Acoustic Side-Channel Attacks on a Computer Mouse

Predicting Mouse Movements through Emitted Audio

Marin Duroyon

Delft University of Technology

TU Delft

Acoustic Side-Channel Attacks on a Computer Mouse

Predicting Mouse Movements through Emitted Audio

by

Marin Duroyon

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Wednesday February 14, 2024 at 8:00 AM.

Student number:	4872355	
Project duration:	April 19, 2023 – February 14, 2024	
Thesis committee:	Prof. dr. ir. M. Conti,	TU Delft, supervisor
	Asst. Prof. dr. ir. R. Marroquim,	TU Delft
	Prof. dr. ir. G. Tsudik,	University of California, Irvine

TU Delft

Preface

This Master Thesis explores the security risk posed by the sounds heard using a computer mouse. As mice are physical systems, they click, grind and squeak as they are used. This study then determines if using those sounds to infer potentially compromising information, such as user activities, is possible. For instance, would it be possible to determine the direction of mouse movements as it slides on a mouse pad?

Cybersecurity has evolved into a personal passion that has followed for a while. When confronted with this research subject, I was amazed at the possibility of diving into an academic gap in side-channel attacks. It fueled my desire to become one of the first publications involving the acoustic leakage model of computer mice. I thank Prof. Mauro Conti and Gene Tsudik for their confidence in this research. Moreover, I would like to express my gratitude for being allowed to travel and discover new study labs during this research period. I would also like to thank Gabriele Orazi, an appreciated supervisor and friend during this thesis.

This thesis will be presented in front of a committee represented by Mauro Conti, Ricardo Marroquim and Gene Tsudik.

Thank you to my friends and family for the support during this work.

Marin Duroyon

Abstract

Acoustic side-channel attacks (SCAs) use audio produced by a system to bypass traditional security measures to extract sensitive information. Human interface devices, such as keyboards, have been the focus of such attacks, however, computer mice are input devices that are currently in a research gap. This paper explores the security risks the emitted mouse sounds pose during usage. The methodology first establishes a proof of concept attack by classifying the mouse movement into up, down, left and right directions. The results lead to a 97% accuracy in distinguishing between the four categories in a controlled environment. This sets the stage by proving a leakage model useful for mouse acoustic SCAs. The research investigated the precision of tracking mouse movements by conducting experiments with ten unique movements on a large mouse pad. The study, using a dual-microphone setup, a smartphone in stereo recording, achieved 95% accuracy in discerning ten different movements. Furthermore, to place the research in a real-world context, the same experiment was repeated by adding two more directions (diagonal movement) and five other participants. The model was trained to become generalizable to six participants and 12 mouse pad movements, resulting in an accuracy of 94%. Given the same environment, this result shows the capability to extract sensitive information using a non-user-specific model. In addition, the paper experimented with a realistic attack scenario to infer a user action of closing a window on a laptop by clicking the red 'X' at the top right of the screen. The trained model could predict with 91% whether a mouse movement and click described the close window event. The experiments and findings within this research confirm audio leakage from a computer mouse in use. Moreover, the SCA poses a security risk in real-world scenarios, as it allows us to trace user activity in a realistic scenario. This work has explored the limits of single microphone use for SCAs and opened the door toward dual-microphone setup for future experiments.

Contents

1

Introduction

"There is no such thing as perfect security" is a common adage in cybersecurity reflecting the notion that given sufficient time and resources, any system can be compromised. Thus, the attacker aims to infiltrate a system while minimizing resource investment. Let us consider, for instance, a hypothetical scenario where an attacker situated in a common environment like a library aims to extract user information from a laptop. There are multiple ways to attack the system; the most expensive attack is a network zero-day on the operating system, costing the malicious actor upwards of millions of dollars. In contrast, a far simpler and less costly tactic could involve 'shoulder surfing' — physically observing over a user's shoulder to extract private information displayed on the laptop screen. Although direct shoulder surfing poses practical challenges, it illustrates how low-tech strategies can circumvent sophisticated security measures effectively. Similarly, microphones present on nearly all user devices are accessible and can be viewed as attacking tools. This thesis investigates the use of microphones for extracting sensitive information, thereby covertly breaching the privacy of laptop users.

Side-channel attacks exploit various unintended emissions from a system to extract sensitive data, such as timing information, power consumption, electromagnetic leaks, or even sound. These attacks are particularly insidious because they do not involve the system's core functionalities but rather its implementation. For instance, by analyzing the time it takes for a system to perform certain cryptographic operations, an attacker can infer the secret key used in the encryption processes. This exploitation of time variance demonstrates the subtlety and effectiveness of side-channel attacks. Moreover, power analysis attacks are another form of attack where the power consumption of a device is measured during cryptographic operations. Within this category are primarily two methodologies: Simple Power Analysis (SPA) and Differential Power Analysis (DPA). SPA involves directly interpreting power consumption measurements to infer data or key information. In contrast, DPA involves statistical analysis of power consumption data collected over many operations to find correlations that leak information about the secret key.

Audio is an interesting data source central to this research attack on computer mice. Acoustic side-channel attacks use audio signals as data inputs or side-channel to assess a system's security. This thesis explores acoustic side-channel attacks, focusing on human interface devices (HIDs), such as keyboards and mice, commonly used with systems like laptops. A notable example in this field is the analysis of keyboard acoustic emanations. The characteristic clicking sounds of a keyboard, often considered innocuous, can be analyzed to infer the keystrokes made by the user. The potency of these attacks lies in their reliance on a low-threat model, requiring only the presence of an attacker equipped with a microphone in proximity to the target device. While considerable research has been conducted on keyboards, exemplified by studies such as [2], the computer mouse still needs to be explored. Since keyboards and computer mice are the primary modes of interaction with laptop systems, this research aims to extend the understanding of potential security vulnerabilities associated with audio side-channel attacks.

The attack scenarios for these attacks are diverse and often depend on the employment strategy. An attacker aims to approach a microphone to the victim's laptop, targeting the mouse. A possible scenario would be a smartphone discreetly placed nearby in an open environment such as a library or open space office. Furthermore, depending on the accuracy of measurements, it would be possible to have a directional microphone placed further away, such as in another building, that could record the audio and predict mouse movements. This type of attack and recording reduces the entry barrier for an attacker to steal the victim's private information, such as user activities on a laptop.

Research into acoustic side-channel attacks has shown the vulnerability that audio emissions pose to the security of HIDs. Many types of attacks allow the leak of information on a user's keystrokes on a keyboard or touchscreen and the content the user observes on an LCD screen. It is notable to remark, however, that mouse movements are never discussed and, as of the writing of this thesis, present an unexplored field. The most similar paper infers user activities based on mouse clicking and scrolling; however, mouse movements are not taken into account.

This research is exploratory, as acoustic side-channel attacks on computer mice are rare in academia. The goal will be to explore the possibilities and limits of acoustic side-channel attacks on computer mice. This is done to complement existing research on keyboards' sound emanations. The objective of this study is to evaluate the correlation between acoustic signals and mouse movements, determining their potential efficacy in enabling side-channel attacks; in order to answer the aim and objectives of this thesis, three research questions are asked:

1. Can the sound produced by a computer mouse movement be used to infer its motion path?
2. If so, what is the accuracy with which we can infer mouse movements from the audio data?
3. What are potential real-world scenarios or environments where acoustic side-channel attacks on mouse movements pose a security risk?

Each question is designed to progressively explore the extent of security vulnerabilities in acoustic emanations from computer mice. The first question is used to establish if there is a viable research path. This is an initial proof of concept to understand if it is possible to infer mouse movements using its sound. The proof of concept step will start with simple experiments to confirm introductory results that could be used to extend the research. The second research question attempts to establish the attack's bounds by defining the side-channel attack's granularity. With a proof of concept, it is important to understand the level of accuracy the attack presents. The final research question will place the acoustic side-channel attack of mouse movements in real-world scenarios to provide a current security analysis of the risks and impacts.

The thesis' goal is to define the security risks posed by the sound produced by computer mice. It is important to define the scope of the research, as there are many different potential experimental setups. As this research is novel, no previous dataset has been generated and will become an important focus for this thesis. The number of data points is a constraint. Constructing a proper dataset required several hours to be collected, making it unrealistic to ask multiple subjects to participate in the initial phases of research. Consequently, until Chapter 7, the results are mostly based on a dataset of a single researcher performing the mouse movements. The primary experimental setup is the main author's laptop, mouse pad, and mouse. While different environments (excluding the laptop) will be explored throughout the research to establish basic levels of accuracy, it is not the intended purpose. The research with different mouse and mouse pads would best be left for cybersecurity surveys at a later stage.

The dataset in this research comprises two primary types of data: movement and sound, with the primary aim being to investigate the influence of mouse movements on the emitted audio. Each data type, movement and sound requires distinct recording techniques and parameters to capture relevant information accurately. Our approach primarily leverages machine learning techniques applied to acoustic signals. We employ a method that transforms audio into a visual format, such as a spectrogram, to facilitate the use of convolutional neural networks, which are prevalent in audio machine learning applications. The different recording methods and signal processing methods will be explained in Chapter 4. A specific

experimental setup that records a dataset using specialized methods will be defined for every experiment.

The structure of this thesis is split into multiple chapters. Chapter 2 will explore the literature surrounding the side-channel attacks. This chapter will define research gaps and relevance in the academic ecosystem. Next, a chapter regarding background information will be detailed in Chapter 3. It will serve as a basis for knowledge of audio and machine learning. Chapter 4 will detail the research methodology for the experiments. The next three chapters, Chapter 5 to 7, will serve as sections answering the three research questions posed in this introduction. Finally, Chapter 8 will conclude the thesis and open the discussion to future works.

2

Literature Review

In today's cybersecurity landscape, sensitive information such as mouse movements and keyboard outputs are considered privileged data sources. Their ability to facilitate the detailed retracing of cyber-activities is invaluable to cyber-criminals and various data collection entities, including advertising firms and nation-states. Due to their ability to provide a near-perfect recreation of a user's actions, these data sources can be used to derive valuable insights such as a user's interests, mouse stopping points on a web page, or even the clandestine cyber-movements of dissidents. Since a mouse and keyboard are mechanical devices and are commonly used as interfaces for human-computer interaction, they invariably produce slight acoustic traces. These unique sounds, whether they originate from keyboard strokes or mouse actions, are not just random noises. They can provide information on activities, such as mouse clicking patterns, which can be collected and analyzed. These acoustic traces present an attractive opportunity for motivated attackers. Through acoustic side-channel analysis, attackers can utilize these environmental variables and extrapolate sensitive information. This paper focuses on the under-explored area of computer mouse audio as a security issue within acoustic side-channel attacks. More specifically, using audio recordings to understand mouse movements can provide critical information about a user's "virtual" movements, possibly indicating a button pressed, a dragged object, or the opening of new windows. While each individual action may have a low security impact, the collective analysis of this data can create a comprehensive picture of the user's activities. An attacker, for instance, could deduce from mouse movements whether a user opened a new browser window, clicked 'yes' or 'no', or even hovered over specific sections of the screen such as ads.

This literature review aims to provide an overview of the current state of research in the field of acoustic side-channel attacks, focusing on the methods used to infer user activities and the techniques developed to enhance the accuracy of these inferences. Acoustic side-channel attacks typically involve recognizing basic keyboard and mouse events to predict high-level computer-usage activities. The Time Difference of Arrival (TDoA) technique is a prevalent method used in these attacks. This method uses the time difference of sound arrival at two different points, usually two microphones on a device, to infer the location of the sound source. This technique has been effectively used to determine the keys pressed on a touchscreen as described by [9]. The paper "Context-free Attacks Using Keyboard Acoustic Emanations" introduces a context-free and geometry-based method to recover keystrokes, [29]. This approach uses smartphones to record acoustic emanations from keystrokes and estimates the physical positions of the keystrokes based on the TDoA method. The paper highlights that modern smartphones, equipped with more than one microphone, can be used effectively for this purpose. Interestingly, some studies have explored the feasibility of keystroke snooping that does not require training or linguistic models. These context-free attacks discriminate keystrokes based on the TDoA of the keystroke sound at the two phone microphones and refine such estimates using acoustic differences in the sound emitted by each key. For instance, [13] can locate the origin of keystrokes using a single phone placed behind the keyboard without needing labelled training data or linguistic context. The system uses the TDoA of keystroke sounds at the phone's two microphones and refines these estimates using the acoustic

differences in the sounds emitted by each key. The method, tested on three types of keyboards and off-the-shelf smartphones, can recover 94% of keystrokes, making it the first single-device technique that enables acoustic snooping of passwords.

Another technique that has been explored is using the Mel-Frequency Cepstral Coefficient (MFCC) to identify keystrokes. The paper, described by [6], introduces PoKeMon, a new keystroke monitoring method for smartphones that does not need to know the phone-keyboard position in advance. PoKeMon combines linguistic and TDoA approaches to identify “pilot keys" that help estimate the phone-keyboard position. A calibration scheme using accelerometer sensors and selective keys improves accuracy. Keystrokes are then identified using an MFCC-based k-means clustering approach. While acoustic side-channel attacks often wait and listen to the audio emanations, some researchers have taken this concept further by transforming devices into active sonar systems [4]. These systems emit inaudible sound signals and track the echoes at the device’s microphones. This innovative approach has been used for fine-grained finger tracking and keystroke inference, further expanding the potential of acoustic side-channel attacks.

Acoustic side-channel attacks (SCAs) are a fast-growing area of study, with methods that are becoming more complex using advanced signal processing and machine learning to pinpoint user actions. As we rely more on tech, it is crucial to understand and prevent such attacks. This review aims to explore the current research in this field thoroughly, pinpointing key methods, techniques, and potential future study areas. Most of the papers in this review will center on keystroke inference since it is the closest existing research due to the research gap in inferring mouse movements. We will start by discussing general acoustic SCAs, then move on to passive ones, and finally focus on active SCAs targeting HIDs.

2.1. Acoustic Side-Channel Attacks

In this section, we explore acoustic side-channel attacks (SCAs) by examining two significant papers, namely Synesthesia [7] and Behavicker [3]. The former paper provides an insightful method for exposing a computer’s screen content through auditory leakage. The latter paper provides a comprehensive approach to a SCA that passively eavesdrops on both keyboard and mouse, thus leaking user information. Furthermore, it is the only one that leverages acoustic data from a computer mouse to infer user activity. The Behavicker paper, however, only uses mouse clicking and scrolling sounds. This novel research effort is noteworthy as it is closest to the current research performed by this thesis. We will analyze both papers’ content and extract information that could contribute to our research objectives.

The paper titled “Synesthesia: Detecting Screen Content via Remote Acoustic Side Channels" by Genkin et al. (2019) examines the concept of acoustic SCAs and their potential for leaking private screen content through audio leakage [7]. The researchers found that LCD screens emit content-dependent audio signals, regardless of age, model, or backlighting type, which can be captured through nearby microphones used in VoIP calls or video conferences. The study labelled this phenomenon ‘synesthesia’ - the “hearing" of on-screen images. They conducted extensive experiments using both built-in and webcam microphones, which revealed acoustic leakages through audio recordings and Google Hangouts. The study had three main objectives: to extract text displayed on the screen, distinguish between websites displayed on the screen, and extract text entered via Ubuntu’s on-screen keyboard. The authors developed an experimental setup to characterize the signal emitted from different screen patterns, which helped to understand the acoustic leakage in screens. They determined that the power consumption of the monitor’s circuits changes depending on the screen content, causing internal components to vibrate and emit sounds. They tested this theory by placing microphones within computer screens and “listening" to various components. In addition, they found that the leakage was indeed acoustic, not electromagnetic, as by physically blocking the microphone, the leakage was not detected. To comprehend the leakage frequency and its relationship with the displayed image, they observed a clear correlation between the acoustic signal and the image displayed on the screen, specifically in the frequency ranges of 3-4 KHz and 27.5-38 KHz. They displayed zebra-like patterns and observed correlations as they changed the images.

The authors identified several potential attack vectors, including close-range and at-distance attacks, phone attacks, and even attacks via virtual assistants. One of their goals was to determine the key presses on a virtual keyboard. For their analysis, the authors leveraged machine learning, especially Convolutional Neural Networks (CNNs), to infer complex dependencies in time series data amidst noise. CNNs were especially potent when configured correctly and backed by sufficient data. They iteratively simulated the acoustic leakage during hundreds of key presses on the on-screen keyboard, which led to a 100% accuracy rate in detecting key presses for both the smartphone and close-range attack. In their attempt at text extraction of “black-on-white text in very large monospace font, “they achieved an accuracy rate between 88% to 98% for most individual characters [7]. When extracting words from the screen, the correct word was identified in the top five most probable words in 72% of the cases. Regarding website distinguishing, their task was to identify one among 97 displayed websites. They achieved a validation set accuracy of 97.09% in close-range settings and slightly lower accuracies of 91.20% and 90.9% in smartphone and at-distance test sets, respectively.

The study presented a novel physical side-channel leak, which unveiled acoustic leakage of screen content. This leak was discovered exploitable even by weak attackers who can only access encoded audio traces from legitimate communication channels. Their research demonstrated highly precise content extraction and identification attacks, revealing the vulnerability of users during seemingly innocent activities, such as using video calls.

The paper “Behavicker: Eavesdropping Computer-Usage Activities through Acoustic Side Channel," written by a team from Shenzhen University, describes a system called Behavicker, which determines user activities through keyboard and mouse clicking and scrolling events [3]. The sounds generated from keyboard typing or mouse clicking can leak essential insights into a user’s behavior. Different actions, such as browsing a news website or playing a video game, induce varying keyboard and mouse usage patterns, each with their unique acoustic footprint. Similarly, distinct keyboard keys produce different sounds, which can help differentiate the specific keys pressed. This can expose sensitive information and highlight behavioral patterns, aiding the orchestration of highly personalized social engineering attacks.

Behavicker is an acoustic side-channel eavesdropping system designed to decipher common computer usage activities like texting, gaming, and web browsing. It operates by recognizing high-level computer-usage activities from patterns in keystroke, mouse clicking, and scrolling. These events are discerned based on their unique acoustic properties because of the mechanical design of keyboards and mice. In its test results, Behavicker demonstrated the ability to identify six basic interaction events with an accuracy of 88.3% and differentiate between seven computer-usage activities with an accuracy of 82.7%. This was achieved by studying the temporal distribution of basic interaction events without access to the victim’s training data.

Behavicker’s attack scenario is an attacker-controlled smartphone placed near the victim’s desktop, recording audio and transmitting it to a server. The system utilizes two functional modules: Acoustic-based Interaction Event Recognition and Computer-Usage Recognition. The first leverages signal processing and machine learning to recognize interaction events, while the second employs hierarchical classifiers via time-series analysis to distinguish between computer usage activities.

In the Acoustic-based Interaction Event Recognition module, audio preprocessing techniques filter out band noise and improve the signal-to-noise ratio—a method known as constant false alarm rate handles event detection and segmentation. A hierarchical learning scheme helps classify interaction events, first into broad categories based on similar signal patterns and then into individual events within a group. To combat the issue of not having access to labelled training data, Behavicker employs an iterative model adaptation scheme, a technique akin to incremental learning in machine learning. By feeding back samples of the target, labelled by the classification procedure, into the original training data set, this

approach helps improve the accuracy of interaction event recognition.

The Computer-Usage Recognition module employs a tree-structured classifier to discern different computer-usage activities from keyboard and mouse interaction events. The classifier can effectively distinguish different activities using factors like Keyboard-Mouse Ratio, Event Switch Rate, Mean Duration of Actions, Gap Between Events, Left Click Frequency, Keystroke Frequency, and Scrolling Up Rate. The thresholds used are user-independent and are determined using the attacker's samples.

In the paper, the authors collected data from 20 participants using different keyboard-mouse combinations in various office environments. The performance of Behavicker was evaluated in recognizing different interaction events and various distances from the keyboard. The accuracy in recognizing different activities was an average of 82.7%.

The Behavicker system integrates the acoustic emissions from the keyboard and mouse to decipher user activities. It is a pioneering work that uses the sound produced by mouse interactions, such as clicks and scrolls, to extract information. This paper aligns directly with our theme of inferring mouse movement, thus making it of high relevance. Additionally, it introduces an intriguing concept of hierarchical learning. This is particularly significant given that a computer mouse is only sometimes in motion. Thus, the implementation of layered learning is necessary first to identify when the mouse is in motion and subsequently infer the context of mouse usage. This methodological approach significantly enriches the existing body of knowledge and serves as a valuable resource for this literature review.

2.2. Passive SCAs on Human Interface Devices

In this segment, our attention turns to passive SCAs on HIDs. Passive SCAs operate by interpreting data derived from the audio released by the system under test. We will first delve into a study that investigates eavesdropping on keyboards during video calls. We will then examine a paper that investigates identifying PIN pads from ATMs through the sounds they emit. This thematic focus allows us to deepen our understanding of the various applications and implications of passive SCAs on HIDs.

The study "Skype & Type: Keyboard Eavesdropping in Voice-over-IP" by Cecconello et al. presents a novel cybersecurity threat, which capitalizes on the fact that Voice-over-IP (VoIP) software like Skype captures and transmits all acoustic emanations, including keyboard sounds [2]. This transmission can reveal sensitive information, such as passwords, that individuals might type during a call. The "Skype & Type" (S&T) attack proposes a relatively weak adversary model that requires no physical proximity to the victim nor precise profiling of their typing style or keyboard. The model can work with minimal leaked keystrokes, making it feasible in many real-world settings. The researchers show that the S&T attack can effectively reconstruct the victim's input based on the audio information transmitted via Skype. With some knowledge of the victim's typing style and keyboard model, an attacker can achieve a top-5 accuracy of 91.7% in guessing a random key pressed by the victim. The findings demonstrate that this attack is effective with various recording devices, diverse typing styles and speeds, and poses a significant threat when the victim is typing in a known language.

VoIP software inherently collects and transmits all acoustic emanations, such as keyboard sounds, which paves the way for potential exploitation by attackers. The methodology of the S&T attack comprises four distinct phases: Data collection, Feature extraction, Model training, and the Attack phase. In the data collection phase, the attacker captures keystroke sounds inadvertently transmitted by the victim during a VoIP call. During feature extraction, the recorded audio is preprocessed by segmenting the waveforms into smaller parts, each containing the sound of a single keystroke. This process uses segmentation based on distinct keystroke sound peaks, slicing to isolate 100ms segments from the first event, and feature extraction using MFCC. The researchers found that MFCC offered superior accuracy over other spectral features. The model training phase follows where the attacker employs the collected labelled data (keystroke sounds and corresponding keys) to train a supervised learning model. Different classifiers were tested, including Logistic Regression and SVM, both showing around

90% top-1 accuracy and over 98.9% top-5 accuracy. In the attack phase, the attacker applies the trained model to predict the keys corresponding to the keystroke sounds collected during the VoIP call. For instance, the victim might type their password during the call, which the attacker can potentially decipher.

The study collected data from 12 unique participants and 14 keyboards of 11 different models, resulting in 156 unique datasets. The efficacy of the S&T attack was evaluated in a stratified 10-fold cross-validation scheme, demonstrating the viability of the attack across different scenarios and devices. Furthermore, the S&T attack was highly successful for word recognition and password cracking. The accuracy escalated with word length, reaching 100% for 10-character-long words on MacBook Pro and Toshiba laptops and 15-character words on Lenovo laptops. The study showed a novel VoIP-based remote keyboard acoustic eavesdropping attack with practical applications in deciphering random and non-random text.

The paper, “We Can Hear Your PIN Drop: An Acoustic Side-Channel Attack on ATM PIN Pads" by Balagani et al., presents a novel acoustic side-channel attack on ATM PIN entry, termed the PinDrop attack [1]. The attack involves two steps: first, an acoustic profile is created for each key on the target PIN pad. Secondly, the attacker records audio emitted by each pressed key during PIN entry and compares these recordings to the acoustic profiles to identify the keys pressed. These steps can be performed in any order.

The initial stage in the feature extraction process outlined in the paper involves segmentation, which entails isolating distinct keystroke sounds within a recording. The technique focuses on identifying two unique peaks within a keystroke sound waveform related to the key press and release events. The isolation of keystrokes is automated, followed by normalizing the signal’s amplitude. Subsequently, Fast Fourier Transform (FFT) coefficients are added together to derive the energy for each window. Since the keystroke sounds tend to be louder than the background noise, “events" are defined as those instances when the waveform’s intensity exceeds the 90th percentile. Post-segmentation, the waveform is divided into sound samples of 100 milliseconds duration, starting from the first identified event. The subsequent step involves feature extraction, where the MFCC is computed to convert the information in the sound samples into actionable data points. The paper uses a logistic regression classifier to categorize these data points. This classifier was tested on a dataset comprising ten samples for each of the 26 English alphabet keys using a 10-fold cross-validation scheme. The accuracy of the classifier was assessed based on different spectral features, including FFT coefficients, cepstral coefficients, and MFCC. The process was replicated with data collected from five users on a MacBook Pro laptop, leading to the following accuracy results: MFCC yielded the best accuracy at 90.61%, followed by FFT coefficients at 86.30%, and cepstral coefficients trailing at 51%. The paper presents PinDrop as an efficient acoustic side-channel attack method on ATM PIN pads. The utilized techniques underscore that MFCC delivers superior results. Consequently, this will be considered a significant factor in our research endeavors.

2.3. Active SCAs on Human Interface Devices

Recent advancements in acoustic SCAs involve transforming smartphones into active sonar systems to detect user interactions. This intriguing possibility presents a ‘sonar-style’ system where we can calculate the distance of a hand’s interaction with a HID, consequently determining the HID’s specific position. To delve into this potential avenue for SCAs, we will first examine a paper that leverages active sonar to pinpoint a finger’s location on a device. While it does not present a direct SCA application, it lays out the methodology that enables us to comprehend the practical implementation of this technique. The subsequent paper will offer the first instance of an active acoustic SCA designed to identify user interactions with a smartphone touchscreen.

In the paper “FingerIO: Using Active Sonar for Fine-Grained Finger Tracking", the authors propose an innovative approach to finger tracking by transforming a smartphone device into an active sonar system [14]. The system transmits inaudible sound signals in the 18-20 kHz range and tracks the echoes of the finger at its microphones. This approach is unique as it does not require sensors on the finger and can work even when obstructions are present. The authors address the challenge of noisy echo signals

and the lack of synchronization between speakers and microphones on mobile devices by employing a modulation technique called Orthogonal Frequency Division Multiplexing (OFDM). This technique, commonly used in modern wireless systems, is designed to estimate the sampling offset for every transmission, thereby enabling the decoding of transmitted information. The proposed approach was implemented on a Samsung Galaxy S4 smartphone and a smartwatch prototype. The experimental evaluations demonstrated high tracking accuracies, even when the device was occluded from the finger. The system achieved 2-D finger tracking accuracies of 8mm and 1.2cm at 169 frames/s for the smartphone and smartwatch prototypes, respectively. This study represents a significant advancement in the field of finger tracking technology, demonstrating the potential of using active sonar systems for this purpose. The high tracking accuracy achieved by the system, even in the presence of obstructions, underscores its robustness and applicability in real-world scenarios. In addition, this research opened the door to using this technology for SCA, which we will explore in the following paper.

In the paper “SonarSnoop: Active Acoustic Side-Channel Attacks," the authors Cheng, Bagci, Roedig, and Yan introduce a novel security threat in the form of an active acoustic side-channel attack [4]. This attack leverages a device’s speakers to emit inaudible acoustic signals, which are then reflected off the user’s fingers and recorded via the smartphone’s microphones. This process effectively transforms the acoustic system of a smartphone into a sonar system. Similarly to the previous paper, the authors use OFDM sound signals emitted via the device’s speakers. The microphones on the device are then used to record the echo of this signal. Analyzing the recorded echo makes it possible to deduce user interaction patterns with the touch screen. The authors illustrate the potential of this novel active acoustic side-channel attack by using the task of stealing Android unlock patterns as a case study. They demonstrate that the number of unlock patterns an attacker must try until a successful authentication can be reduced by up to 70% using the acoustic side-channel, which is entirely unnoticeable to a victim. The authors also discuss the generalizability of their approach. Although their experiments are done with a phone, their method applies to many other devices and environments where microphones and speakers are available. This highlights a new family of security threats that need to be addressed in cybersecurity.

2.4. Research Gaps

The literature surveyed thus far offers insights into acoustic side-channel attacks, revealing various techniques that enable the leakage of sensitive data from various HIDs, including screens, keyboards, and mice. A close examination of different feature selection methodologies, such as segmentation and MFCC, was conducted, and these proved beneficial in deriving research directions. Most inferences were performed through machine or deep learning models, which indicates how the research will be oriented. However, it is crucial to recognize the existence of research gaps, with the most notable omission being the exploration of mouse movements. The closest we get to the study of acoustic emanations from mouse usage is in the Behavicker paper, where scrolling and clicking sounds were used to create a profile of user activity. This highlights a research gap in the inference of mouse movements, specifically the detection of such movements through associated audio signals. Mouse movements can leak different actions, such as closing a computer window, or more nuanced tracking, such as pinpointing mouse positioning.

Our review of existing research and background studies reveals a significant gap: the absence of a leakage model for computer mice in contemporary research. While there has been extensive analysis of keyboards and touchscreens, with various studies demonstrating differing levels of precision in replicating user input, the domain of computer mouse movements still needs to be explored. This oversight is notable given that there are two primary input mechanisms in desktop computing – the keyboard for textual input and the mouse for navigating the pointer. This raises a pertinent question: What about the mouse? As a crucial component of desktop interaction, the mouse warrants similar scrutiny. It is essential to understand the potential for information leakage from mouse movements and clicks, as it completes the security of side-channel attacks. Thus, our research aims to bridge this gap, exploring the feasibility and implications of a leakage model for mouse interactions.

3

Background

The focus of this thesis is acoustic side-channel attacks. Therefore, it is important to take some time to learn or review our knowledge of acoustic signals or audio. In the first section, we will review the basics of sound and the terms we can use to describe them. In the second section, this thesis focused on a machine learning approach to the side-channel attack. We will also dive deeper into the representation of acoustic signals for machine learning.

3.1. Fundamentals of Sound Perception

Sound is a key mechanism for humans to interpret the world by detecting air vibrations. Hearing is a signalling function that allows us to derive information on events that we cannot see, according to [8]. In nature, this perceptive lens of the world can inform us of the presence of water when near a stream, for example. Moreover, it signals dangers; for example, a cracking sound from a piece of wood would indicate something falling from a nearby tree. This perception of sound originates from the vibration of objects and its effect on the medium around it. A loudspeaker produces sound by vibrating a membrane, thus compressing the air around it and creating patterns of high and low-pressure regions, which is described as the sound wave. This wave travels at 340 meters per second in the air, which is then collected by the recording device. Waves have two key properties: amplitude and frequency. The amplitude of a sound wave can be described, according to [8], as the “difference in pressure between the high and low peaks of the sound wave." Meanwhile, the frequency is the number of times the sound cycle or period is repeated per second. On the one hand, the frequency metric is measured in Hertz (Hz), with the human audible frequencies ranging from 20 to 20,000 Hz [5]. On the other hand, as amplitude can represent many different ranges of pressures, it is often represented using decibels to be at a more manageable scale. The sound pressure to decibel formula is the following:

$$\mathrm{dB} = 10 \cdot \log_{10}\left(\frac{P}{P_0}\right) \tag{3.1}$$

“where dB stands for decibels, P is the sound pressure of the stimulus, and P_0 is a standard sound pressure that is usually set at 20 micropascals, an extremely small pressure", according to [8].

Pure sounds consist of a single sine wave. However, it is rare to find pure sound in the natural environment. Complex sounds are a concoction of multiple base sine waves. To extract the different frequencies, it is possible to apply Fourier analysis, which deconstructs complex sound into its sine-wave components. According to [8], the “lowest frequency is the tone’s fundamental frequency (or the first harmonic), and the other sine waves are the tone’s harmonics."

A louder sound is achieved by increasing the sound pressure without modifying its frequency. Thus, according to [18], loudness is characterized as “the magnitude of auditory sensation." Sounds with the

same loudness and note can still be differentiated if they originate from different sources due to the sound's timbre. Thus, sound has additional qualities independent of the pitch and loudness, which qualify the timbre. For instance, the decay of the sound or relative strengths of the harmonics can indicate the type of instrument used.

The texture of sound depends on the environment in which the sound is perceived as much as how the sound is produced. For example, the sound is perceived differently based on what happens to the sound during its travel time from the source to the sink. The sound reaching the ear or recording directly is called direct sound, while sounds taking a more complicated path, such as bouncing off surfaces, are called indirect sound. The room's reverberation time is the amount of indirect sound produced by the room, according to [8]. In this research, the mix between direct and indirect sound will be used for analysis.

For this project, sound localization is crucial to determine the mouse's position. For human perception, the interaural differences or differences in sound heard by the left and right ears provide cues about sound localization, according to [8]. The interaural time difference and the interaural intensity difference are the major contributing cues to this localization. As sound is produced, it is heard by each ear (or microphone) at different moments, called the interaural time difference. As sound sources originate from different positions, the acoustic signals will take different paths to reach the ear. For instance, a source on the side will go through a different route than a source that originates directly in front. These differences allow some sense of differentiation from the audio sources. Moreover, the contrast in intensity or interaural intensity difference is also an important binaural cue which allows localization information on the left-right positions of a human head. If we imagine a head, the acoustic shadow corresponds to an area where the high frequencies are not present due to being affected by the head acting as a barrier. According to [8], the differences in both time and intensity provide information on the localization in sound, a key aspect of this thesis.

3.2. Machine Learning, CNNs and Metrics

Machine or deep learning (ML) will be a large part of the research performed in this thesis. This technique allows us to learn patterns by extracting insights and patterns in a large amount of data. For instance, it is possible to differentiate between bird songs using machine learning algorithms [20]. In our situation, the goal is to identify the audio emitted by basic mouse movements, giving importance to machine learning. ML involves the process of gathering a dataset consisting of input values and corresponding output values, which is then bifurcated into two distinct subsets: a training dataset and a testing dataset [28]. The training dataset serves as the primary resource for the machine learning algorithm, enabling it to understand the patterns within the data. Concurrently, the algorithm's internal parameters are fine-tuned based on this training data. In contrast, the testing dataset plays a role in the evaluation phase. It is utilized to assess the performance and reliability of the developed machine learning model. To ensure the integrity of this evaluation, it is necessary that the model is not exposed to the testing data during its learning phase. This segregation guarantees that the model's effectiveness is measured accurately. In addition, it shows its capacity to generalize from the training data to unseen examples.

Machine learning encompasses various approaches, notably supervised and unsupervised learning [10]. Unsupervised learning operates autonomously by training with unlabeled datasets, where the algorithm discerns structure or clusters within the data. Conversely, supervised learning relies on labelled datasets, where each input data is paired with a corresponding output label. In this context, the algorithm learns to map inputs to outputs, improving its accuracy with each example. This thesis predominantly focuses on supervised machine learning in the context of learning mouse movement data. Specific movements or actions are distinctly labelled, providing a setup for the supervised learning algorithm to learn and predict based on these pre-defined categories.

In supervised machine learning, this thesis primarily explores two fundamental branches: regression

and classification [22]. Classification-oriented machine learning models specialize in categorizing input values into distinct, discrete output classes [11]. An example is binary classification, where inputs are separated into two categories, often denoted as 'True' or 'False'. Conversely, regression models are employed when the objective is to predict continuous values. Among the various regression techniques, linear regression is the simplest [22] as it establishes a linear relationship between input variables and continuous output values, thereby predicting outcomes that follow a linear trajectory.

In this research, the input data to the ML models is sound. Audio signals are one-dimensional time series data, delineating variations in amplitude over time, as depicted in a waveform. While it is feasible to use this waveform directly as input for machine learning algorithms, it is also possible to transform the audio into an image to run a more complicated deep learning model, according to [17]. These transformations are typically realized through spectrograms or Mel Frequency Cepstral Coefficients (MFCCs) [27]. A spectrogram provides a two-dimensional visual representation of an audio signal, where the three axes represent the time, frequency, and intensity (or amplitude) of frequencies. A Fourier transform, giving amplitude and phase spectrum, is computed over audio segments, giving the time domain [21]. This allows us to identify and analyze various temporal and spectral characteristics in the audio data, making it particularly useful for machine or deep learning models [19]. MFCCs, on the other hand, are computed through a series of steps involving the application of the Mel scale to the power spectrum, followed by a logarithmic transformation and then a discrete cosine transform [24]. MFCCs "mimics the human auditory system very closely" [24] and have therefore been used in many applications as evoked in Chapter 2.

Both spectrograms and MFCCs transform audio signals into more informative formats suitable for deep learning applications such as Convolutional Neural Networks (CNNs). CNNs will be the center of focus due to their application in audio classification tasks. A CNN's architecture is often split into three main focuses [16]: convolutional, pooling, and fully connected layers. The convolutional layer applies a filter kernel on the 2D input vector to extract a feature map representing the information in different data points. The pooling layer is used to reduce the "dimensionality of the representation"[16], reducing the learning parameters of the model. The final fully connected layer is a more traditional artificial neural network, a set of fully connected neurons that progressively learn a state of parameters together. The CNN architecture can evolve by adding additional layers that serve different learning purposes, such as doubling the convolutional and pooling layers [16].

Several metrics will be utilized throughout this research to evaluate the performance of machine learning models. The most important metrics will be precision, recall, accuracy, and the F1-score, which we will define according to [15]. First, the precision metric defines the ratio of true positives out of all predicted positives. The recall metric is the ratio of true positives predicted out of all actual positives. The accuracy evaluates the proportion of true predictions on the total number of samples. While the F1-score is the "harmonic mean of the precision and recall" [15], it thus finds the balance between precision and recall. These metrics allow to evaluate whether a machine learning model is overfitting, a common issue that must be avoided. According to [26], overfitting is a phenomenon that happens when the model "does not generalize well from observed data to unseen data." This means that the often positive performance metrics on the training dataset cannot be reproduced on the testing dataset, as often, the model learns the noise and anomalies rather than a generalizable structure of the data—this lack of generalization results in an unusable model, failing on unseen or real-world data points. There exist multiple issues why overfitting could happen; however, none of them can be entirely proved; thus, resolving the issue is often a case of testing different techniques, such as more training data, regularization or network-reduction [26].

4

Methodology

This section explores the methodologies employed in our research, diving into the correlation between mouse movements and the audio generated by that computer mouse. The use of a computer mouse varies depending on the user and the surrounding environment since gaming or scrolling the web generates different sounds due to the various movements. Our experiments focused on two key data points for our datasets: the audio data and the mouse movement data. The crux of our investigation lies in examining the correlation between these data points. The movement data points capture the mouse's positional coordinates at specific intervals. This data will be further processed to trace the trajectory of mouse movements by obtaining direction vectors. Conversely, the audio data are the acoustic emissions recorded simultaneously with the mouse's movements.

This research comprises various experiments designed to address specific research questions posed in Chapter 1. The initial phase of our research involved exploratory investigations to comprehend the acoustic leakage model associated with computer mice. These preliminary experiments focused on understanding and analyzing basic mouse movements, such as directional patterns. In this context, the primary goal was to predict the linear, back-and-forth movements of the mouse along cardinal axes on a mouse pad. As we progressed, the second phase of our research delved into refining the resolution of our predictive models. Here, the experiments were geared towards enhancing prediction granularity. This involved moving beyond linear predictions to inferring directional vectors or movement angles. In the final phase, the research aimed to validate these models in real-world scenarios by recording and analyzing mouse movements in more complex and realistic settings, such as user activities on a laptop. By doing so, we sought to assess the practical applicability and effectiveness of our predictive models in everyday computing environments.

Given the thesis' nature, the scope and boundaries of the study were dynamically shaped as the research progressed. This approach allowed for a flexible exploration of various possibilities within mouse movement SCAs. Initially, not all mouse movements under investigation were assumed to reflect typical user behavior. For example, while it might seem improbable for a user to move their mouse repeatedly along a single axis, such scenarios cannot be entirely dismissed. Determining the SCA acoustic leakage model was an important part of the research; thus, while the plausibility of mouse movements was initially ignored, it would be analyzed through various experiments later. This was done by envisioning scenarios to place the work into a real-world context.

All machine learning models were trained on one laptop, a Dell Inspiron 14 Plus 7420, with the following specifications:

- CPU: 12th Gen Intel(R) Core(TM) i7-12700H 2.30 GHz
- GPU: NVIDIA GeForce RTX 3050

- RAM: 16GB

4.1. Types of Mouse Pad

In this study, the 'mouse pad' is the mouse movement area. This term encompasses the physical space within which the mouse operates. There are two types of mouse pads: fixed and flexible. A fixed mouse pad implies a confined space with predetermined dimensions, representing a scenario where users operate within a set boundary. Conversely, a flexible mouse pad denotes more unrestricted usage, allowing for broader movement patterns. We will employ various settings throughout our experiments, each with a clearly defined mouse pad area.

In this study, we will sometimes utilize a coordinate system on the mouse pad to categorize mouse movements. Directional movements are defined relative to specific points on the pad: 'T' for top, 'B' for bottom, 'L' for left, and 'R' for right. Additionally, 'M' represents the middle along the y-axis, while 'C' indicates the central point along the x-axis. Therefore, a movement labelled 'TL to BR' signifies a diagonal trajectory from the top-left to the bottom-right corner of the pad. In some cases, simpler designations based on cardinal directions—north (N), south (S), east (E), and west (W)—will also be employed. In this context, a north-to-south movement corresponds to 'TC to BC', and an east-to-west movement is represented as 'ML to MR'.

4.2. Audio Recording Methods

Each experiment within this study will use different recording methods aligned with the testing objectives. For example, in addressing the initial research question regarding the existence of an acoustic leakage model, our initial measurements will start and end recording at the start and end of the mouse movements. This approach allows the recording of mouse movement acoustic signals without any noise. In subsequent phases, some experiments will continuously record sound, even when the mouse is not moving. Subsequent analysis will then split the sound to find the audio corresponding to mouse movements. A more complex recording environment aims to provide a more representative setting, as an attacker will never have the perfect context to analyze audio.

Key parameters of audio recordings, such as sample rate, bit depth, channels, and file format, play a crucial role in recorded data. The sample rate, typically noted in kilohertz (kHz), defines the number of samples of audio captured per second, where higher rates allow for a more accurate representation of the original sound [5]. Moreover, according to [5], bit depth determines the resolution of the amplitude of each audio sample. This can influence the noise level of the recording. The number of channels affects the spatial representation of the sound: mono contains only one waveform, while stereo contains two. Finally, the file format, be it WAV, MP3, FLAC, or another, dictates the encoding method and can affect both the quality and size of the audio file.

During our experimental analysis, we will use the WAV file format due to being uncompressed, ensuring high fidelity in our recordings [5]. We will record in mono for experiments utilizing a single microphone setup as the sound comes from a single source. We will switch to a stereo configuration in scenarios where a second microphone is employed. The bit depth selected for each experiment varies around 16 to 32 bits. The sample rate will generally range from 44.1 kHz to 48 kHz in the smartphone range. This choice is driven by our threat model, which envisions an attacker using a smartphone placed in proximity to the victim for information gathering.

Throughout the experiments, the placement of the microphone changed for testing scenarios, the specifics of which will be explained in the following sections. For example, positioning a microphone adjacent to the mouse pad enhances audio precision along that particular axis. Furthermore, with the two microphones from a smartphone, it is possible to experiment with different locations for varying accuracy of audio capture. This diversity in microphone placement served as an assessment of the predictive capabilities under varying parameters.

The experiments' initial phase involved synchronizing the audio recording, starting with the start of mouse movement. This was achieved using software designed to simultaneously activate two timers, ensuring one-to-one correspondence between each mouse movement and its accompanying audio signal. In contrast, subsequent experiments adopted a continuous audio recording strategy. In this setup, audio frames were constantly captured in the background, independent of the mouse movement. This method did not assure a perfect synchronization between movement and audio frames but was more realistic as it captured sounds without any outside triggers. These varied experimental designs helped answer the different research questions, particularly focusing on the capabilities and precision of the system under different conditions.

In most of our experimental setups, the microphone is directly connected to the computer. This configuration facilitates the synchronization of mouse position and movement data by initiating the recording process and activating the mouse logger simultaneously. To record with a second microphone, using a smartphone, we used the Android application 'AudioRec' (available on the Google PlayStore). This adds a layer of complexity to synchronize the mouse movements with the stereo audio recordings. A clear left mouse click generates a distinct sound, an easily detectable audio marker. This marker is then used as the reference point for synchronization. The recording script on the laptop is designed to capture all mouse movements along with their respective timestamps. It also records instances of mouse clicks and their timestamp. This facilitates the establishment of time windows aligned with the timestamps from the mouse click, thus synchronizing audio and mouse movement data.

4.3. Detailed Analysis of Audio-Mouse Synchronization Script

The script's core focuses on processing an audio file to detect mouse click events, which are then used for synchronization with mouse movement data. The following steps outline this process:

1. **Loading the Audio File**: The audio file containing the mouse click sounds is loaded in a Jupyter notebook using `TorchAudio`:

```
file = "true_click.wav"
waveform, sample_rate = torchaudio.load(file)
```

2. **Audio Processing for Click Detection**: The script proceeds to process this audio data to isolate the click sounds.
   - *Stereo to Mono Conversion*: If the loaded audio is in stereo format, it is converted to mono by averaging the two channels to process the data uniformly:

```
waveform = torch.mean(waveform, dim=0, keepdim=True)
```

   - *Absolute Values Calculation*: The next step involves calculating the absolute value of the waveform to highlight fluctuations in the audio, which is useful for extracting mouse clicks:

```
abs_wave = torch.abs(waveform)
```

   - *Threshold Determination for Peak Detection*: To detect the peaks corresponding to clicks, a threshold is set. This threshold is determined using a high quantile of the absolute values, ensuring that only the most prominent peaks are considered:

```
threshold = torch.quantile(abs_wave, 0.9998)
```

   - *Identifying Mouse Click Events*: The script then identifies points in the absolute values that exceed the threshold, marking these as potential click events:

```
click_samples = torch.where(abs_wave > threshold)[1]
```

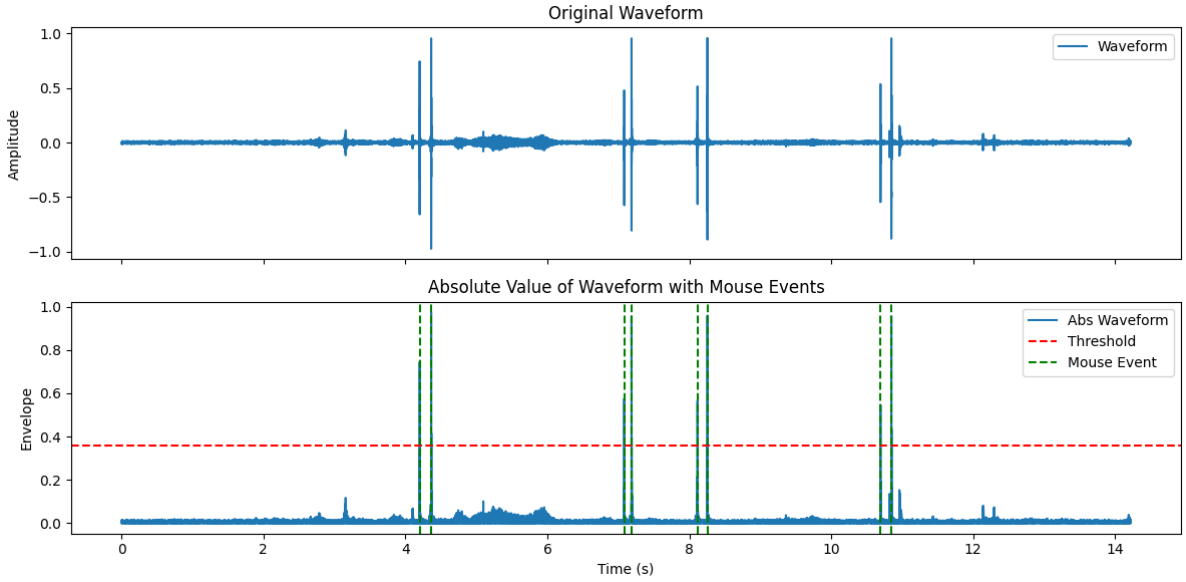


Figure 4.1: Waveform Analysis for Click Detection

Figure 4.1 illustrates the waveform analysis for detecting mouse click events. The top subplot presents the original audio waveform, displaying the amplitude variations over time. Twin peaks in the waveform represent click events. The bottom subplot shows the processed waveform. The green dashed lines mark the instances where the waveform exceeds the calculated threshold, indicated by the red dashed line. These peaks are interpreted as detected clicks. The synchronization algorithm then groups the nearest peaks in the audio waveform as mouse clicks will produce two sounds for the mouse press and release. A maximum peak distance is calculated from the audio file's sample rate to group peaks together, allowing for a time-based threshold (e.g., 0.2 seconds).

```
max_distance_samples = int(sample_rate * 0.2)
grouped_clicks = []
current_group = []

for i, peak in enumerate(valid_peaks):
    if not current_group:
        current_group.append(peak)
    else:
        if peak - current_group[-1] < max_distance_samples:
            current_group.append(peak)
            if len(current_group) == 2:
                grouped_clicks.append(current_group)
                current_group = []
        else:
            current_group = [peak]
```

Listing 4.1: Source code to group mouse press and release

Upon identifying the groups of peaks representing clicks, the algorithm calculates the audio start timestamp, as the mouse clicks are the only data points to have an associated timestamp. This timestamp is important for synchronization as it establishes the offset between the audio recording and the mouse click events. It is then possible to identify an approximate starting timestamp for the audio data, thus aligning the two streams of data:

```
first_mouse_click_timestamp = mouse_click_pairs[0][0]
first_audio_click_sample_index = grouped_clicks[0][0].item()

audio_start_timestamp = first_mouse_click_timestamp -
                        (first_audio_click_sample_index / sample_rate)
```

Listing 4.2: Calculating the audio recording's starting timestamp

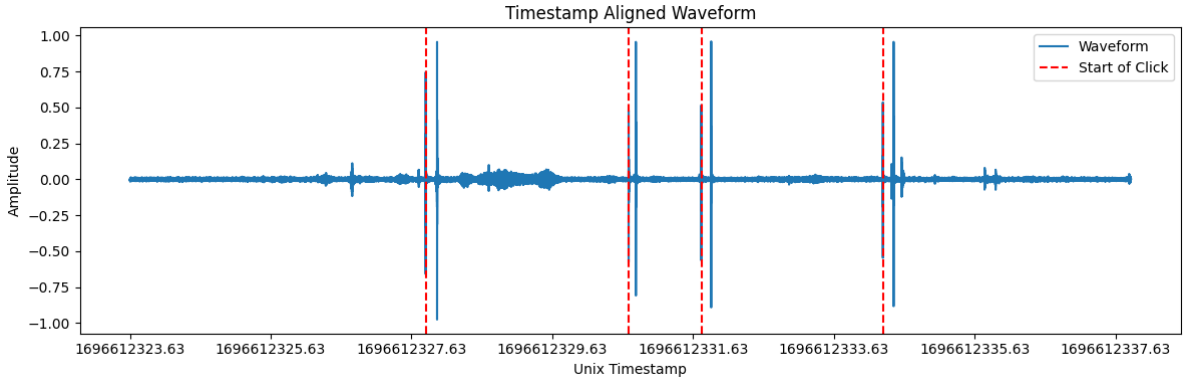


Figure 4.2: Waveform with Synchronized Timestamp

The graphical representation in Figure 4.2 shows the audio waveform with the x-axis represented in Unix epoch timestamps. The previous steps isolate mouse click events from the audio file and use their timestamps to synchronize the audio data. Consequently, we can now use the two datasets for future analysis.

5

Acoustic Analysis of Mouse Movements

This research attempts to determine the correlation between audio signals and mouse movements. While existing literature has discussed the inference of clicking and scrolling events through acoustic signals, the potential for deducing mouse movements from these emissions remains unexplored. The foundational phase of this thesis is centered around investigating the acoustic leakage model of a computer mouse. Initial experiments were conducted using a single microphone, focusing on identifying and categorizing distinct movement patterns. This section addresses the central research question: *Can the sounds generated by the movements of a computer mouse be used to infer its trajectory*? Through this question, the study aims to contribute novel insights to the field of acoustic side-channel attacks on HIDs.

5.1. Single Microphone Analysis

The research started with one microphone near a mouse pad to record the computer mouse movements. The use of a singular microphone was done in order to test the limits of one axis for recording. As machine/deep learning algorithms were used, one microphone was thought to capture enough granularity to determine the two-dimensional position of the mouse on the mouse pad. In the experiments detailed in this section, we will explore this theory and understand the capabilities that one microphone brings to analyze the leakage model of a computer mouse. Subsequent experiments involved varying the microphone placement, altering the surrounding environment, and using diverse mouse pads. The results presented in this section are consistent experiment results even with varying experimental environments.

The first step in the work is to devise a plan to collect and synchronize the audio and mouse movements of these two datasets. As this only contains one microphone connected to the same laptop as the mouse, we used a single script that runs in a loop and records the audio frame and mouse position simultaneously, as seen in listing 5.1. The mouse positions are x and y coordinates on the screen, which could later be converted into directions for analysis. The audio data is captured using the sound card and can be configured to have various sampling rates, formats, channels and chunk sizes. Throughout the research and experiments, various parameters will be tested and evaluated; however, with the single microphone setup, different configurations never had a large impact on the results.

```
while True:
    x, y = pyautogui.position()

    audiodata = stream.read(config.chunk)

    audioframes.append(data)
```

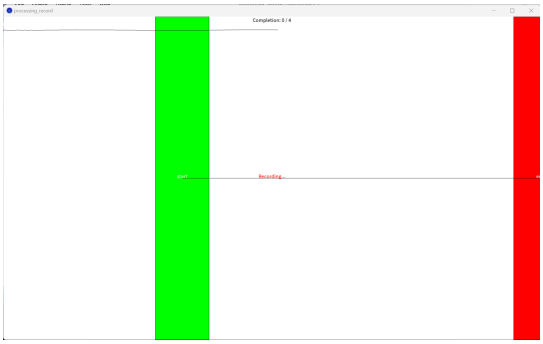
```
7     mouseframes.append((x, y))
```

Listing 5.1: Recording audio and mouse data simultaneously

The raw datasets provide audio and mouse data points that must be analyzed to extract the important features. There are several ways to analyze and compute the data points that depend on the required experiment. The x and y coordinates of mouse positions in mouse movements do not provide important information, as movement depends on direction vectors. Thus, calculating the difference between the x and y coordinates at two moments allows us to determine a general direction of movement. Furthermore, the direction vector can also be converted to an angle by calculating the `arctan2` between the y and x direction difference. For the audio, as machine learning will be applied to the acoustic signals, the acoustic signal can be feature extracted using various transformations, e.g. MFCCs, that will depend on the experiment. Combining these data points provides a dataset for machine learning computation.

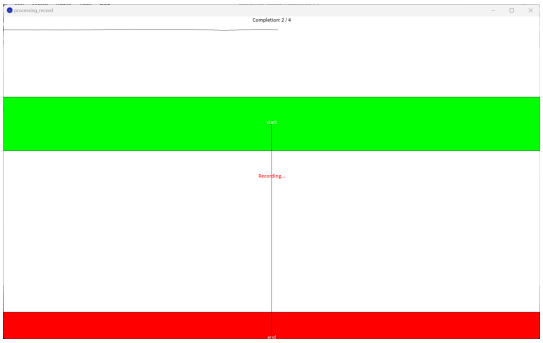
5.1.1. Audio Capture via Processing Software

The initial phase of the experiment aimed to assess whether an acoustic leakage model could distinguish between four basic directional movements of a mouse: up, down, left, and right. Specifically, the experiment sought to determine the feasibility of detecting mouse movements along the cardinal directions using acoustic signals. To facilitate this, a microphone was positioned on the left side of a right-handed mouse pad. The mouse was then moved repetitively along the x-axis and y-axis. These movements were captured using specialized experimental software designed to record these events. The software utilized a Java Processing sketch, with defined starting and ending points activated with hovering. The starting point, highlighted in green, initiated the recording process, whereas the red-colored ending point stopped it, as seen in Figure 5.1. As a result, the Processing sketch returned acoustic representations of the mouse's movements in various directions.

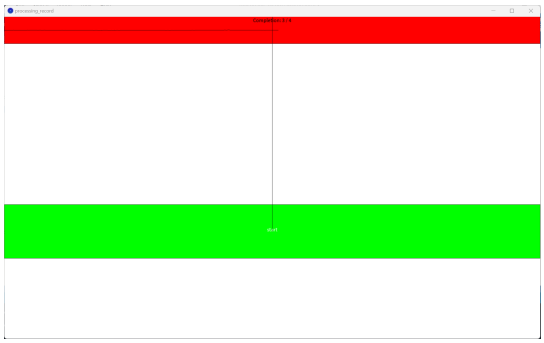


(a) Record from Left to Right

(b) Record from Right to Left



(c) Record from Up to Down



(d) Record from Down to Up

Figure 5.1: Processing Record different recording setups

The LEFT category classification corresponds to movements from the left of the mouse pad to the right, so moving away from the microphone. The RIGHT category classification corresponds to movements

from the right of the mouse pad to the left, so moving towards the microphone. The UP category classification corresponds to movements from the top of the mouse pad to the bottom, while the DOWN category classification is the inverse. This allowed us to run the classification on four categories, serving as proof of concept to identify axis movements that the microphone is sensitive to, as displayed in Figure 5.2. The axes indicate various mouse usages, can reveal information on the general direction of mouse displacements, and serve as a good proof of concept.

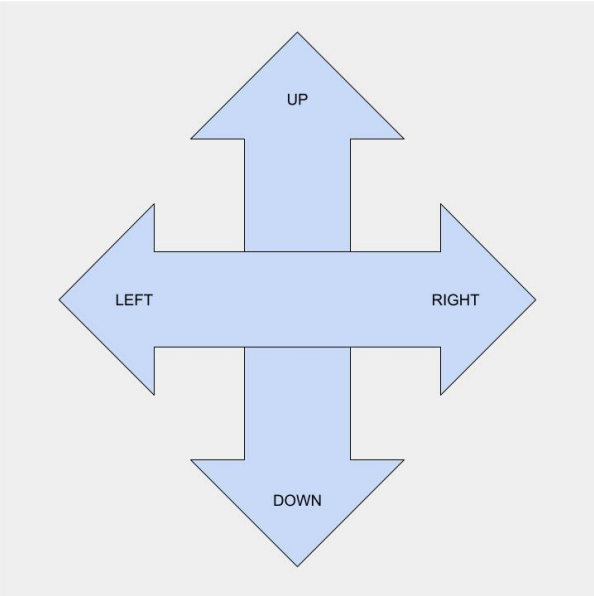


Figure 5.2: Types of movement directions

In the initial experiment, 6,000 samples were collected for each direction, a cumulative total of 24,000 samples. A crucial note is that the acoustic signal dataframes, due to the characteristics of the recording procedure, did not conform to a uniform size. A preprocessing step, illustrated in Listing 5.2, was used to ensure a homogenous and structured input to machine learning models. The corresponding code initiates by calculating the mean (`mean_length`) and standard deviation (`std_length`) of the lengths of the acoustic signal sequences. Then, the optimal sequence length (`max_sequence_length`) is defined with the sum of the mean length and one standard deviation. Given that the data assumes a normal distribution, this `max_sequence_length` should contain approximately 68% of the data. After that, the `pad_sequences` function from TensorFlow's Keras API standardizes all sequences within the dataset to this uniform length by appending zero-padding or truncating as requisite, as seen in the example of Figure 5.3.

```
# Code to preprocess the sequences and labels
from tensorflow.keras.preprocessing.sequence import pad_sequences
import numpy as np

# Code to determine optimal sequence length
sequence_lengths = []
for sequence in tqdm(dataset):
    sequence_lengths.append(len(sequence))  # assuming 'sequence' is list-like
mean_length = int(np.mean(sequence_lengths))
std_length = int(np.std(sequence_lengths))
print(mean_length, std_length)
max_sequence_length = mean_length + 1 * std_length

# Ensure all sequences in the dataset have consistent length
dataset_padded = pad_sequences(dataset, padding='post', truncating='post', maxlen=
    max_sequence_length, dtype=np.float32)
# Convert labels to a NumPy array
```

```
labels_array = np.array(labels)
```

Listing 5.2: Computing acoustic signals

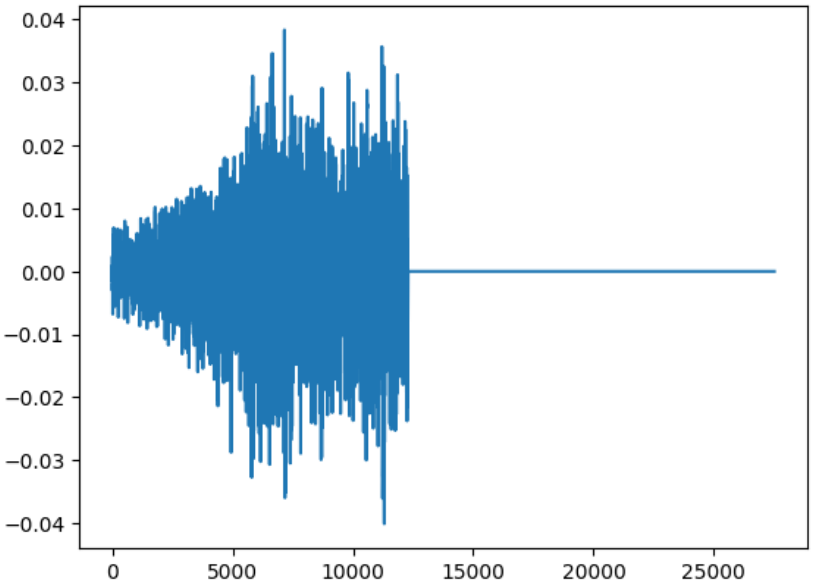


Figure 5.3: Example of a padded audio wave to the appropriate length

```
model = models.Sequential([
    layers.Conv1D(32, 3, activation='relu', input_shape=X_train.shape[1:]),
    layers.MaxPooling1D(2),
    layers.Conv1D(64, 3, activation='relu'),
    layers.MaxPooling1D(2),
    layers.Flatten(),
    layers.Dense(num_classes, activation='softmax')
])
```

Listing 5.3: ML model - 4 category classifications

The signals computed by Listing 5.2 are then transformed using the MFCC technique, which facilitates the conversion of these signals into a two-dimensional format suitable for machine learning algorithms. The specific machine learning model employed, as detailed in Listing 5.3, incorporates two convolutional layers CNN to categorize the data into four distinct classes. At this point, we end up with two lists, one of MFCCs and the other of encoded labels from 0 to 3 corresponding to each direction. The list is split into training and testing samples, with the testing dataset taking up 35% of the original dataset size. The data are entered in a TensorFlow CNN model described in Listing 5.3, compiled with the default Adam optimizer and categorical cross-entropy as a loss function. The model is run using ten epochs. While the model may appear complex, the experiment showed no signs of overfitting by observing the loss functions in 5.4. In addition, the lack of overfitting was verified with a second validation dataset split before any computations. After several verifications, the results of this experiment were deemed reliable and represented what was intended through the experiment. This model has demonstrated a high level of accuracy, achieving a 97% classification accuracy rate, with F1-scores ranging between 97% and 98% for each category. These performance metrics are illustrated in Table 5.1, while the accompanying graphs depicting the model's overall accuracy and loss are presented in Figure 5.4.

This experiment underscores the capability of a solitary microphone to differentiate between four types of mouse movements across various axes. It is important to acknowledge that this study highlights the feasibility of using a single microphone to capture and interpret audio data emanating from computer mouse movements. Based on this approach, it also suggests the potential viability of a side-channel attack. While the current experiment is limited to simplified and constrained movements, it nonetheless

Class	Precision	Recall	F1-Score	Support
0	0.95	0.98	0.97	2062
1	0.99	0.97	0.98	2104
2	0.97	0.99	0.98	2131
3	0.98	0.95	0.97	2103
		Accuracy	0.97	8400

Table 5.1: Classification Report for classifying four movement categories using Processing Record script

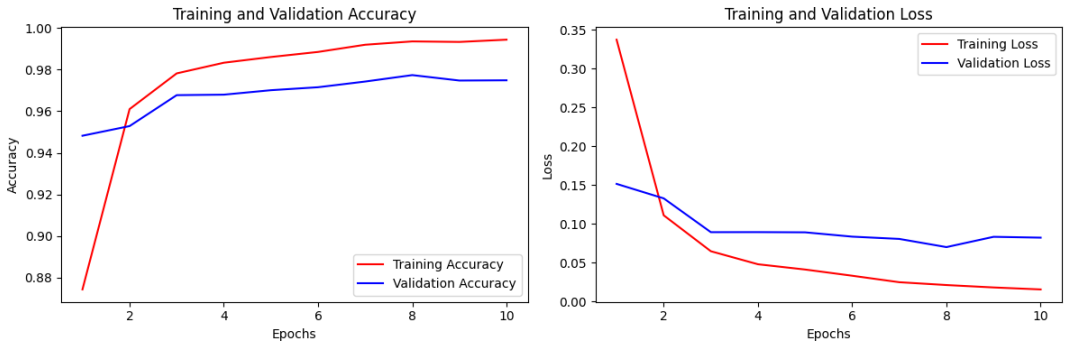


Figure 5.4: Accuracy and loss graph for classifying four movement categories using Processing Record script

serves as a proof of concept, laying the groundwork for more research.

5.1.2. Continuous Audio Capture During Mouse Movements

Using the experiments formed by the Processing sketch, it is an initial proof of concept that shows that in a specific setting, an attacker can infer mouse movements, thus leaking potentially sensitive information. The experiment is still limited to very specific movements; thus, the following experiment attempted to remove the Processing software recording and use continuous audio capture to approach a more real-world setup. This means that the mouse moves on the desktop without predefined recording software, and the audio is recorded simultaneously. The background recording and mouse movements are synchronized to establish a direction vector corresponding to movements for an audio frame. Using this new recording setup, we will experiment with precision in comparison to the previous Processing Sketch experiment.

The experiment consisted of repetitive mouse movements in the cardinal directions, as shown in Figure 5.2. The mouse coordinates are then converted to direction vectors and associated with acoustic signal frames. The acoustic signal chunk sizes are of sample size 8192 with a frame rate of 44.1 kHz. To determine the duration in milliseconds of 8192 frames at a sample rate of 44,100 Hz, we can calculate proportion based on the definition of the sample rate. Specifically, 44,100 Hz (or 44,100 samples/frames per second) are 44,100 frames in 1,000 milliseconds. The proportion is then:

$$\frac{\text{frames}}{\text{milliseconds}} = \frac{44100}{1000} = \frac{8192}{x}$$

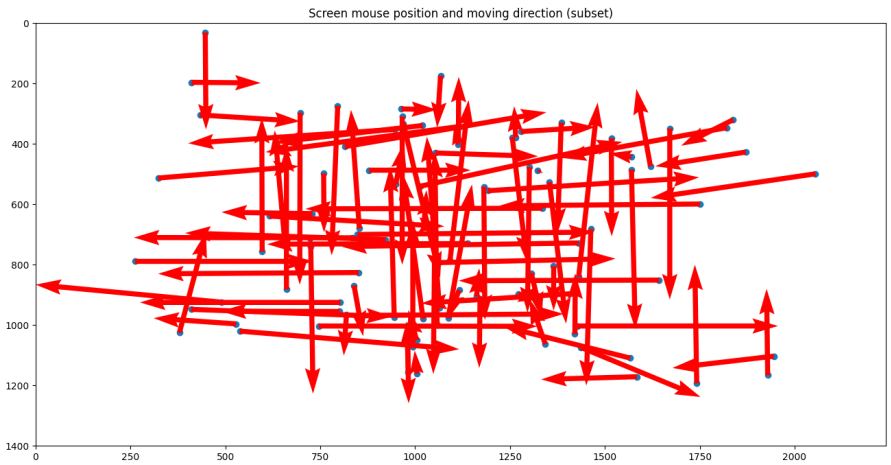
Solving for x gives:

$$x = \frac{8192 \times 1000}{44100}$$
$$\approx 185.941$$

Thus, 8192 frames at a sample rate of 44,100 Hz lasts approximately 186 milliseconds. Accordingly, the mouse coordinates, thus, the direction vector and angle, are measured every 186 milliseconds. As an

example recording, Figure 5.5a shows a graph representing the computer screen. The blue points are mouse positions, while the red arrows represent the direction vectors. The image represents the mouse moving along the screen coordinates. If we take a few points as a subset of this recording, as seen in Figure 5.5b, we can have a clearer image of the up, down, left, and right motions that were recorded. In addition, this visual representation demonstrates the rough movements that resemble the movements captured by the Processing Sketch. This is a general representation as all movements are recorded, even some that do not correspond to one of the four interesting categories. Nonetheless, this is the experiment's goal: to understand if it is still possible to classify using non-perfect recordings.

(a) Mouse movements on screen



(b) Subset of mouse movements on screen

Figure 5.5: Visual representation of the mouse movement direction vectors

Now that we have recorded some samples and have visually confirmed the ability to record the four directions, we need to start labelling and categorizing every direction vector. For classification, the angles are categorized into four directions, representing the four cardinal points. For simplicity, we simply take 25% of the angles on a circle to be labelled and classified as an angle; this can be viewed in figure 5.6. For instance, the first label corresponds to angles 0 to 90 degrees and can be represented as the blue shading in the pie chart of figure 5.6. This means that every direction vector that points upwards will have that same label.

Now that we have the data points, let us utilize the same pipeline as previously described. The audio is preprocessed but does not need to be truncated or padded. The computation of MFCCs and their normalization are the only preprocessing steps required. Using ten epochs with the same TensorFlow

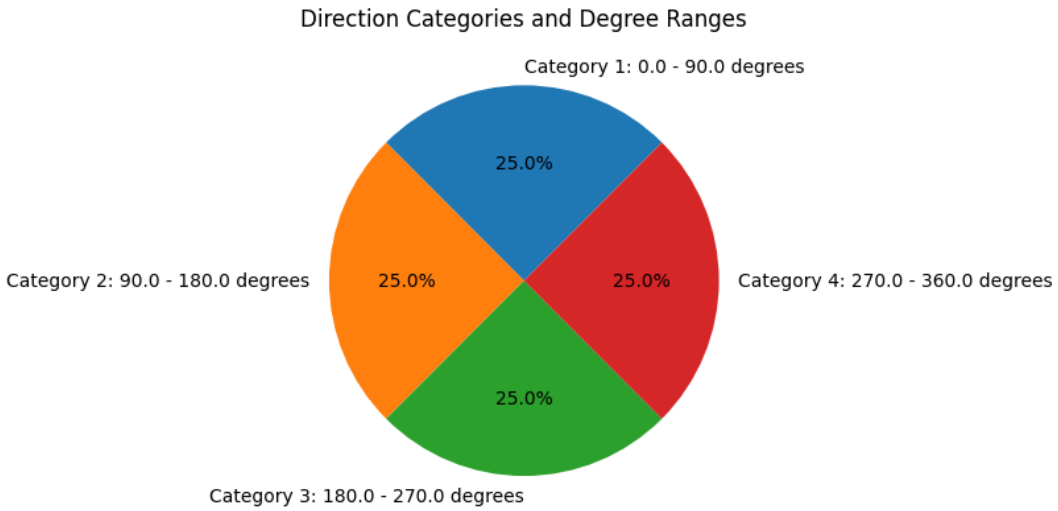


Figure 5.6: Pie chart representing direction categories by slices of angle

CNN model described in Listing 5.3, an overall accuracy of 74% is achieved. The classification report can be viewed in Table 5.2 alongside the accuracy and loss function graph in Figure 5.7.

Class	Precision	Recall	F1-Score	Support
0	0.67	0.75	0.71	718
1	0.80	0.64	0.71	791
2	0.76	0.83	0.79	745
3	0.75	0.75	0.75	816
		Accuracy	0.74	3070

Table 5.2: Classification Report for four area classification using background recording

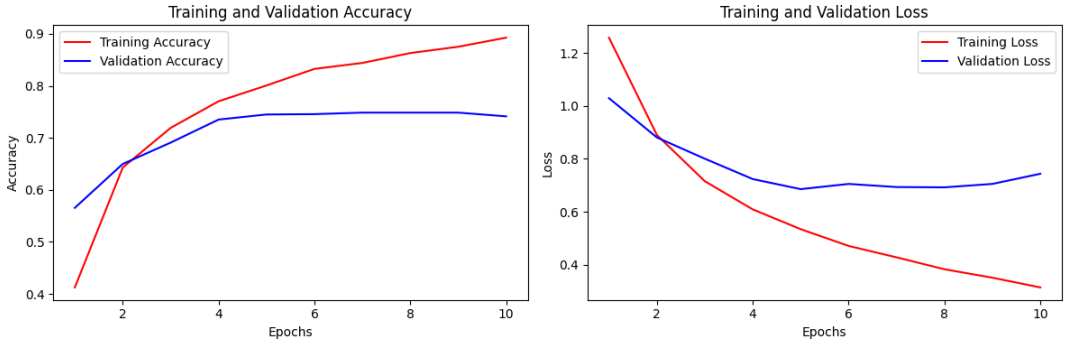


Figure 5.7: Accuracy and loss graph for four area classification using background recording

The accuracy is lower than that of the same experiment using the Processing sketch; however, the recording setup differs. With 74% accuracy, however, we can state with more or less confidence that movements in four directions leak enough audio information to determine some user activities. Other models were tested with varying results. Using a pipeline of a standard scaler to normalize, principal component analysis and a support vector machine, an accuracy of 83% was obtained. This section answers the leakage model research question. There is an acoustic leakage model for SCA, and it is indeed possible to use audio to infer mouse movements.

5.2. Applying the Doppler Effect to Analyze Computer Mouse Movements

Another approach to analyzing mouse movements could involve leveraging physical phenomena. Given the project's focus on interpreting the acoustic patterns generated by the movement of an object, it is natural to examine the Doppler effect. This phenomenon is characterized by a change in frequency perceived when an object in motion passes about a stationary observer. In the context of our research, the mouse serves as the moving object, while the microphone acts as the stationary observer. Although the mouse does not emit a continuous audible signal, the sound generated by its friction can be captured. Hence, exploring the application of the Doppler effect could provide a novel perspective in our study.

```python
def get_information(folders):
    durations = []
    yins = []

    for file in tqdm(folders):
        waveform, sample_rate = torchaudio.load(file, normalize=True)

        # === DURATION ===
        duration_ms = (waveform.shape[-1] / sample_rate) * 1000
        durations.append(duration_ms)

        # === LIBROSA YINS - BASELINE FREQUENCY ESTIMATION ===
        waveform_np = waveform.numpy()

        # Estimate the baseline frequency using librosa's yin function
        f0 = librosa.yin(
            waveform_np[0],
            sr=sample_rate,
            fmin=librosa.note_to_hz('C2'),
            fmax=librosa.note_to_hz('C7')
        )

        yins.append(np.median(f0))

    return np.mean(durations), np.mean(yins), yins
```

Listing 5.4: Calculating noise frequencies

The experimental setup is designed to record mouse movements directed toward and away from the microphone. Approximately 45 samples were gathered for each directional movement. We can use the Python script described in Listing 5.4 to extract the frequency data from these samples. The `get_information` function initializes two lists to store the duration of each audio sample and the estimated baseline frequencies, respectively. For each audio file in the provided folders, the function reads the waveform and the sample rate using `torchaudio.load`, normalizing the audio data in the process. The duration of each sample in milliseconds is calculated and stored. To estimate the baseline frequencies, the script uses `librosa`'s `yin` algorithm, selecting a frequency range corresponding to musical notes between C2 and C7. The average of the estimated frequencies is calculated to compare each measurement, which is the most important data returned to provide a dataset for further analysis.

```python
towards = glob.glob("data/towards-*.wav")
away = glob.glob("data/away-*.wav")

right_to_left_wav = glob.glob("data/whiteboard_cleaner_right_to_left-*.wav")
left_to_right_wav = glob.glob("data/whiteboard_cleaner_left_to_right-*.wav")
print(f"Lengths - LEFT-RIGHT: {len(left_to_right_wav)}, RIGHT-LEFT: {len(right_to_left_wav)}"
    )
# Lengths - LEFT-RIGHT: 45, RIGHT-LEFT: 58

_, tavghz, _ = get_information(towards)
_, aavghz, _ = get_information(away)
# tavghz, aavghz -> (730.460927306092, 715.7556151883197)

rl_avg_ms, rl_avg_hz, rl_freqs = get_information(right_to_left_wav)
```

```
# rl_avg_ms, rl_avg_hz -> (462.0671246408046, 721.8200111240826)

lr_avg_ms, lr_avg_hz, lr_freqs = get_information(left_to_right_wav)
# lr_avg_ms, lr_avg_hz -> (562.1809027777778, 708.271970714366)

freq_approaching = rl_freqs
freq_moving_away = lr_freqs

# Assuming freq_approaching and freq_moving_away are your data
t_statistic, p_value = stats.ttest_ind(freq_approaching, freq_moving_away)

print("t-statistic:", t_statistic)
print("p-value:", p_value)
# t-statistic: 1.256868339502702
# p-value: 0.21170024395607887

# calculate the effect size
effect_size = np.abs(t_statistic)

# parameters for power analysis
alpha = 0.05  # significance level
power = 0.8  # desired power

# perform power analysis
analysis = TTestIndPower()
result = analysis.solve_power(effect_size, power=power, nobs1=None, ratio=1.0, alpha=alpha)
print('Sample size should be more or less - %.3f - to show statistical significance.' %
    result)

# Sample size should be more or less - 10.986 - to show statistical significance.
```

Listing 5.5: Statistical signficance experiment

The resulting values can be seen below:

1. Right to Left - average duration: 462 ms
2. Right to Left - average frequency: 721.8 Hz
3. Left to Right - average duration: 562 ms
4. Left to Right - average frequency: 708.3 Hz
5. t-statistic: 1.257
6. p-value: 0.212
7. Expected sample size for statistical significance: 11

While the measurements initially appear promising as a higher frequency is measured during approaching movements, they do not show statistical significance, as demonstrated in the p-value test of 0.2117. Additionally, when applied to previously collected data from identical experiments, the method fails to yield successful results and, thus, is unreliable. We have two datasets, and although the experimental methodology is correct, neither dataset achieves statistical significance. The Python script in Listing 5.5 details the measurements and their outcomes. In the code, we have two sets of audio files that capture mouse movements. Two additional sets represent left-to-right and right-to-left movements. The code retrieves the file paths and then calculates the average frequency for each movement direction using the `get_information` function.

The next statistical analysis is performed using a two-sample t-test, which compares the average frequencies from the two directional movements to assess whether there is a statistically significant difference between them [23]. The resulting t-statistic and p-value are essential indicators: the t-statistic measures the size of the difference relative to the variation in the sample data, and the p-value quantifies the probability that any observed difference has occurred by chance. However, the obtained p-value exceeds the conventional alpha level of 0.05, indicating no statistical significance. This suggests that any observed differences in frequency between the samples might be due to chance rather than a systematic effect. A power analysis is then conducted to determine the necessary sample size to achieve a desired

power level with a certain degree of confidence, set at 0.8 [25]. The analysis suggests that a sample size of approximately 11 for each group would be needed to potentially reveal statistical significance in this experiment, validated with 45 samples per value. Despite having a sample size larger than the one recommended by the power analysis, the lack of statistical significance persists. Given the current conditions, this outcome implies that increasing the sample size is unlikely to reveal any significant effect. The Doppler effect may be too subtle to be detected with the current methodology.

The exploration into the Doppler effect on a $7 \times 7\,\text{in}^2$ mouse pad showed low-frequency changes that lacked statistical significance. It does raise the question of whether the speed of mouse movement might be a contributing factor. By changing the velocity of the mouse across the pad and observing the consequent frequency alterations, we tried to answer that question. Given the speed of sound at 343 m/s or equivalently 34.3 cm/ms, we applied the fundamental speed equation $v = \frac{\text{distance}}{\text{time}}$ with a constant distance equal to the mouse pad's length (7 in or 17.78 cm) and a time interval ranging from 200 ms to 600 ms (times measured during the experiments). The Doppler effect, as experienced by a stationary observer, can be modelled by the equation:

$$f_o = \left(\frac{c}{c - \left(\frac{d}{t}\right)} \right) \times f_s$$

$$f_o = \left(\frac{34.3}{34.3 - \left(\frac{7}{t}\right)} \right) \times 1$$

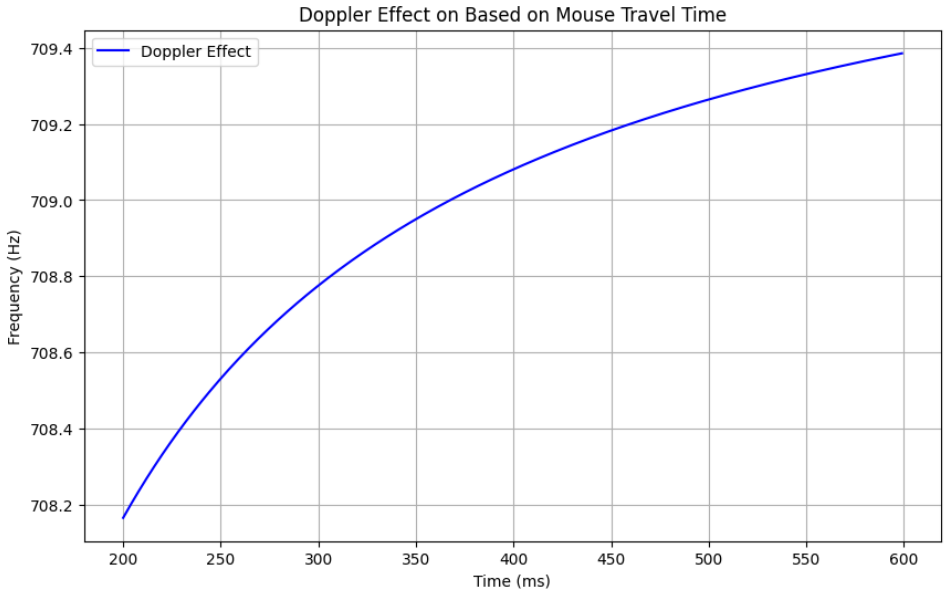


Figure 5.8: Doppler Effect based on Mouse Travel Time on a $7 \times 7\,\text{in}^2$ Mouse Pad

Figure 5.8 illustrates the observed frequency changes over the time interval. The curve's shape suggests a nonlinear relationship between time and frequency, characteristic of the Doppler effect. However, the frequency changes are minimal, rising only by approximately 1.2 Hz . This minor variation over a relatively large mouse movement period suggests that the speed of mouse movement on this scale is unlikely to be significant. This means the Doppler Effect for mouse speeds is not sensitive enough in an experimental context.

In summary, the investigation into the Doppler effect has not yielded statistically significant results. The observed frequency variations were minimal, suggesting that the Doppler effect may not provide

a reliable attribute to infer mouse movements from audio. However, this exploration inspired other researched methodologies. Rather than focusing on changing frequencies, the sound amplitude is a simple measure that will be explored.

5.3. Analyzing Proximity Through Sound Amplitude Variations

The methodology for analyzing acoustic signals required a new approach. The idea was that by examining the amplitude of sound waves, one could infer the proximity of sound-generating movements to a specific microphone. Sounds produced nearer to the microphone register with higher amplitudes than those originating at a greater distance. This variance in amplitude could potentially provide a metric for determining the relative distance of the sound source along an axis. Consequently, the movement of an object, such as a computer mouse, which moves towards or away from the microphone's location could be inferred.

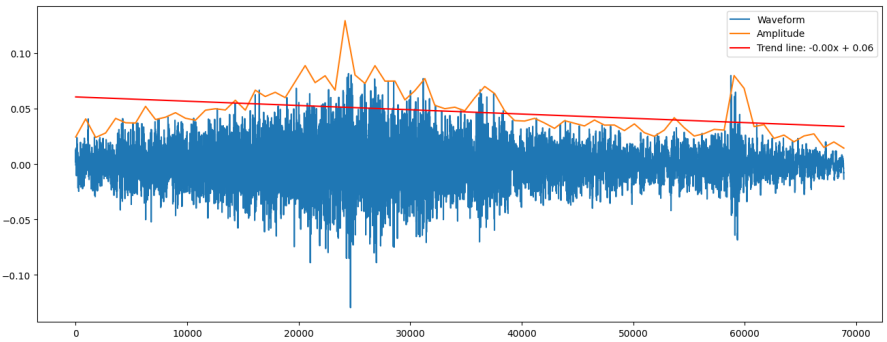
The test involves a controlled series of bidirectional movements along the microphone's directional orientation axis with ten samples. During this process, we will segment the audio recordings into two categories based on the direction of the mouse's movement: from left to right to represent the mouse's trajectory away from the microphone and from right to left to denote its approach towards the microphone. Next, we will plot the audio amplitude over time. This graphical representation will allow us to observe the amplitude changes corresponding to the mouse's directional changes. We anticipate that a progressive increase in amplitude will characterize the approaching movements while the receding movements will demonstrate a corresponding decrease. The analysis aims to confirm the presence of a differential acoustic signature based on directionality and quantify the relationship between amplitude and movement vector relative to the microphone's position.

We will segment the sound into discrete windows for each sound recording of mouse movement. Within each window, we will calculate the maximum amplitude, which will then be plotted to visualize the waveform's differences in time. A linear regression analysis will be applied to the amplitude data points for a better graphical interpretation. This will facilitate the display of a general trend, revealing whether there is a consistent increase or decrease in amplitude. The slope of the regression line will serve as an indicator of this relationship.

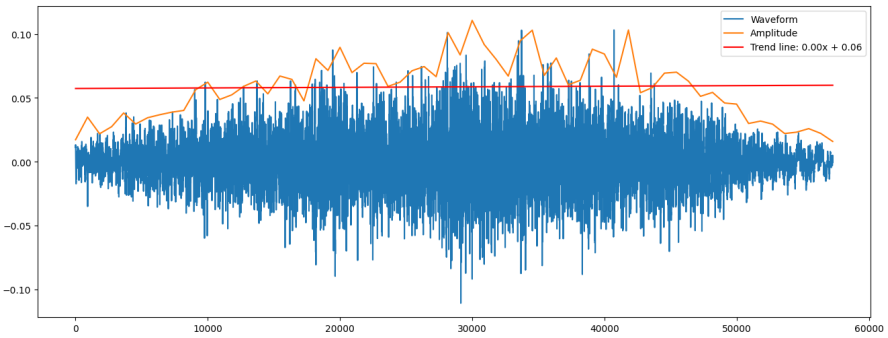
As depicted in Figures 5.9a and 5.9b, the sound amplitude modulation seems to be correlated with the two distinct directional movements of the mouse: away from and towards the microphone. Nonetheless, the slopes do not show *obvious* visual trends. There is an idea, but it became clear that we needed to improve the experimental setup for better analysis; this was done by adding more acoustic context. To do so, the recent tests incorporated three distinct auditory markers recorded at the start and end of the mouse's movement. The markers consist of three circles done with the mouse. This modification aims to extend the time frame in which the microphone captures the audio. It is expected to have a more pronounced and detectable amplitude contrast, simplifying the visual classification task.

The experimental data presented in Figures 5.10a and 5.10b visually illustrate the subtle yet consistent trends in sound amplitude corresponding to the mouse's movement about the microphone. Figure 5.10a captures the slight decrement in amplitude when the mouse has moved away from the microphone, with a negative slope angle value of -0.0000099 degrees. Conversely, Figure 5.10b demonstrates the opposite effect. The slope angle value is 0.0000068 degrees, demonstrating a positive slope in amplitude as the mouse moves toward the microphone. Although the differences are minimal, they are consistent across different tests, which inspires success that a machine learning model could leverage. The graphical analysis allows us to discern differences in movements. However, a machine learning model would be more sensitive to these trends.

The outcomes of these experiments are indeed encouraging. Through amplitude analysis, we have demonstrated the ability to visually determine the direction of a computer mouse's movement, distinguishing whether it is approaching or moving away from a microphone. This technique validates

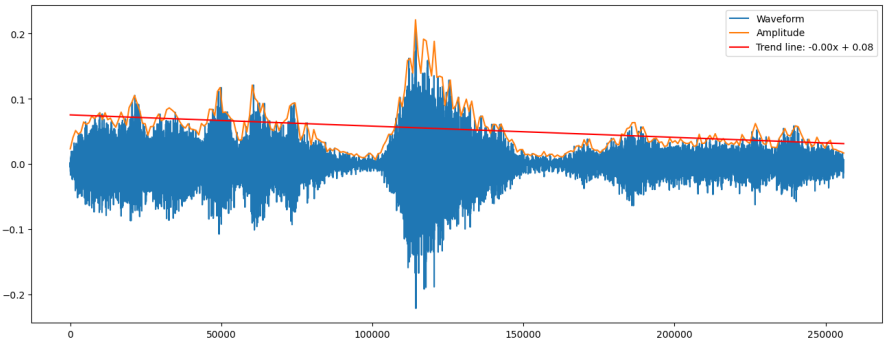


(a) Waveform of the sound moving away from the microphone

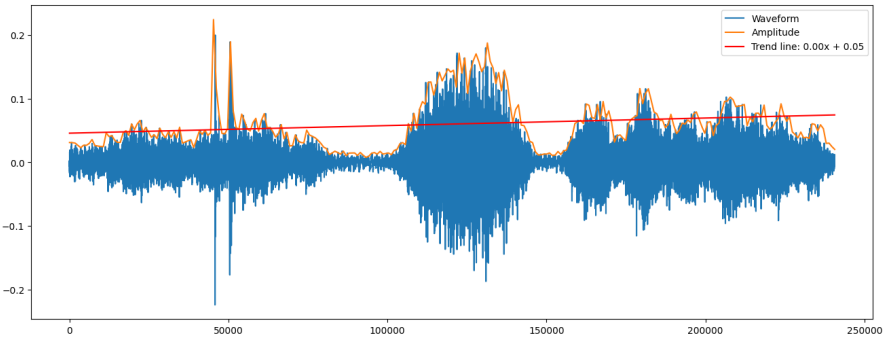


(b) Waveform of the sound moving towards the microphone

Figure 5.9: Waveforms of audio for the mouse moving towards and away from the microphone



(a) Waveform of the sound moving away from the microphone with the amplitude depicting a negative slope trend line



(b) Waveform of the sound moving towards the microphone with the amplitude depicting a positive slope trend line

Figure 5.10: Waveforms of audio for the mouse moving towards and away from the microphone

that a mouse's distinctive movements along a single axis can be accurately inferred. This breakthrough is particularly intriguing as it lays the foundation for more complex tracking. We can extend our detection

capabilities to a second axis by integrating a second microphone and effectively capturing the full scope of movements across a mouse pad's surface. With these results, our next research phase will explore two-dimensional mouse movements.

5.4. Conclusion: Affirming Mouse Motion Inference from Mouse Acoustic Emissions

This research set out to address the question: *Can the audio signals generated by the movements of a computer mouse be used to infer its trajectory?* Our exploration encompassed three techniques, each contributing to our understanding of sound emissions in the context of mouse movement. The initial approach adopted a data-intensive strategy, relying on machine learning algorithms to understand correlations within the collected acoustic data. This method proved successful, demonstrating that specific mouse movements, up, down, left, and right, could be inferred from their associated audio leakage. We then investigated a more physics-based method by applying the Doppler effect. Although this approach did not yield definitive results, it paved the way for a simpler yet effective audio amplitude analysis technique. By examining sound amplitude variations, we could visually and quantitatively detect differences in one-dimensional mouse movements. Looking ahead, as detailed in Chapter 6, we aim to further evaluate and refine these techniques, particularly focusing on enhancing the granularity and accuracy of our methods. Our research has demonstrated that acoustic emissions from computer mouse movements can be analyzed to infer their trajectory, thus answering the research question. It is now important to ask how precise these attacks are.

6

Accuracy in Mouse Movement Inference

Building upon the findings from Chapter 5, which established the feasibility of inferring mouse movements through acoustic signals, this chapter dives deeper into this research. The ability to track mouse movements via emitted sound uncovers a potential security SCA vulnerability in computer mice and raises questions about the accuracy and limits of such attacks. Therefore, the next phase of our research is dedicated to examining the granularity of the investigated techniques. The question we address is: *To what extent can we accurately and precisely infer mouse movements from acoustic data?* Understanding the resolution and limitations of this acoustic leakage model is important for contextualizing the security risk it poses. By quantifying the extent of this vulnerability, we can better assess its impact in real-world scenarios.

6.1. Refining Angle Categorization: Enhanced Granularity with a Single Microphone

The previous experiments from section 5.1 have shown acoustic leakage that can be split into four categories, which is an interesting result; however, it does not represent truly realistic mouse movements. Naturally, the idea is to increase the level of granularity now in order to understand what angles are discernible with a single microphone and machine learning algorithm. For instance, if a mouse movement angle can be inferred from a short sound sample, then a string of inferences would allow us to track movements and determine the mouse positions on the screen continuously.

In this section, we expand the directional classification model. The preceding section, chapter 5, detailed the success in differentiating between the four cardinal directions – specifically, 0°(up), 90°(right), 180°(down), and 270°(left). Building upon this foundation, we now focus on augmenting the model's precision by incorporating intermediate directions, namely 45°, 135°, 225°, and 315°. The goal is to serve as a benchmark for evaluating the capability of our model to recognize more angle complexities in the audio. Figure 6.1 shows a radial histogram detailing the distribution of recorded angles. The histogram is partitioned into multiple categories, each represented by a color bin. These categories correspond to the various directional angles, segmented in increments of 45°with a leniency of 20°. A total of 23,259 samples were taken, with between 2,000 and 3,500 samples for each direction. Upon close inspection, it is evident that each colored bin captures similar observations. This uniformity suggests that the dataset provides an equal distribution across all directional categories. Such a balanced dataset is beneficial for machine learning tasks as it avoids potential biases associated with underrepresented or overrepresented classes.

The same machine learning explained in Listing 5.3 was initially used with different epoch counts

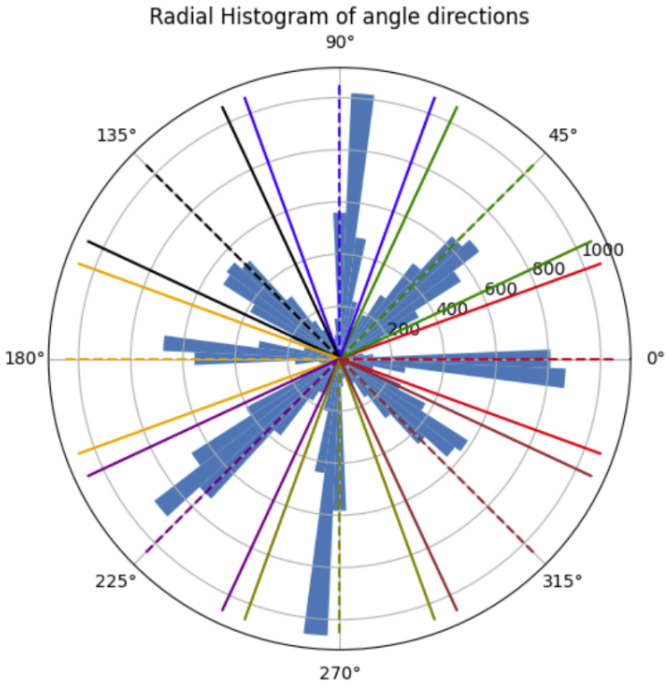


Figure 6.1: Radial histogram of angle directions

from 10 to 50. All of these attempts led to the same results. Table 6.1 presents the classification report corresponding to the radial histogram-based machine learning model. This report offers a classification report of the model's performance across individual classes (denoted by numbers 0 to 7). In addition, Figure 6.2 demonstrates the training and validation accuracy and the loss to follow the machine learning's learning process across ten epochs.

Class	Precision	Recall	F1-Score	Support
0	0.11	0.10	0.11	724
1	0.13	0.14	0.14	805
2	0.12	0.11	0.11	618
3	0.16	0.16	0.16	1095
4	0.12	0.11	0.11	795
5	0.13	0.15	0.14	817
6	0.18	0.23	0.20	1026
7	0.16	0.12	0.14	854
		Accuracy	0.14	6734

Table 6.1: Classification report for classifying eight movement categories

The main observation from the classification report, seen in Table 6.1, is that the model does not have an accuracy above 15%. Despite employing a machine learning framework similar to prior successful models, the algorithm's performance with the eight different categories is suboptimal. Different machine learning models and multiple datasets that covered various environments were tested. From simple SVR CNNs to more complex CNNs, these experiments are not described for simplicity; the results were the same, and all models could not differentiate between multiple eight categories.

In our initial attempt, we evaluated our model to distinguish between all eight directions simultaneously. The results indicate that this approach was not effective. Following a hierarchical learning method, we will explore inferring larger chunks of data by merging certain angle categories. This hierarchy attempts to follow Behavicker's paper model [3] by using various models that incrementally learn the outcome

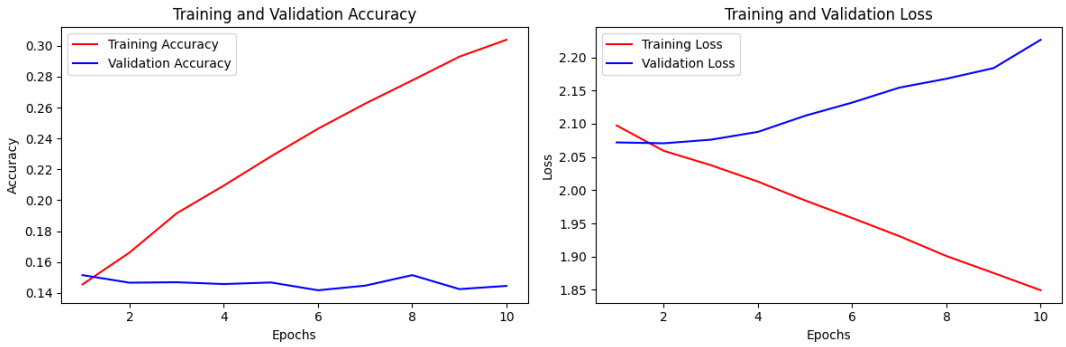


Figure 6.2: Accuracy and loss graph for classifying eight movement categories

more precisely. For example, we might combine the up-down and diagonal movements to represent distinct axes, aiming to discern if such concentrations of classes improve performance to a point which allows future subdivision. The circle in Figure 6.3 demonstrates the methodology adopted for axis merging. Each color represents a distinct merged axis:

- The **blue** line denotes the vertical axis, merging the upward and downward motions.
- The **red** line indicates the horizontal axis, representing both leftward and rightward movements.
- The **cyan** lines represent the two diagonal axes, merging larger categories.

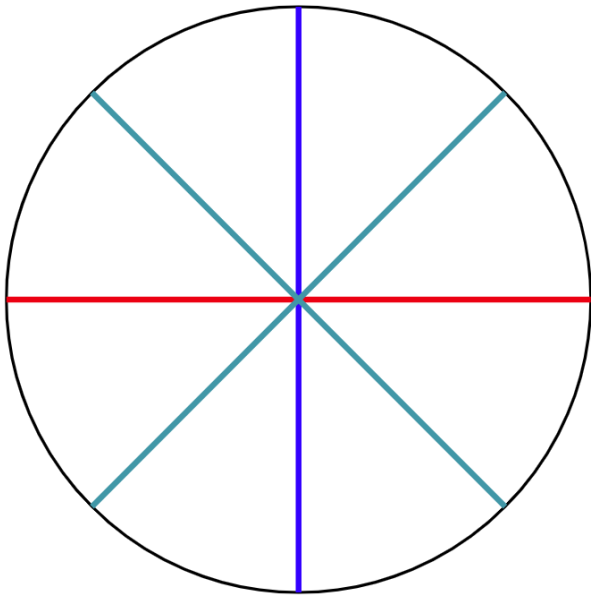


Figure 6.3: Representation of merged axes with three categories

Class	Precision	Recall	F1-Score	Support
0	0.59	0.60	0.59	3683
1	0.28	0.28	0.28	1400
2	0.31	0.29	0.30	1651
		Accuracy	0.46	6734

Table 6.2: Classification Report for classifying three merged movement categories

After consolidating the axes as illustrated in Figure 6.3, the machine learning model was retrained, and its performance metrics were re-evaluated. Table 6.2 shows that the model's accuracy is 46%. This level of accuracy is notably low for practical applications; thus, this model is not ideal. Merging categories

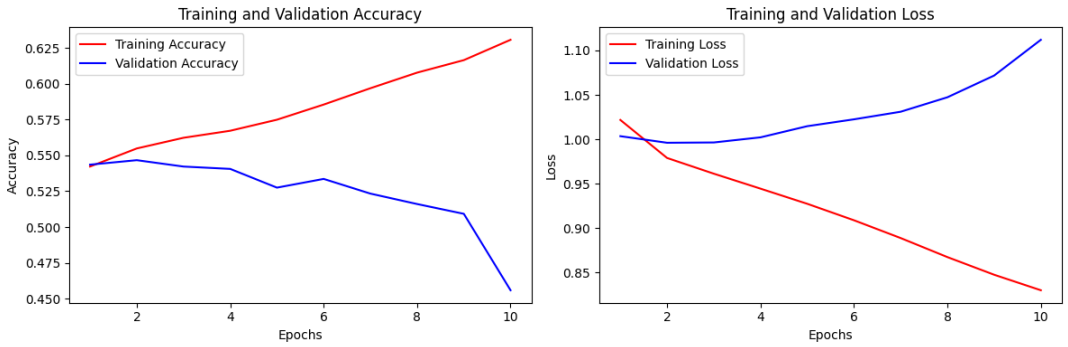


Figure 6.4: Accuracy and loss graph for classifying three merged movement categories

should have effectively performed simpler training on higher-level data, but the model's performance is lacking. This unsuccessful model, which is only an example of multiple attempted experiments, demonstrates the incapability to classify complex mouse movements using acoustic signals. As up-down and left-right motions were discernible, the model performed adequately. However, once the diagonals are added, it becomes limited, indicating that the audio similarities confuse the model and limit its classification capabilities.

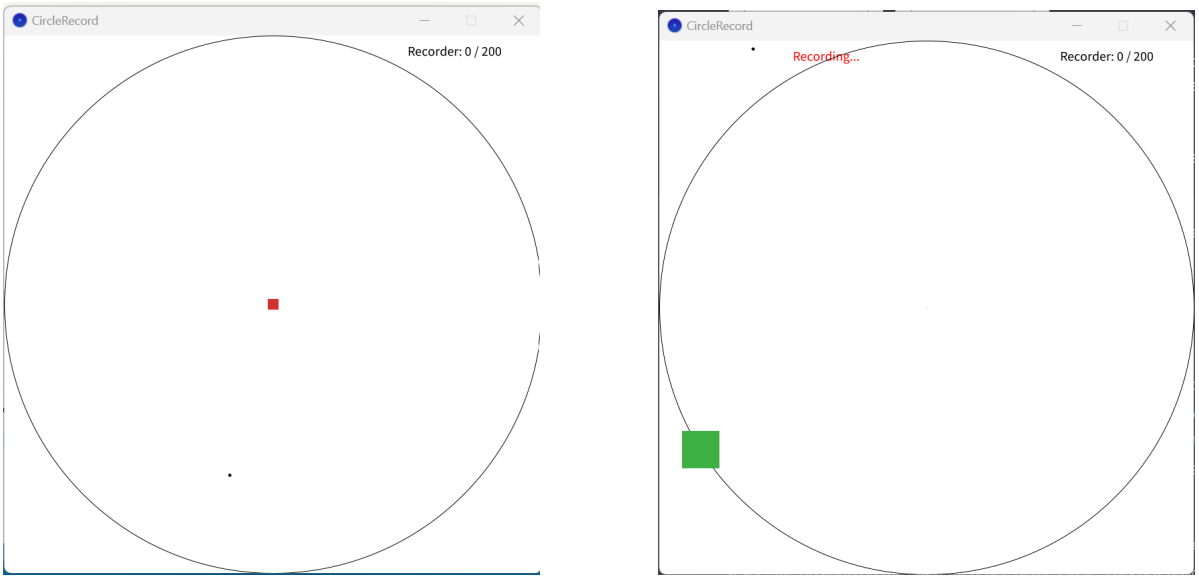
The experiments consistently fell short of expectations in our efforts to distinguish incrementally between the movement categories. Despite varying experimental setups, this pattern raised questions about the underlying data and its context. It became clear that a singular microphone was a limiting factor in capturing similar mouse movement data. We learned that relying on a single microphone is not a viable strategy to increase the granularity of the leakage model. This realization made us redirect our focus toward exploring other methods for mouse movement inferences.

6.2. Continuous Angle Prediction: A Regression Approach

Due to the previous classification experiments, another direction had to be explored with a singular microphone. Rather than classifying angles into bins, it would be better to have a regression learning algorithm to predict the angle continuously. The primary inputs in the proposed regression pipeline are audio waveforms with a sampling rate of 44100 Hz and a duration of 50 milliseconds. These waveforms are processed and transformed into MFCCs. Next, the transformed audio signals are batched and readied for training. The CNN model comprises convolutional layers and fully connected layers to map to the target angle prediction. An L1 loss function, which simply calculates the mean absolute difference, is used to approach the target angle as closely as possible. The output from the model provides an angle in degrees.

6.2.1. Initial Regression

The new recording method, depicted in Figure 6.5, involves users starting from the screen and mouse pad's center to reach circularly arranged targets. This approach standardizes the starting position, minimizing variability, and allows for the measurement of random angle movements ranging from 0 to 360 degrees. In the experimental setup, the user uses two distinct interfaces. As seen in Figure 6.5a, the interface initiates by displaying a target red dotted center, which will start the audio recording. This forces the user to position the mouse at the center of the screen and mouse pad before starting any movement. This ensures a standardized starting point for all recorded movements, giving consistent and reliable data collection. Following this, as illustrated in Figure 6.5b, a green target box can appear at any position on the circle's circumference. This box represents the target destination to which the user must move the mouse. The position of the green box is determined randomly. Moving from the standardized center point to various green box targets, we can generate a dataset of acoustic signatures associated with different angular movements. Figure 6.6 represents the distribution of recorded angles ranging from 0 to 360 degrees. The histogram is divided into bins of 10 degrees each. It can be observed that the frequencies of occurrences for the angles are relatively uniform across the entire range. This



(a) Move cursor towards red center to start the recording

(b) Move cursor towards green box to stop the recording

Figure 6.5: New circular recording method

suggests that there is not a significant bias towards any particular angle in the recorded data.

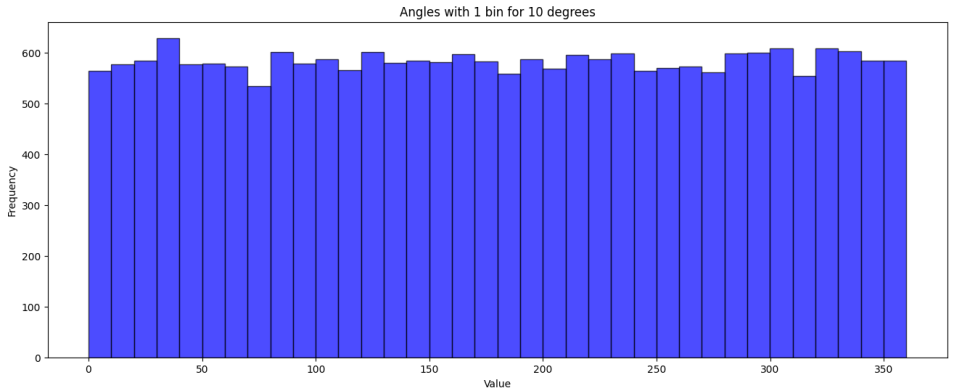


Figure 6.6: Histogram showcasing the distribution of recorded angles

There are around ~21,000 samples in the dataset of audio MFCCs and angles corresponding to a direction in this example dataset for the report. The data is then split into training and testing datasets to prepare for PyTorch deep learning. The code begins with the definition of a simplified CNN named `SimpleCNNModel`. The network architecture can be viewed in Listing 6.1. It contains a simple convolutional layer and a fully linearly connected layer to output the final regression value.

```
class SimpleCNNModel(nn.Module):
    def __init__(self):
        super(SimpleCNNModel, self).__init__()
        # One convolutional layer
        self.conv1 = nn.Conv2d(1, 4, (3, 3))
        # One linear layer
        self.fc1 = nn.Linear(4 * 63 * 52, 1)

    def forward(self, x):
        x = nn.functional.relu(self.conv1(x))
        x = nn.functional.max_pool2d(x, 2)  # Pooling layer
        x = x.view(x.size(0), -1)  # Flatten
        x = self.fc1(x)
```

```
        return x

# Initialize the model and move it to GPU
model = SimpleCNNModel().to(device)
```

Listing 6.1: CNN Model Architecture

The training uses the Mean Absolute Error (L1) as its loss function and the Adam optimization algorithm. Training is conducted over 150 epochs. For each epoch, the model iterates over the training data. For every batch in this data, it computes the forward pass, calculates the loss, and performs backpropagation to adjust the model's weights. The average loss for each epoch is then computed and displayed.

The scatter plot shown in Figure 6.7 visualizes the model's performance by plotting the true values from the test set against the predicted values generated by the model. Each point on the graph represents a test sample, with its x-coordinate being its true value and the y-coordinate being the value predicted by the model. The red dashed line represents the ideal scenario where every prediction matches the true value perfectly. Most predictions are not clustered around this line, indicating that the model's predictions are inaccurate.

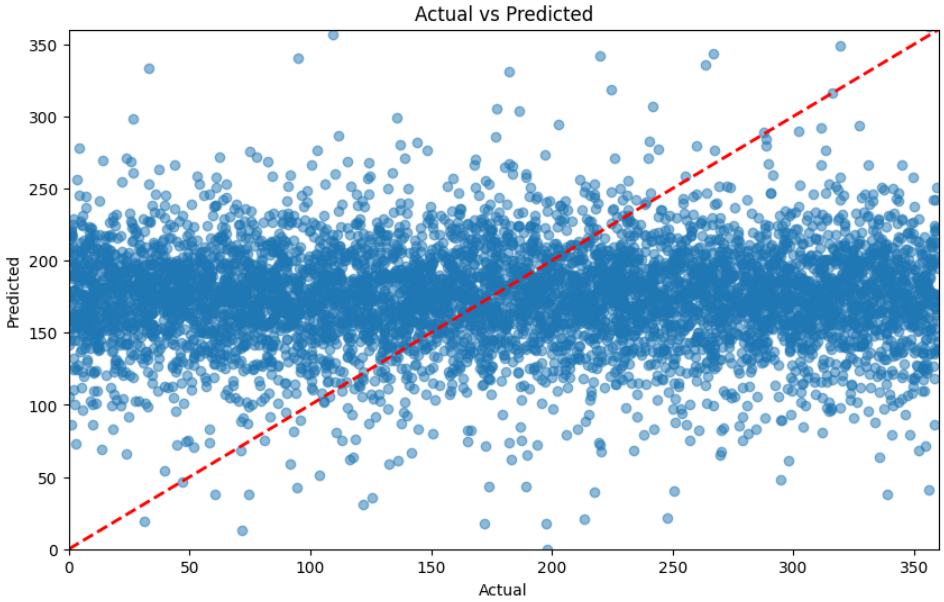


Figure 6.7: Actual vs Predicted values scatter plot

This section attempted to predict the continuous angle values. The model failed in its prediction, and this was consistent with multiple datasets, models, and parameters throughout various experiments. The loss and optimizer functions were swapped out. However, no improvements were noticed. This failure continued the trend, indicating that a single microphone could not accurately predict mouse movement angles.

6.2.2. Cosine and Sine Regression

Predicting angles from 0 to 360 degrees presents an inherent challenge, mainly due to the circular nature of angle measurements. For instance, angles close to 0 degrees and 360 degrees are nearly identical in orientation but are numerically distant. This discontinuity can challenge regression models, "angle periodicity" [12]. To address this, a new experiment was conducted where angles were represented using their cosine and sine values [12].

The changes to the code were not much different; the dataset is set to load the cosine and sine of the angle as label values. A CNN with two convolutional layers and three fully connected layers was designed.

The final layer outputs two values corresponding to the cosine and sine of the angle. Since both targets are continuous and range between -1 and 1, the Mean Squared Error (MSE) loss was chosen as the objective. Given the two outputs, individual losses were computed for the cosine and sine predictions. Mathematically, for true values $y_{\cos}$ and $y_{\sin}$ and their respective predictions $\hat{y}_{\cos}$ and $\hat{y}_{\sin}$, the combined loss L is defined as:

$$L = (y_{\cos} - \hat{y}_{\cos})^2 + (y_{\sin} - \hat{y}_{\sin})^2$$

This formulation ensures that errors in the sine and cosine predictions contribute equally to the total loss.

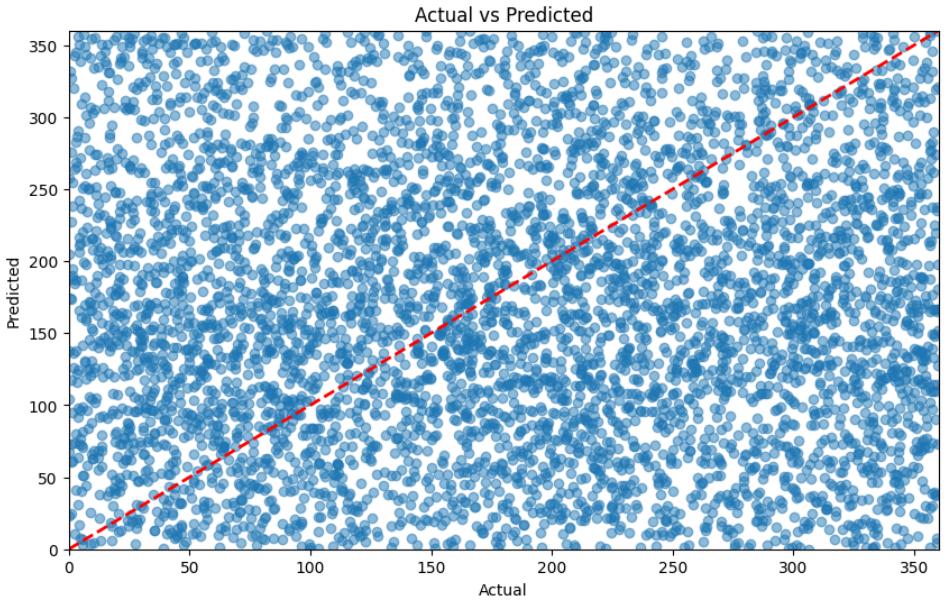


Figure 6.8: Actual vs Predicted values scatter plot for Cosine and Sine regression

Using the same data, model (except with two outputs) and training parameters, the regression on cosine and sine was performed, giving once again suboptimal results. After 150 epochs, the average angle difference was 88.79°, near a random angle choice. Figure 6.8 shows the scatter plot, demonstrating that the predicted angles do not approach the red target line. The MSE loss is such a substantial deviation that it indicates that the model's predictions are nearly orthogonal to the true values, suggesting that the performance is similar to a random guess and is suboptimal. If a correlation exists between the audio generated by mouse movements, the model should identify and indicate this relationship. Once again, the research hit a dead end, and new techniques must be explored.

6.2.3. Data Augmentation for Regression

At this juncture, various experiments with diverse models, data points, and collection methodologies failed to show notable success. Beyond the inference of the four cardinal directions—Up, Down, Left, and Right there was no evidence supporting the detection of arbitrary angles ranging from 0° to 360° using one microphone. A common issue was the overfitting in the models. Overfitting occurs for various reasons, such as complex models, noisy data, too many epochs and lack of data. Through different experiments, it seemed that simple or more complex models did not impact the results, as well as the changes from 10, 50, 100, 150 and 200 epochs. In addition, the datasets were provided using different experimental recording setups; thus, as noisy as the data gets, a hint of direction would have become evident in the multitude of experiments. Consequently, that did not seem to be the issue. One of the last remaining possible issues was the amount of data in the dataset. Throughout the experiments, different datasets from 10,000 to 50,000 samples were used. Thus, it did not appear that too few samples were an issue; however, it was the only missing variable. Data augmentation was used to explore this avenue.

Data augmentation is a technique that involves creating new training samples by applying various transformations to the original data. These transformations can include rotations, translations, scaling, and other perturbations, ensuring the model is exposed to a wider variety of data during training. By adding diversity to the training data, data augmentation can mitigate the overfitting problem. It offers a model with more scenarios and variations to learn from, thereby improving its generalization capabilities on unseen data.

Data augmentation was performed 2 to 10 times to model the differences. As we will see, the more augmentation, the better the model performed. However, it foreshadowed an overfitting on the randomized augmented data. In addition, the data augmentation was only done by adding random noise to the original data. This was done to conserve the data as closely as possible to the original. The variations in sound should come from white noise style as the most likely in a real-world environment, such as the hum of a laptop fan.

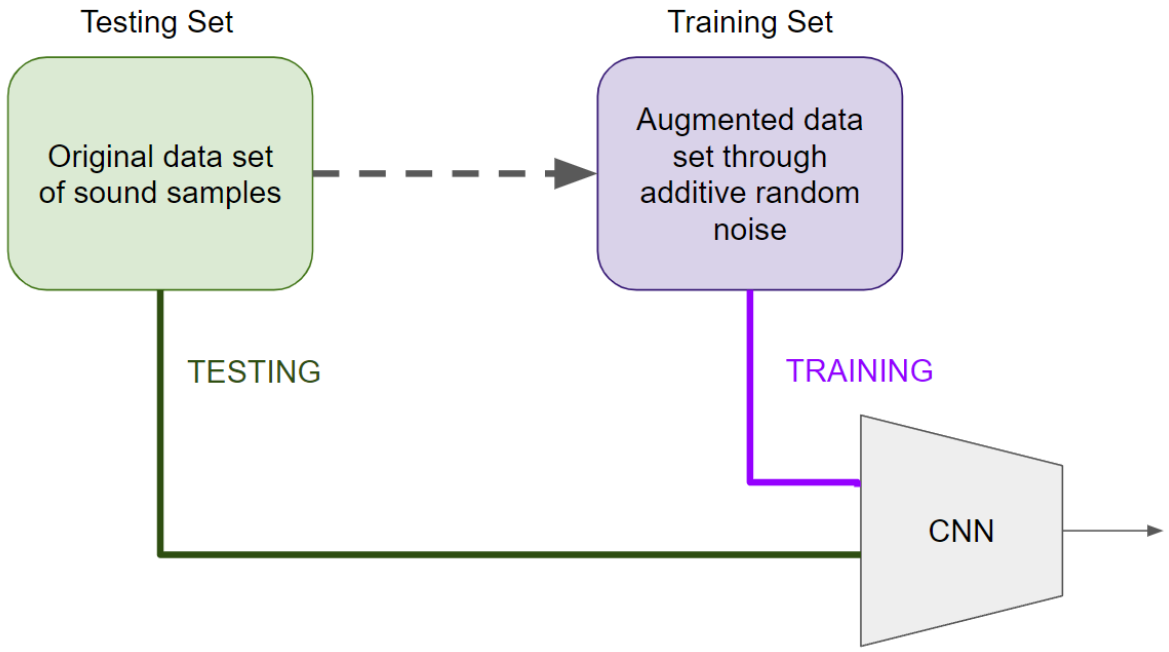


Figure 6.9: Augmentation model architecture for sound sample processing.

In the experiments, data augmentation created two parallel datasets: one for training purely based on augmented data and another for testing with the original samples. The architecture can be viewed in Figure 6.9. Once trained, the model returned hopeful results with a Mean Squared Error close to 178.95, a R^2 score of 0.9835, and a Mean Absolute Error of 9.80. The graphical representation further reinforced the model's efficacy. As seen in Figure 6.10, the agglomeration of data points around the red line, meaning an ideal prediction, shows the model's successful results.

The extreme improvements in the model's performance were too impressive, so much so that it became a red flag that something was wrong. Data augmentation comes with its own set of issues, such as the model's tendency to overfit to the variations of the augmented data. The added augmentation artifacts can become the primary learning target. The model could fixate on the random augmentations within waveforms rather than finding patterns in the underlying MFCC. Thus, it is important to verify this possibility.

The original split between the training and testing datasets can be viewed in Figure 6.9; the architecture depicts a two-step process of handling sound samples segmented into training and testing sets. The

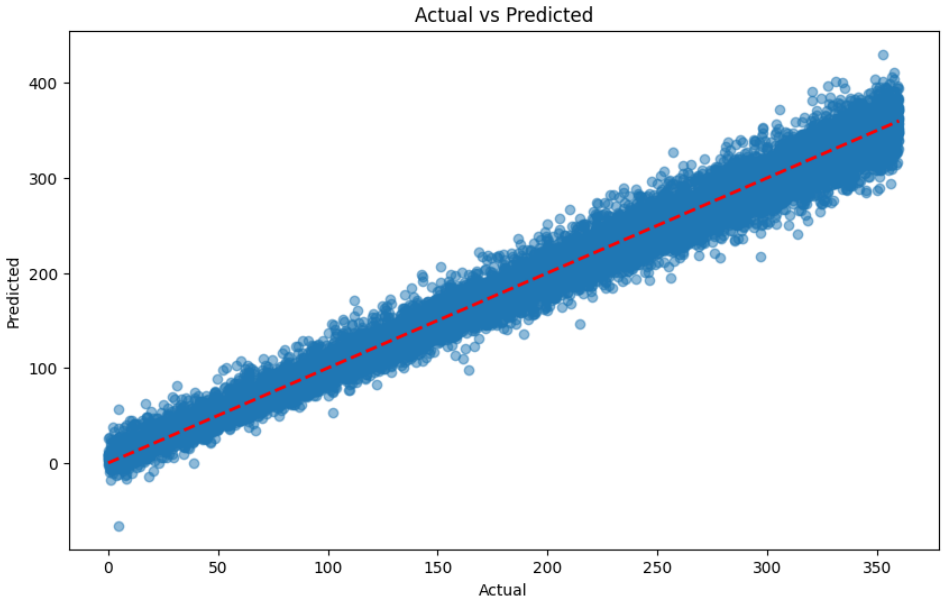


Figure 6.10: Actual vs Predicted values scatter plot with ten-time date augmentation

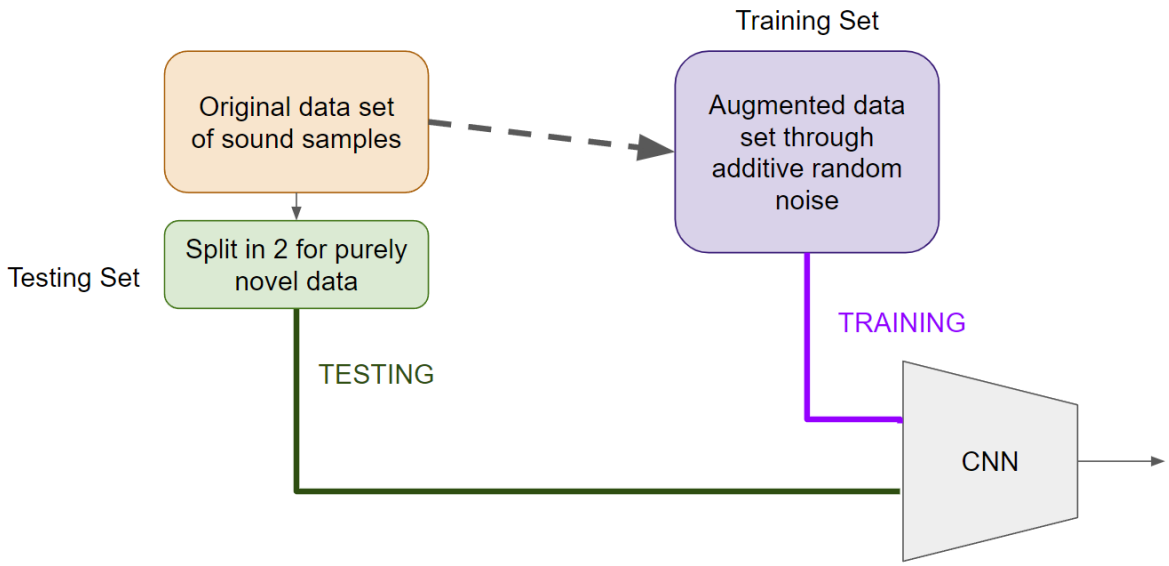


Figure 6.11: Revised augmentation model architecture for sound sample processing

model employs a training set of augmented datasets created by introducing additive random noise to the original sound samples. The augmented dataset is then fed into the CNN for training. Unlike the training set, the testing set uses the original sound samples without augmentation. This approach tests its ability to recognize patterns in unchanged data based on learnings from the augmented dataset. A new model training and testing architecture was used to verify that the model does not overfit on the random additions, shown in Figure 6.11. In this new architecture, the original data was split into two; the testing data is still unseen data points. However, the training data is only augmented data points, which is not in the testing set. This is done not to have any data augmentation overlap in the training and testing sets, thereby assessing its true capability to generalize.

The revised architecture, from figure 6.11, aimed to address concerns from the previous setup. Unfortunately, the results from this architecture echoed our apprehensions. Indeed, the trained model did not

perform well on previously unseen data, as seen in the graphical representation of results in Figure 6.12. Consequently, it can be said that data augmentation did not improve the results.

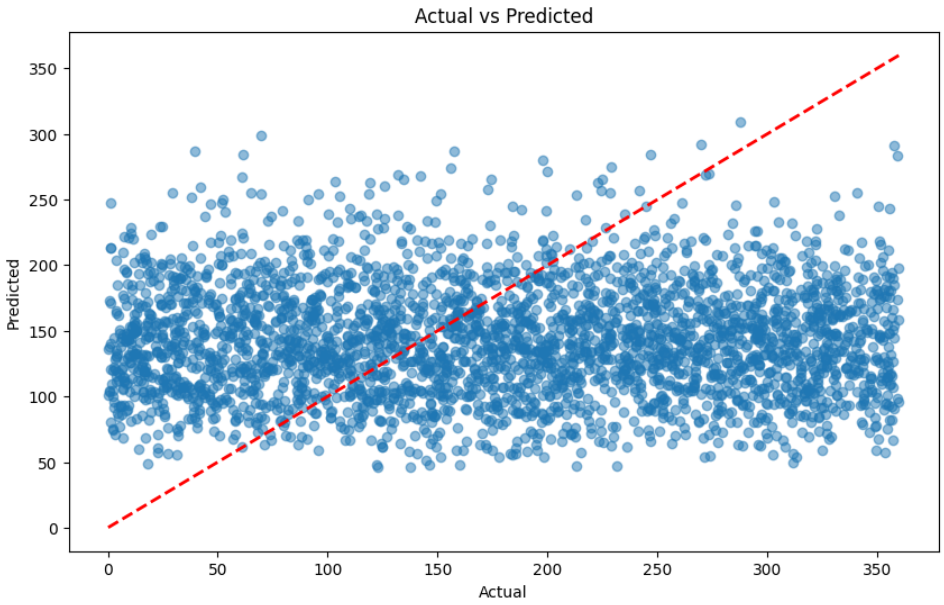


Figure 6.12: Actual vs Predicted values scatter plot Revised Data Augmentation architecture

The series of regression experiments conducted using a single microphone yielded insights into the possibility of accurately predicting continuous mouse movement angles. A single microphone was shown to be insufficient to regress the angle of movement based on MFCC. This shortfall points to the complicated nature of audio processing, where single-dimensional data may fall short of capturing the complexities of two-dimensional mouse movements. Despite this, the research refined the experimental framework. The knowledge gained from these trials will be instrumental in future research. Secondly, this exploration served as an experimental cul-de-sac, steering the research focus toward other interesting research avenues.

6.3. Dual-Microphone Approach for Two-Dimensional Amplitude Analysis

Building on the findings from section 5.3, where audio amplitude analysis proved effective in determining mouse movement proximity along a single axis, this section advances the research by introducing a second microphone. This addition aims to enhance the granularity of our measurements, exploring the limits of precision in mouse motion inference. Adding a second microphone enables the creation of a two-dimensional measurement plane, similar to a mouse pad's coordinate system. The experiments presented here used two synchronized smartphone microphones on the side of the mouse pad, as outlined in section 4.2. The following analysis seeks to understand the extent to which a two-microphone configuration can improve the accuracy of the mouse's acoustic leakage model for SCA.

The recording movement pattern involved traversing the length of the table, moving from top to bottom and back, and then horizontally from near the phone to the far edge and back, as seen in Figure 6.13. The graphs' visual analysis of amplitude differences and trends reveals interesting insights. The graphs in Figure 6.14 show a plot of each microphone amplitude and the amplitude difference. In addition, regression lines for the microphone amplitudes are plotted to show general trends. As can be seen in Figures 6.14a and 6.14b, representing moving up and down the table, a criss-cross effect is displayed, indicating a microphone "hearing" louder than the other at certain times. It is possible to determine the movement direction by understanding which graph represents which microphone. Furthermore, Figures 6.14c and 6.14d show similar microphone amplitude differences. The variations, however, are

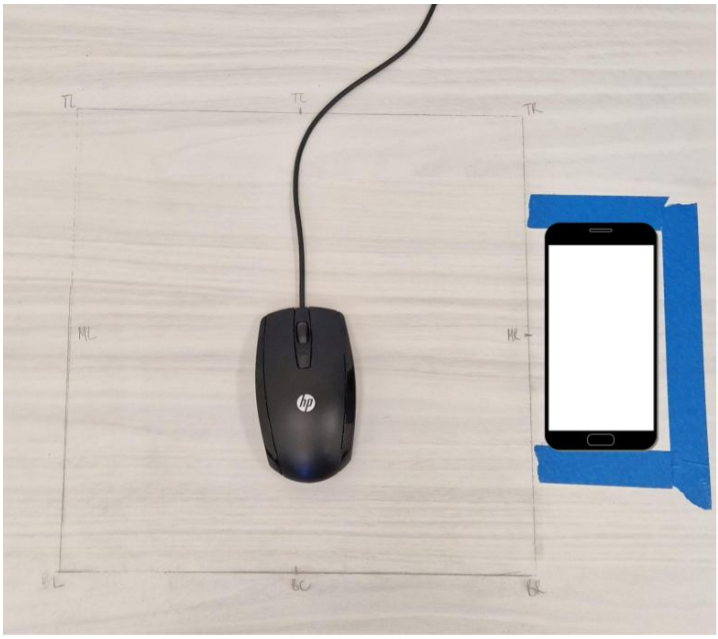


Figure 6.13: Dual-Microphone experimental setup

indicated by a negative or positive slope trend in the regression line depending on moving towards or away from the microphone. These plots highlight the potential to visually tell the difference between the four cardinal movements, setting the stage for a model to tell the movement differences based on audio features.

The success of the experiments, which employed one microphone, suggests the feasibility of detecting two-dimensional directional movements of a computer mouse. The ability to accurately infer directional shifts—from bottom-left to top-right on a sizable mouse pad—demonstrates that even the subtle sounds produced by mouse movements can be captured and potentially interpreted by attackers. The next experiment attempted to increase the granularity of the attack by attempting to distinguish 12 movement patterns. The following list will enumerate the movements and the hypothesis of the outcomes on the visual graphs:

1. **MIDDLE RIGHT to TOP RIGHT**: The top microphone (Amplitude 0) is expected to experience an amplitude increase, asserting acoustic dominance as the sound source moves closer. In contrast, the bottom microphone (Amplitude 1) will likely record a decrease in amplitude, becoming overshadowed by the end of the movement. The differential amplitude (Diff Amplitude) is projected to rise linearly.
    - **Result Graph:** Figure 6.15a
2. **TOP RIGHT to BOTTOM RIGHT**: The top microphone's amplitude is predicted to decrease, indicating a diminishing presence as the sound source moves away. In contrast, the bottom microphone's amplitude is expected to increase, taking on a dominant acoustic role by the end. The amplitude difference should decrease linearly.
    - **Result Graph:** Figure 6.15b
3. **BOTTOM RIGHT to TOP RIGHT**: An increase in amplitude is expected for the top microphone, becoming the predominant acoustic source, whereas the bottom microphone should see a decrease, indicating a lesser acoustic impact. The differential amplitude is anticipated to increase linearly.
    - **Result Graph:** Figure 6.15c
4. **TOP RIGHT to TOP LEFT**: Both microphones are anticipated to register a decrease in amplitude; however, the top microphone will likely remain louder than the bottom. The differential amplitude

should remain positive and display a flat, straight line.

- **Result Graph:** Figure 6.15d

5. **TOP LEFT to TOP RIGHT**: An increase in amplitude is expected for both microphones, with the top microphone maintaining a louder presence. The differential amplitude should stay positive, following a straight-line trajectory.
   - **Result Graph:** Figure 6.15e
6. **TOP RIGHT to MIDDLE RIGHT**: The top microphone's amplitude is forecasted to decrease, reaching equilibrium with the bottom microphone, which, conversely, is expected to increase. The differential amplitude should decrease linearly, approaching and remaining near zero.
   - **Result Graph:** Figure 6.15f
7. **MIDDLE RIGHT to MIDDLE LEFT**: Both microphones are expected to see a decrease in amplitude, reaching an equilibrium state. The differential amplitude is predicted to remain a straight line, hovering near zero value.
   - **Result Graph:** Figure 6.15g
8. **MIDDLE LEFT to MIDDLE RIGHT**: An increase in amplitude for both microphones is anticipated, maintaining equilibrium. The differential amplitude is expected to be a straight line, remaining near zero value.
   - **Result Graph:** Figure 6.15h
9. **MIDDLE RIGHT to BOTTOM RIGHT**: The top microphone's amplitude is predicted to decrease, becoming overpowered by the end, while the bottom microphone's amplitude should increase, becoming the dominant acoustic force. The differential amplitude is projected to decrease linearly to a negative value.
   - **Result Graph:** Figure 6.15i
10. **BOTTOM RIGHT to BOTTOM LEFT**: A decrease in amplitude for the top microphone is anticipated, where it will be overshadowed by the bottom microphone, which is also expected to decrease but maintain a louder amplitude than the top. The differential amplitude should be a straight line with a negative value.
    - **Result Graph:** Figure 6.15j
11. **BOTTOM LEFT to BOTTOM RIGHT**: The top microphone is expected to increase in amplitude but still be overshadowed by the bottom microphone, which will also increase and remain the louder source. The differential amplitude should present as a straight line with a negative value.
    - **Result Graph:** Figure 6.15k
12. **BOTTOM RIGHT to MIDDLE RIGHT**: An increase in amplitude is expected for the top microphone, reaching an equilibrium with the bottom microphone, which should decrease in amplitude. The differential amplitude is anticipated to increase, approaching a zero value.
    - **Result Graph:** Figure 6.15l

The graphical representations of the experimental results are delineated in Figure 6.15. The experimental outcomes fitting the predictions across all measurements support the premise that two-dimensional movements can be discerned with a dual-microphone configuration. It is important to note that there are a few inconsistencies due to the large distances taken while performing the mouse movements.

The experiments have yielded visual representations that clearly distinguish various directional movements on the mouse pad. These consistent and discernible patterns suggest that a machine learning algorithm could be trained to recognize these movements. Thus, ten movements—top-right to bottom-right, middle-right to middle-left, top-right to top-left, bottom-right to bottom-left, and top-central to bottom-central—will be recorded and segmented using Audacity. Approximately 250 samples will be captured for each directional shift, culminating in a dataset encompassing ten categories.

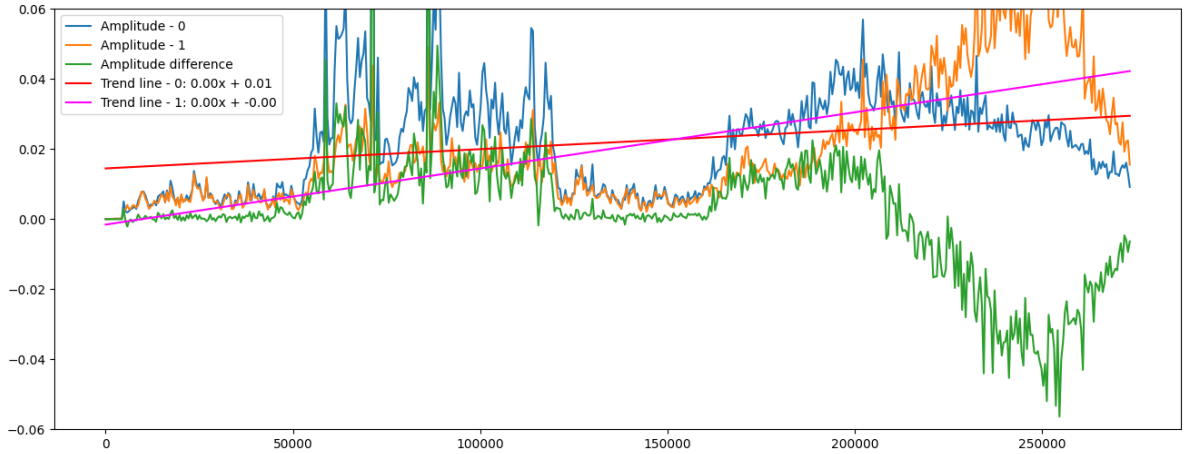
Class	Precision	Recall	F1-Score	Support
BC->TC	0.91	0.94	0.92	52
BL->BR	1.00	1.00	1.00	50
BR->BL	1.00	0.96	0.98	50
BR->TR	1.00	0.97	0.99	39
ML->MR	0.83	0.88	0.85	49
MR->ML	0.88	0.86	0.87	49
TC->BC	0.98	0.98	0.98	52
TL->TR	0.98	1.00	0.99	49
TR->BR	0.97	0.92	0.95	39
TR->TL	1.00	1.00	1.00	49
		Accuracy	0.95	478

Table 6.3: Classification report: Machine Learning on ten movements with dual-microphone setup

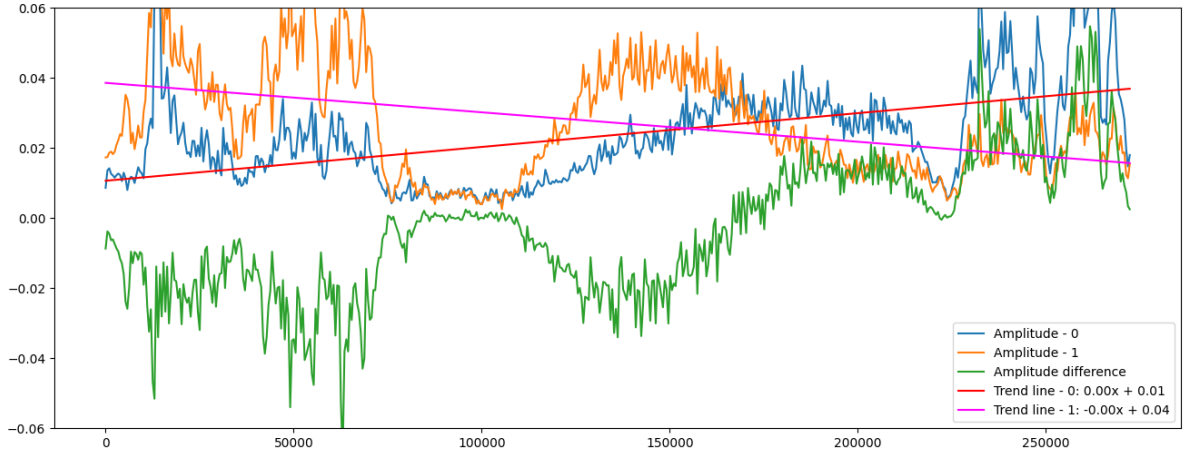
Implementing a machine learning model for classifying mouse pad movements has successful results. Utilizing a Random Forest Classifier with 100 estimators, the model achieved high accuracy. The test accuracy was recorded at approximately 95.19%, indicating a good performance across various movement categories. The classification report in Table 6.3 provides further insights into the model's efficacy. Overall, the model displayed exceptional classification capabilities with an accuracy of 95% across all categories. The high precision and recall values across the board underscore the model's ability to recognize and distinguish between different two-dimensional movements. This experiment works on mouse pads as small as 7×7 inch^2. This experiment demonstrated the ability to infer ten distinct two-dimensional movements, exposing the capability to understand the leakage model of a computer mouse using a smartphone.

6.4. Conclusion: Assessing the Limits of Acoustic Mouse Movement Inference

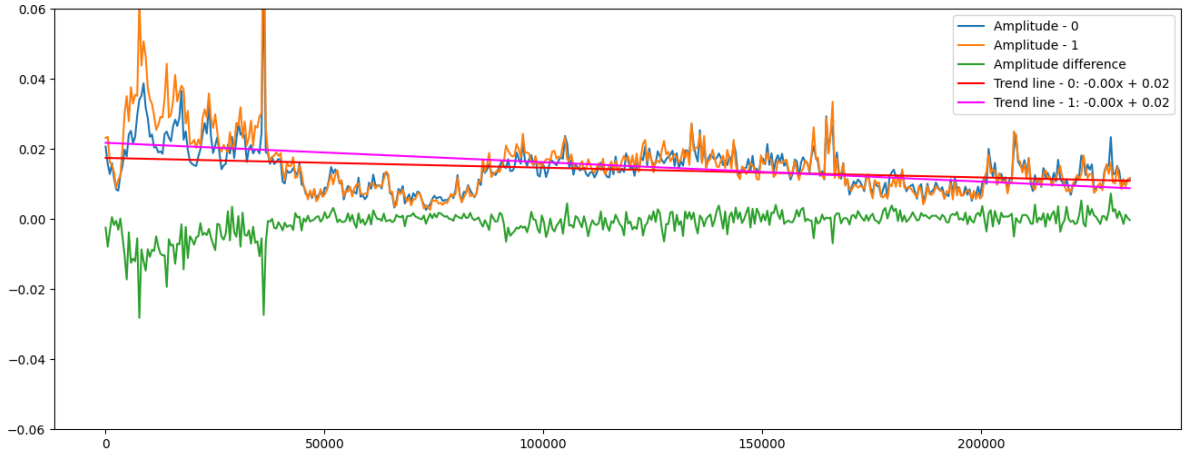
This chapter showed an exploratory journey to determine the precision achievable in mouse movement inference using audio. Our investigation initially focused on a single microphone setup, but it transitioned to a dual-microphone approach due to limited success. Initially, we attempted to enhance directional classification by incorporating more movement categories. However, this approach reached a limited effectiveness when using only one microphone. Subsequently, we explored the potential of regression algorithms for predicting continuous movement angles, hypothesizing that this method might offer a more nuanced understanding of the correlation between acoustic signals and mouse motions. Unfortunately, this technique also failed to establish a consistent and reliable correlation. The final phase of our research involved employing two microphones in conjunction with amplitude analysis. This technique marked a significant breakthrough, with the model successfully discerning between 10 distinct movements on the mouse pad with an accuracy of 95%. In conclusion, while single microphone approaches faced limitations, using a second microphone from a smartphone significantly enhanced our ability to track and interpret mouse movements through acoustic analysis accurately. This advancement in methodology demonstrates the potential for acoustic tracking in practical applications, setting a foundation for security risks posed by computer mice.



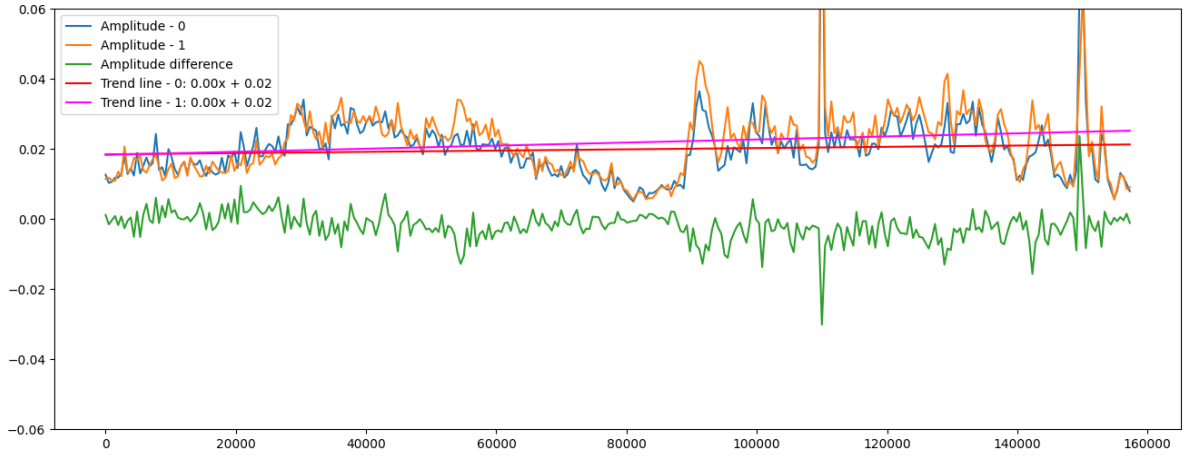
(a) Amplitude difference and trend when moving vertically down the table



(b) Amplitude difference and trend when moving vertically up the table

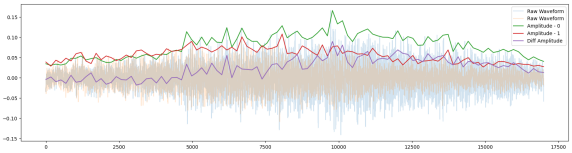


(c) Amplitude trend with a negative slope, indicating movement away from the phone

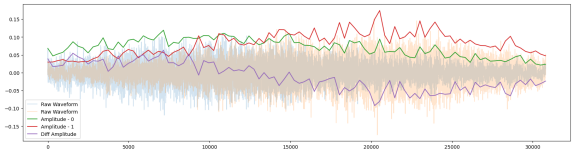


(d) Amplitude trend with a positive slope, indicating movement towards the phone

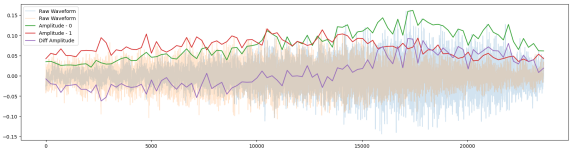
Figure 6.14: Graphs showing amplitude differences and trends based on the mouse's movement



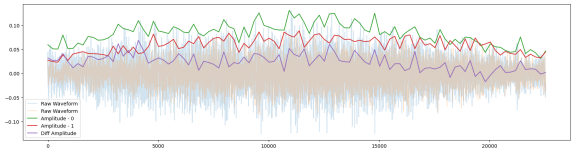
(a) MIDDLE RIGHT to TOP RIGHT



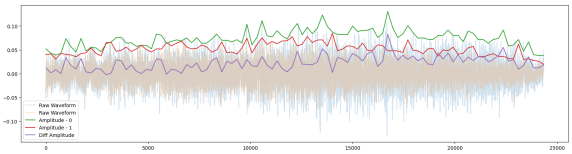
(b) TOP RIGHT to BOTTOM RIGHT



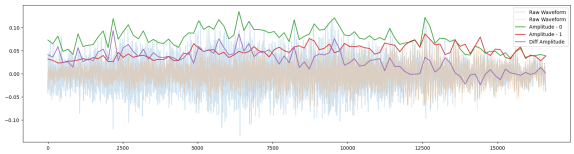
(c) BOTTOM RIGHT to TOP RIGHT



(d) TOP RIGHT to TOP LEFT

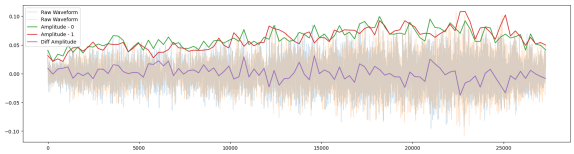


(e) TOP LEFT to TOP RIGHT

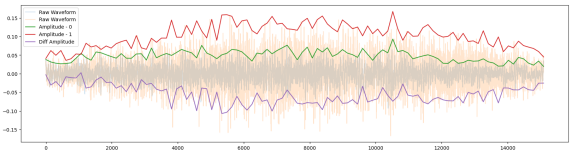


(f) TOP RIGHT to MIDDLE RIGHT

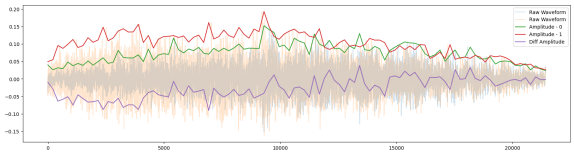
(g) MIDDLE RIGHT to MIDDLE LEFT



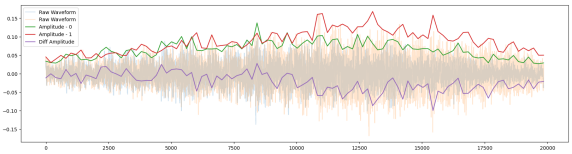
(h) MIDDLE LEFT to MIDDLE RIGHT



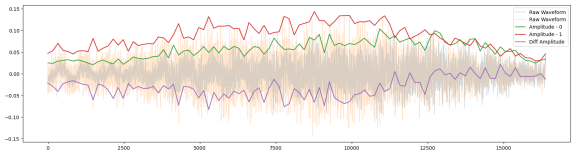
(i) MIDDLE RIGHT to BOTTOM RIGHT



(j) BOTTOM RIGHT to BOTTOM LEFT



(k) BOTTOM LEFT to BOTTOM RIGHT



(l) BOTTOM RIGHT to MIDDLE RIGHT

Figure 6.15: Experimenting with different movements using a dual-microphone setup

7

Real-world Implications and Security Risks

7.1. Experiment with Other Participants

In Section 6.3, our experiments demonstrated the potential to track two-dimensional mouse movements on a mouse pad. Based on data from a single participant, these findings show the ability to tailor attacks to an individual victim. This research is interesting as it proves the ability of side-channel attacks on mouse movements. To further validate and understand the scope of this method, the next phase of our research involves extending the experiment to include a larger range of participants. This will help determine the generalizability of the model and its applicability in various environments.

Using the same setup as previous tests, the new experiment involved six participants from the UCI SPROUT lab performing a series of mouse movements on a mouse pad. The experiment setup can be seen in Figure 6.13 and will be used to record data. We developed a protocol where each session lasted between 6 and 12 minutes, a duration chosen to balance the amount of data collected with the recording time per participant. Throughout the exploratory research in this thesis, a significant challenge encountered was the extensive time and resources required to build a dataset through many recording sessions. However, the shorter duration of the experiments allowed greater participation but also recorded enough data for the machine learning model.

Participants were instructed to perform distinct back-and-forth movements along various axes within a 12-inch by 12-inch mouse pad. Recording distinct axes is important for the SCA analysis research as it produces unique audio signals. The experiment detailed specific movement patterns, seen in Figure 7.1: from bottom-right (BR) to top-right (TR), bottom-right (BR) to bottom-left (BL), bottom-left (BL) to top-left (TL), top-left (TL) to top-right (TR), top-left (TL) to bottom-right (BR), and bottom-left (BL) to top-right (TR). Participants were requested to repeat each back-and-forth movement for one to two minutes, resulting in a total experiment time ranging from 6 to 12 minutes. To help with the segmentation during the preprocessing phase, participants were also asked to clap before and after recording each axis as an audio recorder. The six participants showed variability in their mouse usage methods during the experiment. For example, some participants executed the movements rapidly, resulting in a higher frequency of actions, while others moved more slowly, taking more time for each motion. In addition, the amount of pressure applied to the mouse differed among participants, producing distinct acoustic signatures. These speed, frequency, and pressure differences were integral to the experiment's design. They provided a diverse dataset that was ideal for testing the generalizability of the machine learning model, as it reflected a more realistic range of user behaviors.

The experiment used a smartphone's top and bottom microphones to record mouse movements

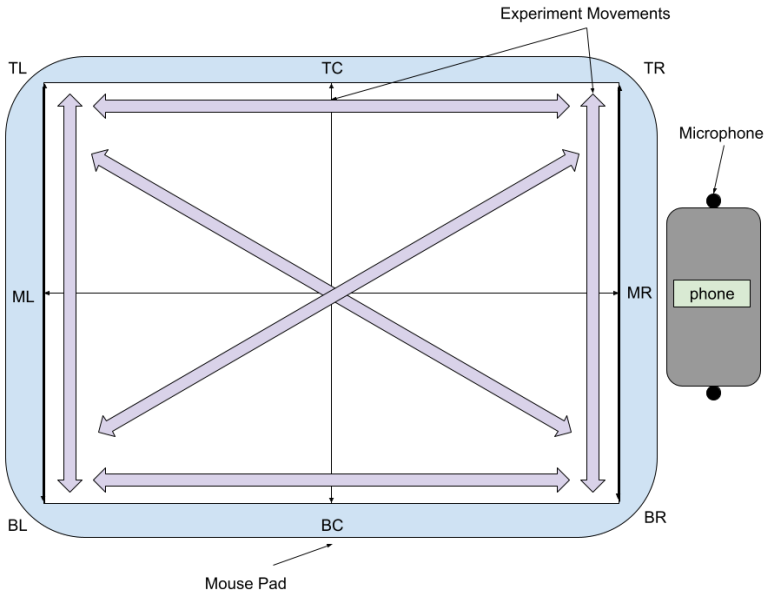


Figure 7.1: Mouse pad with the example mouse movements in the experiment

in stereo recording. These recordings were then imported into Audacity[1], a digital audio editing software, for processing. Within Audacity, we specifically targeted the segments corresponding to mouse movements. These segments were isolated using a thresholding technique to distinguish them from background noise, followed by a manual verification process to ensure accuracy. This method extracted only the relevant sound frames representing the mouse movements, as shown in Figure 7.2. These extracted segments were then exported and analyzed in a Python Jupyter notebook. The waveforms from both channels were split into 50 discrete sections for each sample. The amplitude for each of the 50 sections was calculated. Thus, we extracted the same amount of data points for each movement acoustic signal regardless of the different recording methods used by the participants.

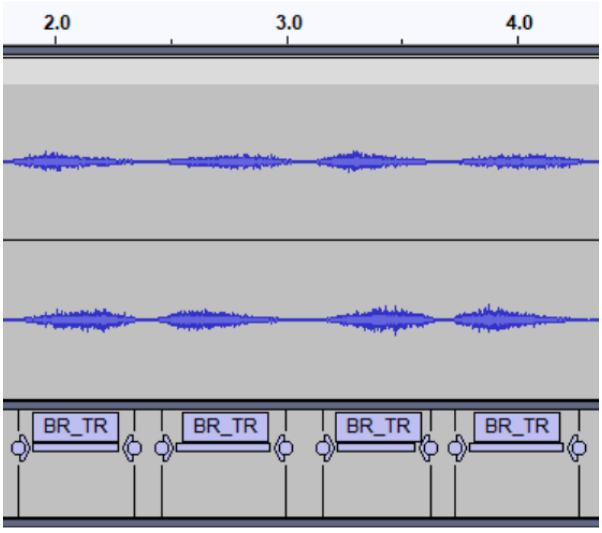


Figure 7.2: Audacity labeling mouse movements

Our analysis further involved calculating the difference in amplitudes between the top and bottom microphone recordings for each segment. Consequently, we generated a 'difference amplitude line' of

[1]https://www.audacityteam.org/

50 data points representing the differential amplitude for each audio sample. In our machine learning model, these data points served as the learning values, the 'X' values. The 'Y' values, or target labels, were assigned based on the type of directional movement recorded. This structured dataset, with its distinct X and Y values, allowed us to train and test the machine learning model to classify the directional movements based on acoustic signatures.

Our experimental setup was designed to record six types of back-and-forth mouse movements, resulting in 12 distinct directional classes. In the course of our experiments, a total of 5507 samples were collected. The distribution of these samples across the different classes is as follows:

- TL → TR: 506 samples
- TR → TL: 504 samples
- BL → TL: 483 samples
- TL → BL: 479 samples
- TL → BR: 459 samples
- BL → TR: 456 samples
- BR → TL: 456 samples
- TR → BL: 453 samples
- BR → TR: 447 samples
- TR → BR: 447 samples
- BR → BL: 411 samples
- BL → BR: 406 samples

The machine learning pipeline is outlined in Listing 7.1. It involves preparing the dataset, splitting it into training and testing sets, training a Random Forest classifier, and evaluating its performance. The dataset is first converted into numpy arrays for the features ('X') and labels ('y'). It is then split into training and testing sets, with 20% of the data reserved for testing. The `RandomForestClassifier` from Scikit-Learn is used with 100 estimators. After training the classifier on the training set, it is used to make predictions on the test set. Finally, the accuracy and detailed classification report is printed, providing insights into the model's performance.

```
X = np.array([x['diff_amplitude'] for x in dataset])
y = np.array([x['category'] for x in dataset])
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, stratify=y)
clf = RandomForestClassifier(n_estimators=100)
clf.fit(X_train, y_train)
y_pred_test = clf.predict(X_test)

print("Test Accuracy: ", accuracy_score(y_test, y_pred_test))
print("Classification Report: \n", classification_report(y_test, y_pred_test))
```

Listing 7.1: Machine learning pipeline

As detailed in Table 7.1, the classification report evaluates the performance of the machine learning model. The report is segmented by the twelve distinct directional movements used in the experiment, namely BL->BR, BL->TL, and so forth. For each directional category, the precision, recall, and F1-score are listed alongside the number of samples (support) used for that category. The report indicates high accuracy in the model's predictions, with a low rate of false positives. The F1-scores, which combine precision and recall into a single measure, consistently reflect high performance, predominantly in the range of 0.93 to 1.00. This underscores the model's balanced accuracy in both precision and recall dimensions. The overall test accuracy of the model is 96%, indicating the model's effectiveness in classifying the directional movements accurately.

While our initial results were obtained using a Random Forest Classifier with 100 estimators, exploring the model's performance with a reduced count, such as ten estimators, would be interesting. This

Direction	Precision	Recall	F1-Score	Support
BL->BR	0.98	0.99	0.98	81
BL->TL	0.91	0.98	0.95	97
BL->TR	0.96	0.90	0.93	91
BR->BL	0.99	0.93	0.96	82
BR->TL	0.94	0.99	0.96	91
BR->TR	1.00	0.99	0.99	89
TL->BL	0.92	0.97	0.94	96
TL->BR	0.98	0.93	0.96	92
TL->TR	0.99	1.00	1.00	101
TR->BL	0.93	0.95	0.94	91
TR->BR	0.98	0.93	0.95	90
TR->TL	0.99	0.99	0.99	101
		Accuracy	0.96	1102

Table 7.1: Classification Report for a Random Forest Classifier with 100 estimators

reduction allows us to evaluate the model's efficiency under computational constraints and its ability to maintain accuracy with fewer decision trees, thus assessing its stability and generalizability.

Direction	Precision	Recall	F1-Score	Support
BL->BR	0.97	0.96	0.97	81
BL->TL	0.89	0.88	0.88	97
BL->TR	0.89	0.91	0.90	91
BR->BL	0.99	0.96	0.98	82
BR->TL	0.96	0.98	0.97	91
BR->TR	1.00	0.97	0.98	89
TL->BL	0.88	0.90	0.89	96
TL->BR	0.95	0.97	0.96	92
TL->TR	0.97	0.98	0.98	101
TR->BL	0.92	0.91	0.92	91
TR->BR	0.98	0.91	0.94	90
TR->TL	0.95	1.00	0.98	101
		Accuracy	0.94	1102

Table 7.2: Classification Report for a Random Forest Classifier with ten estimators

The classification report for the Random Forest Classifier with ten estimators, detailed in Table 7.2, presents an overview of the model's performance. Despite the reduction in estimators, the model exhibits high precision across all categories, with scores predominantly above 0.88. The F1 scores, which balance precision and recall, remain high, affirming the model's robustness even with fewer estimators. The overall accuracy of 94% underscores the model's efficiency and reliability in classification tasks.

This section allowed us to show the generalizability of a machine learning model to distinguish between 12 movements along a mouse pad. With an average of 95% accuracy, it shows that a model can be derived to infer different mouse movements in a detailed environment. It is interesting research as a generalizable model was applied to multiple participants.

7.2. Inferring Realistic Mouse Movements

This section explores a real-world case study where acoustic signals from mouse movements are used to extract user information. Previous research in this thesis has been primarily exploratory, focusing on the extent of the leakage model rather than the realism of user mouse movements. In this section, we shift towards a more practical approach, experimenting to realistically detect a victim's mouse movement

activity through sound. This will demonstrate the feasibility of such attacks and also help in assessing their potential impact in real-world scenarios.

The next experiment aims to determine whether it is possible to detect when a user clicks the 'close' button, typically found at the top right of a Windows laptop screen. It offers a practical and accessible method for experimentation while demonstrating the potential for leakage of user activity information. While closing a window may not directly reveal sensitive security information, the ability to discern precise user interactions like button clicks could imply broader implications for user privacy. In particular, this experiment underscores how, given enough contextual knowledge, clicking a specific button on the screen could become a vector for sensitive information leakage. Analyzing a sequence of user button clicks makes it possible to reconstruct a user's activities on an online form or website. This scenario is a security vulnerability, allowing an attacker to create a timeline of user events. In the context of our experiment, detecting the act of closing a window becomes particularly revealing. It signifies a change in the user's focus or the end of a specific task. Such information could infer patterns in the user's attention or behavior.

The experimental protocol was structured around two recording sessions of five minutes with a single participant. The experiment was performed on a traditional mouse setting of 5 on a Windows laptop and an entry level office mouse pad. This setup is a realistic setup used on a daily basis. In the first session, the participant was instructed to move the cursor from random points on the screen to the top-right corner and click the red 'X', a standard action for closing a window. The second session involved recording mouse movements followed by clicks at various random points on the screen, a pattern different from the first recording session. In our binary classification model, the first type of movement is designated as the 'positive' class, representing the targeted action we aim to detect. Conversely, the second type of movement is classified as the 'negative' class, contrasting to the distinct action of closing a window.

We will focus on a time window around click events in the analysis phase. This approach allows us to compare the acoustic characteristics of mouse movements and clicking sounds associated with both positive and negative classes. By isolating these sounds, we can train our model to distinguish between the targeted action of closing a window and other random mouse activities.

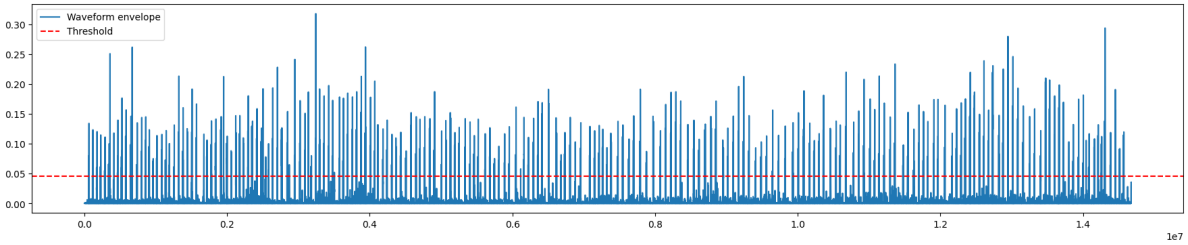


Figure 7.3: Thresholding mouse activities

The recordings of the two sessions are loaded into a Jupyter Notebook, where they are first analyzed to determine the clicking events through thresholding methods. To do so, the waveforms are converted to a mono channel to calculate the absolute value of the waveform. This allows us to highlight the significant peaks, such as mouse clicks. A quantile threshold method is used to isolate the peaks. This threshold can be modified based on the visual representation and manually adjusted, as seen in Figure 7.3. This step attempts to find a threshold method considering as many peaks as possible.

Having found an adequate threshold, we isolate audio samples corresponding to the detected clicking events in the waveform. With these clicking events, the goal will be to find the two consecutive peaks corresponding to the mouse press and release events during a mouse click. To do so, a minimum and maximum distance between detected peaks will be used to isolate the events that seem to correspond to mouse-clicking events. The minimum and maximum distances are again estimated through empirical

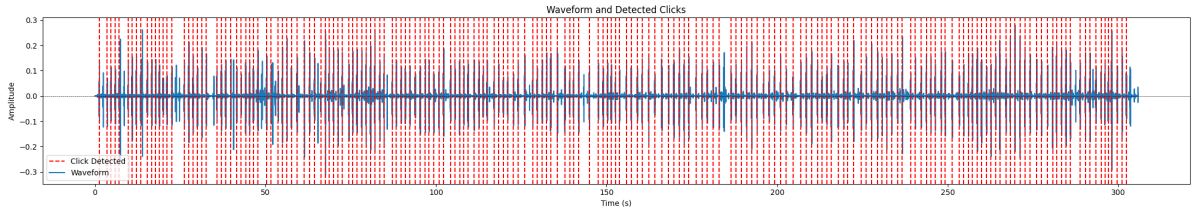
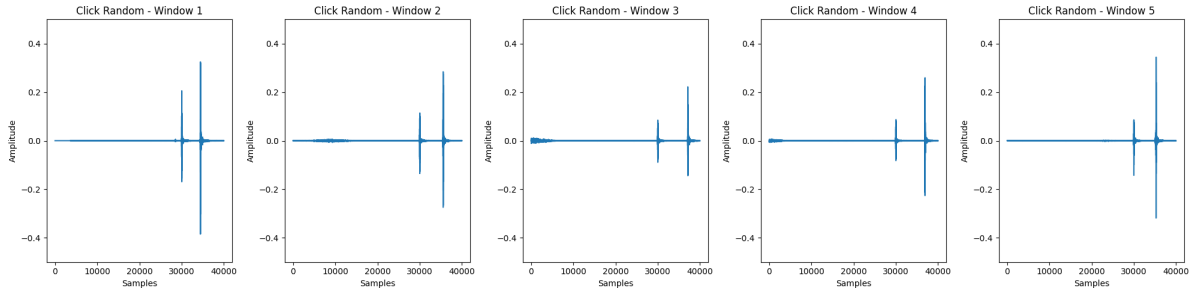
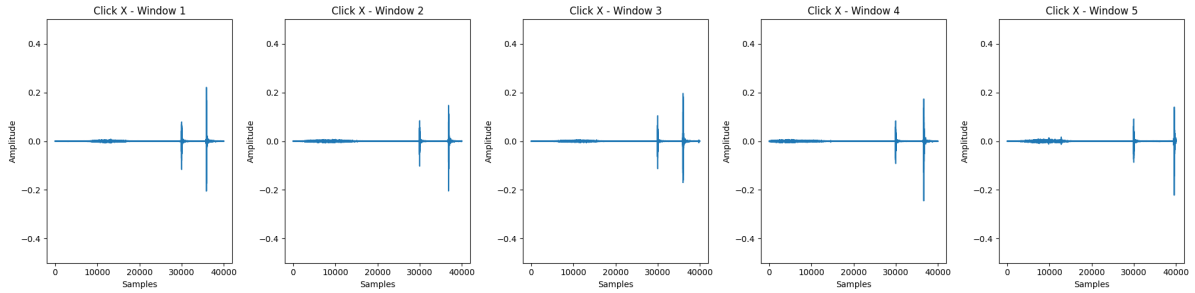


Figure 7.4: Detected clicks out of mouse events

values, in this case from 10ms to 200ms. This algorithm allows the isolation of clicking events with decent precision. Figure 7.4 shows the waveform with red vertical lines representing the detected clicks.



(a) Windows around mouse clicking randomly



(b) Windows around mouse clicking X

Figure 7.5: Windows around mouse click events

As we iterate through each identified mouse click event, a specific time window surrounding each event is extracted to prepare the data for the machine learning pipeline. We ensure these windows are uniformly sized to maintain consistency across all mouse clicks, as depicted in Figure 7.5. Next, an MFCC transformation is applied to each window. These transformed data points are fed into a machine learning pipeline utilizing a Random Forest Classifier with 50 estimators. The classification task is binary: determining whether the audio frame window corresponds to a click on the red 'X' (closing event) or a random click elsewhere.

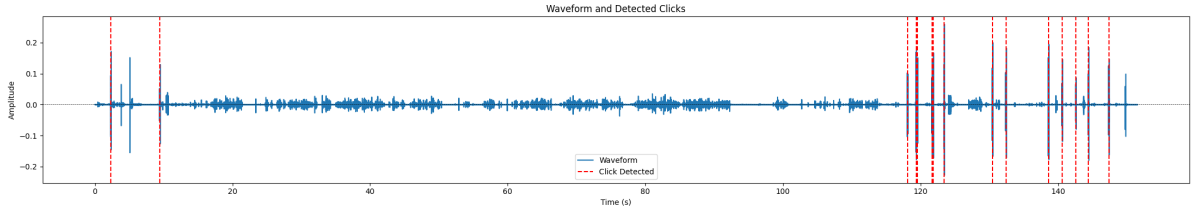
Class	Precision	Recall	F1-Score	Support
0	0.89	0.96	0.92	77
1	0.94	0.85	0.89	59
		Accuracy	0.91	136

Table 7.3: Classification Report for the Binary Classification Model

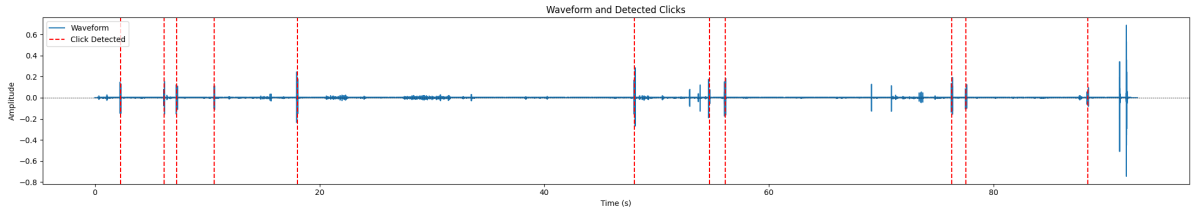
The classification report, as illustrated in Table 7.3, reflects the performance of the binary classification model. The model demonstrates a high level F1-score, with class '0' reaching 0.92 and class '1' at 0.89. These scores indicate a strong balance between precision and recall across both classes. The overall

accuracy of the model stands at 91% accuracy for the 136 samples tested, further reinforcing the model's effectiveness.

The experiment outlined in this section successfully demonstrated the ability to distinguish between various mouse movements and their associated mouse clicks. Using controlled recordings, we trained a random forest classifier to recognize these specific activities. The next phase of our study aimed to extend this approach to more naturalistic computer usage scenarios. In the follow-up experiment, we recorded the author's typical computer activities, including working on this thesis in Overleaf, ending with the action of closing the window. These recordings were then processed using the algorithmic steps previously described. The pre-trained random forest classifier was then applied to the isolated mouse click events to infer the type and nature of the mouse movements and clicks. This step marks a significant advancement in applying our model to real-world scenarios, bridging the gap between controlled experimental conditions and everyday computer usage.



(a) First experiment recording



(b) Second experiment recording

Figure 7.6: Comparison of waveform detections in two different iterations of the experiment

We experimented with two iterations. In the first iteration, the red 'X' was clicked to close the window, and two additional clicks were made. However, due to the thresholding method employed, the final click was not detected, as illustrated in Figure 7.6a. In the second iteration, the red 'X' to close the window was clicked and followed with two hand claps. This means that the final detected mouse click is the one of interest, demonstrated in Figure 7.6b. In the waveforms, we can observe background audio that includes random clicks and typing, creating a consistent hum of ambient noise.

In the first recording, among the 15 clicks detected, the before-last sample was accurately identified as corresponding to the 'clicking red X' event. The prediction output for this sequence was [0,0,0,0,0,0,0,0,0,0,0,0,0,1,0], confirming the model's precision. Similarly, in the second recording, out of 11 notable events, the model correctly inferred the last one as the 'click red X' event, with the prediction array being [0,0,0,0,0,0,0,0,0,0,1]. These results demonstrate the trained model's ability to accurately identify window-closing events, suggesting the feasibility of employing such a method in an acoustic side-channel attack to target victims.

7.3. Security Risks of Acoustic Side-Channel Attacks on Mouse Movement Inference

This chapter has presented a series of experiments highlighting the security implications of acoustic side-channel attacks in mouse movement detection. Our research successfully used smartphone microphones to train a model capable of distinguishing between 12 different mouse movement operations. More importantly, we showcased a practical attack scenario where the model could infer user mouse movement

actions, such as closing a window on a Windows laptop. Thus demonstrating the potential security threats posed by acoustic SCAs. The findings illustrate the feasibility of developing a generalizable model applicable to various users and the capability to accurately identify specific mouse movements in realistic settings. Looking ahead, these insights open the door to potential future threats where acoustic SCAs might be combined with keyboard inference techniques, further reconstructing a victim's computer activities. For instance, one plausible scenario involves targeting users filling out sensitive online forms, such as tax documents. In such cases, the model could be fine-tuned to recognize distinctive patterns associated with clicking specific buttons or entering information into text fields. Moreover, combined with existing research in keyboard SCAs, this method could be expanded to deduce the information being entered. Such a threat highlights the importance of ongoing research into acoustic SCAs.

8

Conclusion

This research aimed to explore the possibility of acoustic side-channel attacks on a computer mouse and whether the sound emitted during mouse movements reveals any sensitive information that can pose a security risk. This research answered that inquiry by asking three underlying research questions, which attempted to examine the extent of the acoustic leakage model incrementally. The first research question was: *can the sound produced by a computer mouse movement be used to infer its motion path?* Indeed, experiments were established which proved the ability to infer basic mouse movements. Through several experimental setups, we have shown that moving mice in up, down, left, and right motions produce sounds that can be interpreted for movement predictions. Moreover, in a single axis, the amplitude of the emitted sounds during mouse movements can inform the direction of movement (towards or away from the microphone). Having established proof of concept experiments, we answered: *what is the accuracy with which we can infer mouse movements from the audio data?* This led to research into granular measurements to predict the direction angles of moving mice, which showed some predictive complexities. Multiple attempts to increase the number of directions from four to eight and run a regression prediction algorithm for movement angles yielded unsuccessful results, which limited the research. The interpreted limit blames using a single microphone, which did not provide enough precise sound measurements. However, an extension to using a stereo recording method through the two microphones present on a smartphone allowed one to distinguish between ten distinctive two-dimensional movements on a mouse pad. Thus, the result shows that in a dual-microphone setup, it is possible to have a leakage model predicting various mouse movements on a pad. The previous experiments were performed in a controlled environment, and it thus became important to answer the question: *what are potential real-world scenarios or environments where acoustic side-channel attacks on mouse movements pose a security risk?* This last question was answered by training a model on six participants capable of predicting 12 distinct two-dimensional movements. This has shown the ability to generalize the model and attack multiple users, considering one recording environment. Furthermore, an experiment that could detect a specific user action, such as closing a Windows screen, was successful in a realistic experimental setup. Consequently, following the described outcomes, we can assure that the acoustic leakage model of the computer mouse poses a security risk in a real-world scenario.

This research dove into a gap in acoustic SCA research targeting human interface devices. It proposed methodologies and experiments to evaluate how susceptible computer mice are to attacks based on the audio they emit during their use. Multiple experiments were conducted, ranging from specialized recording with a single microphone to stereo recording using a smartphone. The envisioned scenarios were initiated as simplistic movements to test the basic theory behind acoustic SCAs. Through regression and large classification models, the attacks attempted to become more granular to predict precise angles of movements. This research direction proved unsuccessful, most likely due to the limitations in spatial awareness posed by a single microphone. It then evolved into experiments using a dual-microphone setup, which showed a greater attack potential; thus, recording mouse movements in stereo poses a real-world security risk on computer mice. This thesis contributes to the body of knowledge by exposing

preliminary research into mouse movement SCAs. By analyzing the limits to using one microphone for the attack, this thesis has shown that to infer mouse movements, it is most interesting to use a dual-microphone setup. Furthermore, this approaches an envisioned attacker model where the attacker would place a smartphone in the environment to eavesdrop on a victim's sensitive information. This research has contributed to academics by demonstrating experimental methodologies which can be used for mouse and audio synchronization as well as machine learning for SCAs.

As this research was exploratory, the results opened the door to potential future works.

The angle regression algorithms analyzed in Chapter 6 are an interesting path to predict the direction vectors continuously. Applying regression algorithms to stereo acoustic data would be interesting, as all previous tests were performed using one microphone. The experiment will require better synchronization methods to capture mouse movements with the stereo audio from a smartphone. The additional dimensions given by the second microphone seem to provide an extra successful push needed for accurate predictions; thus, coordinating with angle directions could allow the continuous recreation of mouse movements.

Moreover, the experiment discussed in Section 7.2 showed results allowing us to infer realistic activities of a user on a computer, closing a window. This experiment could be extended by finding other situations where mouse movements reveal sensitive information. This experiment was simple enough to prove and required an accessible testing protocol. Consequently, this could be further extended into new prediction scenarios.

References

[1] Kiran Balagani et al. "We can hear your PIN drop: An acoustic side-channel attack on ATM PIN pads". In: *European Symposium on Research in Computer Security*. Springer. 2022, pp. 633–652.

[2] Stefano Cecconello et al. "Skype & type: Keyboard eavesdropping in voice-over-IP". In: *ACM Transactions on Privacy and Security (TOPS)* 22.4 (2019), pp. 1–34.

[3] Mengqi Chen, Yongpan Zou, and Kaishun Wu. "Behavicker: Eavesdropping Computer-Usage Activities Through Acoustic Side Channel". In: *Available at SSRN 4019830* ().

[4] Peng Cheng et al. "SonarSnoop: Active acoustic side-channel attacks". In: *International Journal of Information Security* 19 (2020), pp. 213–228.

[5] Robert Ciesla. "Bits, Sample Rates, and Other Fundamentals of Digital Audio". In: *Sound and Music for Games: The Basics of Digital Audio for Video Games*. Berkeley, CA: Apress, 2022, pp. 1–24. ISBN: 978-1-4842-8661-6. DOI: 10.1007/978-1-4842-8661-6_1. URL: https://doi.org/10.1007/978-1-4842-8661-6_1.

[6] Yuyi Fang et al. "Eavesdrop with PoKeMon: Position free keystroke monitoring using acoustic data". In: *Future Generation Computer Systems* 87 (2018), pp. 704–711.

[7] Daniel Genkin et al. "Synesthesia: Detecting screen content via remote acoustic side channels". In: *2019 IEEE Symposium on Security and Privacy (SP)*. IEEE. 2019, pp. 853–869.

[8] E Bruce Goldstein and Laura Cacciamani. *Sensation and perception*. Cengage Learning, 2021.

[9] Haritabh Gupta et al. "Deciphering text from touchscreen key taps". In: *Data and Applications Security and Privacy XXX: 30th Annual IFIP WG 11.3 Conference, DBSec 2016, Trento, Italy, July 18-20, 2016. Proceedings 30*. Springer. 2016, pp. 3–18.

[10] Michael I Jordan and Tom M Mitchell. "Machine learning: Trends, perspectives, and prospects". In: *Science* 349.6245 (2015), pp. 255–260.

[11] Sotiris B Kotsiantis, Ioannis D Zaharakis, and Panayiotis E Pintelas. "Machine learning: a review of classification and combining techniques". In: *Artificial Intelligence Review* 26 (2006), pp. 159–190.

[12] Haiou Li et al. "Deep learning methods for protein torsion angle prediction". In: *BMC bioinformatics* 18.1 (2017), pp. 1–13.

[13] Jian Liu et al. "Snooping keystrokes with mm-level audio ranging on a single phone". In: *Proceedings of the 21st Annual International Conference on Mobile Computing and Networking*. 2015, pp. 142–154.

[14] Rajalakshmi Nandakumar et al. "Fingerio: Using active sonar for fine-grained finger tracking". In: *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*. 2016, pp. 1515–1525.

[15] MZ Naser and Amir Alavi. "Insights into performance fitness and error metrics for machine learning". In: *arXiv preprint arXiv:2006.00887* (2020).

[16] Keiron O'Shea and Ryan Nash. "An introduction to convolutional neural networks". In: *arXiv preprint arXiv:1511.08458* (2015).

[17] Karol J Piczak. "Environmental sound classification with convolutional neural networks". In: *2015 IEEE 25th international workshop on machine learning for signal processing (MLSP)*. IEEE. 2015, pp. 1–6.

[18] Christopher J Plack and Robert P Carlyon. "Loudness perception and intensity coding". In: *Hearing* (1995), pp. 123–160.

[19] Jordi Pons et al. "Timbre analysis of music audio signals with convolutional neural networks". In: *2017 25th European Signal Processing Conference (EUSIPCO)*. IEEE. 2017, pp. 2744–2748.

[20] Kun Qian et al. "Bird sounds classification by large scale acoustic features and extreme learning machine". In: *2015 IEEE Global Conference on Signal and Information Processing (GlobalSIP)*. IEEE. 2015, pp. 1317–1321.

[21] Preeti Rao. "Audio signal processing". In: *Speech, Audio, Image and Biomedical Signal Processing using Neural Networks*. Springer, 2008, pp. 169–189.

[22] Susmita Ray. "A quick review of machine learning algorithms". In: *2019 International conference on machine learning, big data, cloud and parallel computing (COMITCon)*. IEEE. 2019, pp. 35–39.

[23] *scipy.stats.ttest_ind – SciPy v1.8.0 Reference Guide*. 2024. URL: https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.ttest_ind.html.

[24] Garima Sharma, Kartikeyan Umapathy, and Sridhar Krishnan. "Trends in audio signal feature extraction methods". In: *Applied Acoustics* 158 (2020), p. 107020.

[25] Gail M Sullivan and Richard Feinn. "Using Effect Size-or Why the P Value Is Not Enough". en. In: *J Grad Med Educ* 4.3 (Sept. 2012), pp. 279–282.

[26] Xue Ying. "An overview of overfitting and its solutions". In: *Journal of physics: Conference series*. Vol. 1168. IOP Publishing. 2019, p. 022022.

[27] Yuni Zeng et al. "Spectrogram based multi-task audio classification". In: *Multimedia Tools and Applications* 78 (2019), pp. 3705–3722.

[28] Zhi-Hua Zhou. *Machine learning*. Springer Nature, 2021.

[29] Tong Zhu et al. "Context-free attacks using keyboard acoustic emanations". In: *Proceedings of the 2014 ACM SIGSAC conference on computer and communications security*. 2014, pp. 453–464.