearson, NJ, 1st edition, 2011. Accessed: 21 January 2022 [Online]. Available: `https://www.pearson.com/store/p/theory-and-applications-of-digital-speech-processing/P100000341585/9780133002539`.

17. B. Bogert, M. Healy and J. Tukey. The Quefrency Alanysis of Time Series for Echoes. In *Proceedings of the symposium on time series analysis*, chapter 15. John Wiley Sons Inc, New York, 1963. pp. 209-243, Accessed: 21 January 2022 [Online]. Available: `https://www.semanticscholar.org/paper/The-quefrency-analysis-of-time-series-for-echoes-%3A-Bogert/15bb1365026071ae3423d64ed2d18c554cafd6f6`.

18. Alan V. Oppenheim and Ronald W. Schafer. From frequency to quefrency: A history of the cepstrum. *IEEE Signal Processing Magazine*, 21(5), 2004. pp. 95-100, Accessed: 21 January 2022 [Online]. Available: `https://ieeexplore.ieee.org/abstract/document/1328092`.

19. Alexandre Kowalczyk. *Support Vector Machines*. Syncfusion, North Carolina, 2017. Accessed: 14 January 2022 [Online]. Available: `https://www.syncfusion.com/succinctly-free-ebooks/support-vector-machines-succinctly`.

20. Riham Altawy and Amr M. Youssef. Security, privacy, and safety aspects of civilian drones: A survey. *ACM Transactions on Cyber-Physical Systems*, 1(2), 2017. pp. 1 - 15, Accessed: 21 January 2022 [Online]. Available: `https://dl.acm.org/doi/abs/10.1145/3001836`.

21. Junkai Peng, Changwen Zheng, Pin Lv, Tianyu Cui, Ye Cheng, and Lingyu Si. Using images rendered by PBRT to train faster R-CNN for UAV detection. *Computer Science Research Notes*, 2802(May), 2018. pp. 13-18, Accessed: 21 January 2022 [Online]. Available: `https://dspace5.zcu.cz/handle/11025/34647`.

22. Eren Unlu, Emmanuel Zenou, and Nicolas Riviere. Using Shape Descriptors for UAV Detection. *IS and T International Symposium on Electronic Imaging Science and Technology*, 2018. pp. 2761-2766, Accessed: 21 January 2022 [Online]. Available: `https://www.ingentaconnect.com/content/ist/ei/2018/00002018/00000009/art00005`.

23. Martins Ezuma, Fatih Erden, Chethan Kumar Anjinappa, Ozgur Ozdemir, and Ismail Guvenc. Micro-UAV Detection and Classification from RF Fingerprints Using Machine Learning Techniques. *IEEE Aerospace Conference Proceedings*, 2019. pp. 1-13, Accessed: 21 January 2022 [Online]. Available: `https://ieeexplore.ieee.org/abstract/document/8741970`.

24. Wenyu Zhang, Gang Li, and Chris Baker. Dictionary Learning for Radar Classification of Multiple Micro-Drones. *2019 International Radar Conference, RADAR*, 2019. pp. 1-4, Accessed: 21 January 2022 [Online]. Available: `https://ieeexplore.ieee.org/abstract/document/9079044`.

25. Muhammad Zohaib Anwar, Zeeshan Kaleem, and Abbas Jamalipour. Machine Learning Inspired Sound-Based Amateur Drone Detection for Public Safety Applications. *IEEE Transactions on Vehicular Technology*, 68(3):2526–2534, 2019. pp. 2526-2534, Accessed: 21 January 2022 [Online]. Available: `https://ieeexplore.ieee.org/abstract/document/8616877`.

26. Andrea Bernardini, Federica Mangiatordi, Emiliano Pallotti, and Licia Capodiferro. Drone detection by acoustic signature identification. *IS and T International Symposium on Electronic Imaging Science and Technology*, 2017. pp. 60-64, Accessed: 21 January 2022 [Online]. Available: `https://www.ingentaconnect.com/content/ist/ei/2017/00002017/00000010/art00009`.

27. Sungho Jeon, Jong Woo Shin, Young Jun Lee, Woong Hee Kim, Young Hyoun Kwon, and Hae Yong Yang. Empirical study of drone sound detection in real-life

environment with deep neural networks. *25th European Signal Processing Conference, EUSIPCO 2017*, 2017. pp. 1858-1862, Accessed: 21 January 2022 [Online]. Available: `https://ieeexplore.ieee.org/abstract/document/8081531`.

28. Juhyun Kim and Dongho Kim. Neural Network based Real-time UAV Detection and Analysis by Sound. *Journal of Advanced Information Technology and Convergence*, 8(1), 2018. pp. 43-52, Accessed: 21 January 2022 [Online]. Available: `https://www.dbpia.co.kr/Journal/articleDetail?nodeId=NODE07504681`.

29. Xuejun Yue, Yongxin Liu, Jian Wang, Houbing Song, and Huiru Cao. Software Defined Radio and Wireless Acoustic Networking for Amateur Drone Surveillance. *IEEE Communications Magazine*, 56(4):90–97, 2018. pp. 90-97, Accessed: 21 January 2022 [Online]. Available: `https://ieeexplore.ieee.org/abstract/document/8337902`.

30. Mathworks. Matlab. Accessed: 19 January 2022 [Online]. Available: `https://www.mathworks.com/products/matlab.html`.

31. J. Crook. Audacity Home Page. Accessed: 16 December 2021 [Online]. Available: `https://www.audacityteam.org`.

32. USB sound card,TechRise USB External Stereo Sound Adapter splitter converter with volume control for windows and mac, plug & play no drivers needed. Accessed: 16 December 2021 [Online]. Available: `https://www.newegg.com/p/1B4-08DC-001A5`.

33. HP. HP Pavilion 15. Accessed: 21 January 2022 [Online]. Available: `https://www.hp.com/us-en/shop/mdp/laptops/pavilion-15-344522--1#!&tab=vao`.

34. Welcome to Python.org. Accessed: 16 December 2021 [Online]. Available: `https://www.python.org/`.

35. Project jupyter. Accessed: 16 December 2021 [Online]. Available: `https://jupyter.org/`.

36. VideoMic Pro Compact directional on-camera microphone. Accessed: 16 December 2021 [Online]. Available: `https://www.rode.com/microphones/videomicpro`.

37. Arthur. What kind of microphones are used in cell phones?, 2021. Accessed: 16 December 2021 [Online]. Available: `https://mynewmicrophone.com/what-kind-of-microphones-are-used-in-cell-phones`.

38. R. T. Legend. Where is the microphone on iPhone 11 located?, 2021. Accessed: 21 December 2021 [Online]. Available: `https://descriptive.audio/where-is-the-microphone-on-iphone-11`.

39. DeviceSpecifications. Home Page. Accessed: 21 December 2021 [Online]. Available: https://www.devicespecifications.com/en.

40. Smart recorder – high-quality voice recorder - apps on Google Play. Accessed: 22 December 2021 [Online]. Available: https://play.google.com/store/apps/details?id=com.andrwq.recorder&hl=en_US&gl=US.

41. Apple. Voice memos. Accessed: 22 December 2021 [Online]. Available: https://apps.apple.com/us/app/voice-memos/id1069512134.

42. Apple. iTunes. Accessed: 22 December 2021 [Online]. Available: https://www.apple.com/itunes/.

43. 3.1. cross-validation: Evaluating estimator performance. Accessed: 22 December 2021 [Online]. Available: https://scikit-learn.org/stable/modules/cross_validation.html.

44. K-Fold Cross Validation. Accessed: 22 December 2021 [Online]. Available: http://ethen8181.github.io/machine-learning/model_selection/model_selection.html.

45. Sklearn.svm.svc. Accessed: 22 December 2021 [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html.

46. Mavic Air 2 - up your game. Accessed: 22 December 2021 [Online]. Available: https://www.dji.com/mavic-air-2.

47. Lachlan Markay. Scoop: U.S. government buying risky Chinese drones, 2021. Accessed: 21 January 2022 [Online]. Available: https://www.axios.com/federal-law-enforcement-china-drone-4b33aca2-b6f5-43d0-8d36-be1d447af1a0.html.

48. Original U.S. Vietnam era Dutch-made v 40 Mini Hand Grenade - as Used by Navy SEALS. Accessed: 16 December 2021 [Online]. Available: https://www.ima-usa.com/products/original-u-s-vietnam-era-dutch-made-v-40-mini-hand-grenade-as-used-by-navy-seals.

49. Dynamite. Accessed: 16 December 2021 [Online]. Available: https://terraria.fandom.com/wiki/Dynamite.

50. GoPro Hero8 Black Basic Kit. Accessed: 16 December 2021 [Online]. Available: https://www.bhphotovideo.com/c/product/1509585-REG/gopro_hero8_black_basic_kit.html/specs.

51. Time - Time Access and conversions. Accessed: 23 December 2021 [Online]. Available: https://docs.python.org/3/library/time.html.

52. RealPython. A Beginner's Guide to the Python Time Module, 2021. Accessed: 23 December 2021 [Online]. Available: `https://realpython.com/python-time-module/`.

53. What is epoch?, 2021. Accessed: 23 December 2021 [Online]. Available: `https://www.computerhope.com/jargon/e/epoch.htm`.

54. E. Lab. NIOSH Sound Level Meter, 2017. Accessed: 23 December 2021 [Online]. Available: `https://apps.apple.com/us/app/niosh-sound-level-meter/id1096545820`.

55. NIOSH Sound Level Meter App, 2019. Accessed: 23 December 2021 [Online]. Available: `https://www.cdc.gov/niosh/topics/noise/app.html`.

REPORT DOCUMENTATION PAGE			*Form Approved OMB No. 0704–0188*
The public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden to Department of Defense, Washington Headquarters Services, Directorate for Information Operations and Reports (0704–0188), 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202–4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to any penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number. **PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.**			

1. REPORT DATE *(DD–MM–YYYY)* 24–03–2022	**2. REPORT TYPE** Master's Thesis	**3. DATES COVERED** *(From — To)* Sept 2020 — Mar 2022

4. TITLE AND SUBTITLE UAV Payload Identification With Acoustic Emissions And Cell Phone Devices	**5a. CONTRACT NUMBER**
	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S) 2d Lt Hunter G. Doster	**5d. PROJECT NUMBER** 21G532A
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Air Force Institute of Technology Graduate School of Engineering and Management (AFIT/EN) 2950 Hobson Way WPAFB OH 45433-7765	**8. PERFORMING ORGANIZATION REPORT NUMBER** AFIT-ENG-MS-20-M-026
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) AFRL/RYAA 2241 Avionic Cir WPAFB OH 45433-7765 COMM 937-713-8573 Email: Eric.Lam.3@us.af.mil	**10. SPONSOR/MONITOR'S ACRONYM(S)** AFRL/RYAA
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION / AVAILABILITY STATEMENT

DISTRIBUTION STATEMENT A:
APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

The growing presence of Unmanned Aerial Vehicle (UAV) brings new threats to the civilian and military front. In response, the Department of Defense (DoD) is developing many drone detection systems. Current systems use Radio Detection and Ranging (RADAR), Light Detection and Ranging (LiDAR), and Radio Frequency (RF). Although useful, these technologies are becoming easier to spoof every year, and some are limited to line of sight. Acoustic emissions are a unique quality all drones emit. Acoustics are difficult to spoof and do not require line of sight for detection. This research expands the research field of study by creating HurtzHunter, a prototype which tests acoustic payload detection at far range (7 m - 100 m) and with cell phone devices. HurtzHunter uses MFCCs to train a SVM for UAV acoustic payload detection. Depending on the recording device and SVM configuration, the results show an 82-98% payload prediction accuracy using cell phone devices.

15. SUBJECT TERMS

This work is declared a work of the U.S. Government and is not subject to copyright protection in the United States

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON** Dr. Barry E. Mullins, AFIT/ENG
a. REPORT	**b. ABSTRACT**	**c. THIS PAGE**			**19b. TELEPHONE NUMBER** *(include area code)*
U	U	U	UU	174	(937) 255-3636, ext 7979; barry.mullins@afit.edu