KEYSTROKE TRANSCRIPTION FROM ACOUSTIC EMANATIONS USING CONTINUOUS WAVELET TRANSFORM

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF INFORMATICS OF
THE MIDDLE EAST TECHNICAL UNIVERSITY

BY

ABDULLAH ÖZKAN

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF SCIENCE
IN
THE DEPARTMENT OF CYBER SECURITY

SEPTEMBER 2022

Approval of the thesis:

KEYSTROKE TRANSCRIPTION FROM ACOUSTIC EMANATIONS USING CONTINUOUS WAVELET TRANSFORM

Submitted by ABDULLAH ÖZKAN in partial fulfillment of the requirements for the degree of **Master of Science in Cyber Security Department, Middle East Technical University** by,

Date: 01.09.2022

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last name : Abdullah Özkan

Signature :

ABSTRACT

KEYSTROKE TRANSCRIPTION FROM ACOUSTIC EMANATIONS USING CONTINUOUS WAVELET TRANSFORM

Özkan, Abdullah

MSc. Department of Cyber Security

September 2022, 62 pages

One of the most common methods of communication is written communication. Written communication has been found in various forms over the years and has changed shape with technical and technological developments. Today, written communication has shifted to digital media and keyboards have become one of the most frequently used entry points. This makes keyboards a critical node in the flow of information. There are several ways in which information entered through the keyboard can leak. Acoustic propagation is one of these leakage pathways. In the literature, various approaches have been proposed for this attack type, which aim to process the keystroke sounds and capture the information. However, the usage of continous wavelet transform for this purpose has not been explored before. Continous wavelet transform provides better resolution in both time and frequency for impulse-like signals. Therefore, this transformation is better suited to the analysis of keystroke sounds than other conventional transforms.

In this thesis, we propose a method based on the continuous wavelet transform for transcription of keystrokes from the acoustic emanations of a keyboard, using wave files as input and recovering the written text as output with up to 57.2% accuracy. Initially a dataset was generated by recording keystroke sounds of 38 keys pressed multiple times in different ways. These were then analysed using the continuous wavelet transform in order to detect, segregate and obtain features of keystroke sounds. Various classification algorithms were tested and performances were recorded. Recommendations were made for improving the system output by using dictionaries and language models, as well as the information obtained from the confusion matrix itself.

Keywords: Acoustic Propagation, Text Extraction, Keyboard, Machine Learning, Continuous Wavelet Transform

ÖZ

SÜREKLİ DALGACIK DÖNÜŞÜMÜ KULLANILARAK AKUSTİK YAYILIMLARDAN TUŞ ÇIKARIMININ YAPILMASI

Özkan, Abdullah

Yüksek Lisans, Siber Güvenlik Bölümü

Eylül 2022, 62 sayfa

En yaygın iletişim yöntemlerinden biri yazılı iletişimdir. Yazılı iletişim yıllar içinde çeşitli biçimlerde bulunmuş, teknik ve teknolojik gelişmelerle şekil değiştirmiştir. Günümüzde yazılı iletişim dijital ortama kaymış ve klavyeler en sık kullanılan giriş noktalarından biri haline gelmiştir. Bu, klavyeleri bilgi akışında kritik bir düğüm haline getirir. Klavyeden girilen bilgilerin sızmasının birkaç yolu vardır. Akustik yayılım bu sızıntı yollarından biridir. Literatürde, tuş seslerini işlemeyi ve bilgiyi yakalamayı amaçlayan bu saldırı türü için çeşitli yaklaşımlar önerilmiştir. Ancak, bu amaçla sürekli dalgacık dönüşümünün kullanımı daha önce araştırılmamıştır. Sürekli dalgacık dönüşümü, darbe benzeri sinyaller için hem zaman hem de frekansta daha iyi çözünürlük sağlar. Bu nedenle, bu dönüşüm, diğer geleneksel dönüşümlere göre tuş seslerinin analizine daha uygundur.

Bu tezde, bir klavyenin akustik yayılımlarından gelen tuş vuruşlarının transkripsiyonu için, dalga dosyalarını girdi olarak kullanarak ve yazılı metni %57,2'ye kadar doğrulukla çıktı olarak geri alarak, sürekli dalgacık dönüşümüne dayalı bir yöntem öneriyoruz. Başlangıçta, farklı şekillerde birden çok kez basılan 38 tuşun tuş seslerini kaydederek bir veri seti oluşturuldu. Bunlar daha sonra tuş vuruşu seslerinin özelliklerini saptamak, ayırmak ve elde etmek için sürekli dalgacık dönüşümü kullanılarak analiz edildi. Çeşitli sınıflandırma algoritmaları test edilmiş ve performanslar kaydedilmiştir. Sözlükler ve dil modelleri kullanılarak sistem çıktısının iyileştirilmesi ve ayrıca hata matrisinin kendisinden elde edilen bilgiler için önerilerde bulunulmuştur.

Anahtar Sözcükler: Akustik Yayılım, Metin Çıkarımı, Klavye, Makine Öğrenmesi, Sürekli Dalgacık Dönüşümü

To my family, who continued to believe in me even when my trials failed, and to my advisors who helped me complete the work by showing family compassion

ACKNOWLEDGEMENTS

The author wishes to express his deepest gratitude to his supervisor [redacted] Kılıç and co-supervisor [redacted] for their guidance, advice, criticism, encouragements and insight throughout the research.

TABLE OF CONTENTS

ABSTRACT v
ÖZ vii
DEDICATION ix
ACKNOWLEDGEMENTS x
TABLE OF CONTENTS xi
LIST OF TABLES xiii
LIST OF FIGURES xiv
CHAPTER
1. INTRODUCTION 17
1.1. Problem Statement 18
1.2. Motivation 19
1.3. Contribution 19
1.4. Scope 19
1.5. Limitations 20
1.6. Structure of the Thesis 20
2. LITERATURE REVIEW 23
2.1. Side-Channel Attacks Using Acoustic Emanations to Recover Input 23
2.2. Life of a Keystroke 24
2.3. Keystroke Detection 24
2.4. Keystroke Detection 25
2.5. Keystroke Detection 26
2.6. Keystroke Detection 28
2.7. Keystroke Detection 29
2.7.1. Tree Models 29
2.7.2. Linear Discriminant Analysis 29
2.7.3. Naive Bayes Classifiers 29

2.7.4. Support Vector Machines 30
2.7.5. K-Nearest Neighbors 30
2.7.6. Ensemble Models 30
3. DATASET CONSTRUCTION 31
4. KEYSTROKE TRANSCRIPTION 35
4.1. Continuous Wavelet Transform 35
4.2. Keystroke Detection and Segmentation 36
4.3. Normalization and Feature Extraction 37
4.4. Keystroke Classification 38
5. RESULTS 39
5.1. Segmentation Results 39
5.2. Feature Extraction Results 42
5.3. Classification Results 46
6. DISCUSSION 53
6.1. Mitigations 53
7. CONCLUSIONS 55
REFERENCES 57
APPENDICES 60
APPENDIX A 60
APPENDIX B 61
APPENDIX C 62

LIST OF TABLES

Table 2.1 The four constraint types and their meanings. Adapted from Dictionary Attacks Using Keyboard Acoustic Emanations, by Berger et al., 2006, Proceedings of the 13th ACM conference on computer and communications security. Copyright (2006) 27
Table 5.1 Validation and test accuracies of all optimizable models in classification learner for part 5 47
Table 5.2 Training and test datasets of the classification phases 47

LIST OF FIGURES

Figure 2.1. Training phase (above) and recognition phase (below). Adapted from Keyboard Acoustic Emanations Revisited, by L. Zhuang, F. Zhou and J. D. Tygar, 2005, *ACM Transactions on Information and System Security 13, 1.* Copyright (2005) 26
Figure 3.1. The keyboard showing the 38 keys (red boxed) considered in the study 31
Figure 3.2. Recording setting where the datasets "TwentyFives", "Panagrams" and "ThreeStates" were recorded, where the distance between the keyboard and the microphone was 50 cm.....33
Figure 4.1. The block diagram of the proposed system. 35
Figure 5.1. Plot of MeanRealCWTForDetection with filter bank frequency range from 100 Hz to 9000 Hz 40
Figure 5.2. Plot of MeanRealCWTForDetection with filter bank frequency range from 1000 Hz to 9000 Hz 40
Figure 5.3. Plot of MeanRealCWTForDetection with filter bank frequency range from 400 Hz to 6000 Hz 41
Figure 5.4. Plot of MeanRealCWTForDetection with filter bank frequency range from 400 Hz to 12000 Hz 41
Figure 5.5. Plot of signal obtained after processing with CWT with frequency range from 400 Hz to 9000 Hz 42
Figure 5.6. Confusion matrix of validation while using the standard deviation to reduce dimensionality 43
Figure 5.7. Confusion matrix of test while using the standard deviation to reduce dimensionality 43
Figure 5.8. Confusion matrix of validation while using the maximum to reduce dimensionality 44

Figure 5.9. Confusion matrix of test while using the maximum to reduce dimensionality 45
Figure 5.10. Confusion matrix of validation while using the mean to reduce dimensionality 45
Figure 5.11. Confusion matrix of test while using the mean to reduce dimensionality... 46
Figure 5.12. Confusion matrix of validation for part 1 48
Figure 5.13. Confusion matrix of test for part 1 48
Figure 5.14. Confusion matrix of validation for part 2 49
Figure 5.15. Confusion matrix of test for part 2 50
Figure 5.16. Confusion matrix of validation for part 3 50
Figure 5.17. Confusion matrix of test for part 3 51
Figure 5.18. Confusion matrix of validation for part 4 51
Figure 5.19. Confusion matrix of test for part 4 51

CHAPTER 1

INTRODUCTION

Communication is one of the building blocks of the human adventure. Communication has made it possible for people to stay together, work together, and to advance the accumulation of humanity by division of labor. We can divide communication into three as verbal, written and body language. Of these, written communication is the most permanent. For this reason, the way of transferring the valuable information to be protected is written. All valuable information such as financial records, military plans, patent information, source codes are available in written form. However, this value has made written communication the most attacked communication method.

In a world that is increasingly digitized and digital information is equated with permanence, the keyboard is among the main tools through which information passes. However, it is often overlooked when it comes to protecting information. One of the main weaknesses in the use of the keyboard is that the sound produced by pressing the keys contains the information of the pressed key. The first study stating that this information could be targeted by attackers was made in 2004 by Asonov (Asonov & Agrawal, 2004) and his team. Since this date, the academy has approached the subject from different angles and introduced different methods of accessing the text entered from the keyboard by using the keyboard acoustic emanation.

Keystroke transcription of recorded keyboard emanation by analyzing it after various transformations is not the only way to extract keystrokes from audio data. In a study conducted after Asonov and Agrawal (2004), the position of the key on the keyboard was tried to be determined by using the difference between the arrival times of the keystroke sound to different microphones (Zhu et al., 2014). In other studies, text entered has been obtained by using the sound produced by the touch screen when the virtual keyboards of smartphones are used (Shumailov et al., 2019; Teo et al., 2021).

All these efforts demonstrate the impact of keystroke emanation on privacy. This effect is often underestimated and necessary precautions are not taken to prevent attacks. We hope that our study will contribute to taking the necessary countermeasures by showing how easily the attack can actually be carried out. The possibility that even a low-level attacker can capture high-critical information by using non-invasive and inexpensive methods shows how big risks are created by keyboard acoustic emanation attacks.

When the studies are examined, in the method followed in general; It can be seen that the features of the audio data recorded with microphones are extracted with various techniques, and then these features are processed with supervised or unsupervised machine learning methods, and the audio data and keyboard keys are matched.

Studies that perform keystroke transcription using the sound emanation of the keyboard alternate between a limited number of methods such as FFT and MFCC in extracting sound characteristics. Despite the fact that many studies have been carried out on the subject over the years, the lack of a systematically prepared and labeled keystroke audio data that can be used in future studies is another shortcoming of the academy. Our study aims to contribute to the solution of both these problems. In the study, a unique technique that was not used in previous studies was prefered to extract the features of audio data and the keystroke audio recordings of the study were presented to the academy.

In this part of the thesis; the problem statement, motivation, aim of the study, scope of the thesis, and outline of the thesis are presented.

1.1. Problem Statement

Obtaining the text entered using keystroke acoustic emanations is a serious problem in terms of information security. It has been shown by academic studies that this risk does not occur only in laboratory conditions, but also affects our daily life routine. When the attack scenarios of the existing studies were examined; a student writing his thesis in the library (Zhu et al., 2014), an employee holding meeting notes with his keyboard while meeting over a video meeting application such as Skype (Compagno et al., 2016), or even a businessman typing the password of his phone (Shumailov et al., 2019) can be affected by the attack surface brought by acoustic emanations.

When a key is pressed on the keyboard, the key vibrates at the base of the keyboard, making a sound. The reason why keyboard acoustic emanation attacks are possible is because different keys make different sounds since each key causes the keyboard to vibrate differently. The difference between the sounds is related to the position of the key on the keyboard (Asonov & Agrawal, 2004). Keys that are close together on the keyboard make similar sounds. Therefore, the probability of confusion with each other during the prediction phase is high (Berger et al., 2006).

There are also factors that affect the sound created by the keystroke on the keyboard and the success of the model, apart from the position of the key on the keyboard. Pressing force on keys, keyboard model (Asonov & Agrawal, 2004), typist speed (Slater et al., 2019) are some of these factors.

In addition to all these, there are attacks that do not deal with the characteristics of the sound coming out of the keyboard. In this study by Berger (Berger et al., 2006) et al., instead of the structural features of the sound, the time it takes for the sound to reach the microphones from the keyboard is used in the extraction of the text.

As a result, keyboard acoustic emanations can turn into a threat in the hands of attackers by using both the structural properties of the sound and the speed of sound. The limited number of academic studies conducted in this area actually reveal how large this threat can reach. However, in the current situation, the studies that progress by extracting the

structural features of the keystroke sound are stuck in certain methods in the feature extraction stage. The fact that the audio recordings used in the studies are not shared adds another difficulty to the new studies, making it difficult to examine the subject in wider circles. These two situations can negatively affect the understanding of the criticality of the problem.

In this study, we aim to obtain the entered text by revealing the structural features of the sound with a new method and evaluating it within the original model we have created. At the same time, we hope to support future work by sharing the audio recordings we created during the study.

1.2. Motivation

The main motivation of this study we have prepared was to approach the problem from a different angle and to find the opportunity to get to know the keystroke emanations better. At the same time, by sharing the research dataset we have prepared, it is to ensure that this ignored security weakness is on our agenda with new studies.

1.3. Contribution

With this study, we bring a new model to the studies on the topic of keystroke transcription by using acoustic keyboard emanations.

The most important contribution of our study is the use of continuous wavelet transform method, which is much more suitable for the analysis of spiky signals such as keystroke, instead of feature extraction methods such as FFT, Cepstrum or cross-correlation, which were used before. With this method, more detailed features of the keystrokes can be extracted.

Another important contribution comes from our dataset. The scarcity of labelled sound data that can be used in the analysis of acoustic emanations has been mentioned in previous studies. With this study, we present the labelled, systematically obtained raw audio dataset to the service of the academy.

1.4. Scope

The study deals with extracting the structural features of keystroke acoustic emanations and obtaining the entered text with machine learning models. Other features of acoustic emanations such as speed and time were out of the scope of this study. Machine learning has been carried out with labelled data, and clustering methods were out of scope. Only 38 keys, covering Turkish and English alphabets, were included in the study. The keys in scope were shared under Chapter 3.

1.5. Limitations

In the prepared records, the typist was asked to write the given text using a single finger. This makes keystrokes very easy to distinguish; however, it also distances the work from real-world conditions. Slater and his team (2019) stated that in the dataset they created in a room where real world conditions are imitated, the overlapping keystrokes situation, where the sound of the next key starts before the sound of a key is completed, occurs for 31.6% of the keys. As a result, it is impossible to completely parse overlapping keys.

The second limitation of the study is the manual determination of the MinPeakHeight parameter used in the keystroke detection stage. This parameter is used to locate keystrokes in the signal. It prevents the values below the value specified for the parameter from being marked as keystroke. In the current study, this value is determined manually. Since keystroke detection is performed on the continuous wavelet transform signal and not on raw audio recordings, there is a clear difference between the amplitude of the range with the keystroke and the amplitude of the range that is not. Therefore, a MinPeakHeight value that can be used for all recordings can be easily determined on the signal plot.

The use of different keyboards is a common limitation of keystroke transcription from acoustic signals studies. The fact that the keyboards used in the training and test stages are the same seems to be effective on the success of the model. In their study, Asonov and Agrawal (2004) tested the model trained with keyboard A with keyboards B and C. The model, which showed 88% success when training and testing with the recordings taken from the same keyboard, showed a success of approximately 47% when different keyboards were used for the recordings taken during the training and testing stages. This situation reveals the need to know the model of the attacked keyboard in the performance of the study.

Another common limitation is the assumption that the analyzed audio data belongs to a typing session. Although a specially noise-prevented laboratory environment was not used during recordings, impulsive sounds that could be confused with key sounds were not specifically added to the recording. The operability of the model in a sound data where all these sounds are present at the same time has not been examined within the scope of this study.

1.6. Structure of the Thesis

Thesis continues as follows:

In Chapter 2, studies that perform keystroke transcription from keyboard acoustic emanations were grouped according to the acoustic feature they use in a detailed literature review.

In Chapter 3, the preparation method of the dataset consisting of keystroke sound recordings, the environmental conditions where the recordings were taken, the

characteristics of the recordings and the specifications of the recording materials were explained.

Chapter 4; explains the stages of the study sequentially, introduces the developed model. Here, the stages of the model were considered as black box and the details of the stages were not mentioned except for the input and output data.

Chapter 5 describes the results of the study and the attempts made to achieve the best results.

Chapter 6 is the discussion of the results and future work.

CHAPTER 2

LITERATURE REVIEW

Keytroke transcription from acoustic signals attracted the attention of the academy with the study “Keyboard Acoustic Emanations” conducted in 2004 (Asonov & Agrawal, 2004). This study is the first one in the field. Since then, several studies have examined the topic and increased the percentage of accurate predictions with the help of new perspectives and techniques.

2.1. Side-Channel Attacks Using Acoustic Emanations to Recover Input

Keystroke transcription from acoustic signals is a type of side-channel attack. Side-channel attacks (SCAs) aim at extracting secrets from a chip or a system, through measurement and analysis of physical parameters (Bhunia & Tehranipoor, 2019). Characteristics that can be exploited in a side-channel attack include timing, power consumption, and electromagnetic and acoustic emissions (Side-Channel Attack - Glossary | CSRC, n.d.).

One of the offensive uses of keyboard acoustic emanations is on keystroke recovery. Keystrokes are recorded and their features are extracted. The sound of each key is unique. These features are used in machine learning models to classify sounds. Corrections can also be made with language models to improve model performance.

Acoustic emanations is not only used for keystroke recovery (Shumailov et al., 2019). Virtual keyboards on phones produce sound when used. It has been shown that these sounds are also distinguishable and can be returned to the input using sound data (Shumailov et al., 2019; Teo et al., 2021).

There is also variation in studies that perform keystroke recovery using keyboard acoustic emanations. Although most of the studies in this field have tried to obtain the input by using the features of the audio data, there are also studies that want to reach the input entered with the time difference of arrival method. In this method, using at least two microphones, the difference between the times the keystroke sound reaches these microphones is examined and the position of the key on the keyboard is tried to be determined by considering the sound velocity. The necessity of knowing the distance of the microphones from the keyboard appears as a limitation (Zhu et al., 2014).

The subject of this thesis is about detecting the entered text using the features of keyboard acoustic emanations. The next part of the Chapter will consider the work done on this topic.

2.2. Life of a Keystroke

This section describes the general features of a keystroke. These features are important in understanding why keystrokes sound different and how this difference can be used.

In early studies (Asonov & Agrawal, 2004), a keystroke duration was found to be approximately 100 ms. A keystroke signal has two peaks: during push and release, respectively. There is relative silence between these peaks. The push peak has two active intervals: the touch peak when the keyboard is touched, and the hit peak when the key is pressed and the key touches the keyboard base, respectively. It has been stated that touch peak is more suitable for feature extraction. However, if the entire active interval is used, the result is more successful.

Asonov & Agrawal (2004) also investigated why different keys make different sounds. In experiments, they found that this is because each key is in a different position on the keyboard and therefore vibrates the base of the keyboard differently.

Zhuang et al. (2005) found the keystroke time to be 100 ms. similar to the previous study.

2.3. Keystroke Detection

Keystroke detection is the first step of the study and the difficulty level changes depending on the typing style. In order to overcome this problem, some studies have introduced typing with the "Straw Man Typing" style as a constraint (Asonov & Agrawal, 2004). In straw man typing, each key is pressed using the same finger vertically to ensure the same force and same angle (Halevi & Saxena, 2015). There is also a clear interval between each press. In this way, the silence and keystroke data are clearly visible on the signal.

However, in the analysis of texts written using multiple fingers at the same time, it has been shown that the data of different keystrokes are intertwined (Slater et al., 2019). This means that the next key starts before the touch-press-release phases of a key are complete.

Slater et al. (2019) demonstrated this phenomenon in their work. In the plot of the writing of the letters O-N-E, they showed that the touch and press peaks of the "N" key are included in the signal before the release peak of the "O" key. (Slater et al., 2019)Another possible obstacle that can be extracted from the work is the background noise. In intense noisy environments, background noise can be confused with key sounds. This makes keystroke detection even more difficult. They showed that some peaks created by noise can be mixed with the key sound.

Zhuang et al. (2005) divided the signal into time windows and calculated energy levels for each window. The Discrete Fourier Transform was used in this calculation. The energy level difference between the keystroke data and silence is high. They accepted the sudden increase in energy level as the start of the keystroke. This method was used by many later studies (Halevi & Saxena, 2015; Kelly, 2010; Wang et al., 2016; Zhu et al., 2014).

After the keystroke start is found, a certain range around this point is reserved as keystroke data. While some studies give this range as 100 ms (Asonov & Agrawal, 2004; Berger et al., 2006; Zhu et al., 2014), some studies have stated that 40 msec gives better results (Kelly, 2010; Slater et al., 2019; Zhuang et al., 2005).

2.4. Keystroke Detection

Asonov & Agrawal (2004) used direct frequency spectrum features. Since the signal is stronger, the frequency distribution was calculated in the push peak part. Using the FFT, the features were extracted from the 2-3 ms long touch peak region in the push peak part. In addition, in this study, it was determined that higher frequencies contain less information. To reduce the computation time and the memory required by the DFT, Fast Fourier Transform (FFT) is preferred (Mneney, 2008),

Zhuang (Zhuang et al., 2005) et al. found that Cepstrum features, in particular MFCC (Mel-Frequency Cepstral Coefficients), yielded better results. A 40 ms interval after the start of the keystroke was used for the calculation of the feature. This period covers approximately the entire push interval.

MFCC features were later preferred by many studies (Anand & Saxena, 2018; Cecconello et al., 2019; Compagno et al., 2016; Wang et al., 2016). MFCC calculation consists of five steps; windowing the signal, applying the DFT, taking the log of the magnitude, and then warping the frequencies on a Mel scale, followed by applying the inverse DCT (Discrete Cosine Transform) (Rao & K E, 2017).

Berger et al. (2006) stated that they got the best results with simple cross-correlation for their study, instead of using the FFT or the Cepstrum. With these properties, they calculated similarity matrices for both the press and release parts of each keystroke. Thus, for a word consisting of N letters, they formed two matrices of N x N dimensions. These matrices were then combined under a single similarity matrix by taking their unweighted average.

Kelly (2010) tried both FFT and Cepstrum features in his study, but stated that the highest success was achieved with the combination of these two features. The combined feature dataset was created by simply concatenating the two sets of values. He found that Cepstrum features achieves accuracies as high as 88.9% on labelled training data and FFT features show similar performance. However, when these features are combined, accuracy goes up to 93.9%.

Slater et al. (2019) said that the FFT was more successful for their dataset. This is due to the very small size of the datasets used by previous studies.

2.5. Keystroke Detection

In the 2004 study (Asonov & Agrawal, 2004), classification was performed with labeled data using a neural network. In order to train the neural network, the raw signal was normalized and the training was done with the created feature-key pairs. The study showed that the effect of different typing styles on the model was low. Therefore, if the keyboard is the same, the model can be trained for one person and tested for a different person. Even with different keyboards, only performance drops, but transcriptions can still be made.

Zhuang et al. (2005) used a model that does not need training with labeled data in the study. Linear classification was preferred instead of a neural network. The basic assumption in this article was that the written text was not random and was influenced by the language model. Here, the classifier was first trained in a training set, then tried both in the training set and in two more sets.

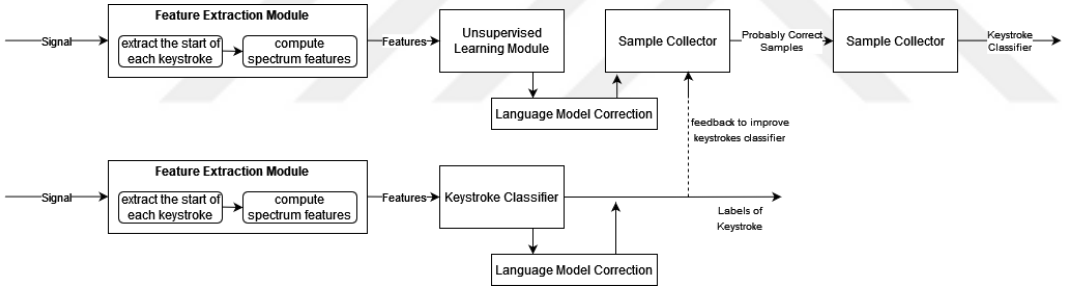


Figure 2.1. Training phase (above) and recognition phase (below). Adapted from Keyboard Acoustic Emanations Revisited, by L. Zhuang, F. Zhou and J. D. Tygar, 2005, *ACM Transactions on Information and System Security 13, 1*. Copyright (2005)

Figure 2.4 shows the stages of the work done by Zhuang et al. In the training phase, the model builds a keystroke classifier using unsupervised learning. Then in the recognition phase, it recognizes keystrokes using the classifier from training phase and feedback to a sample collector to improve the classifier. Unlabelled training data is clustered into one of the K classes in the unsupervised learning module. K must be a value greater than the number of keys in the scope of the study. As K grows, more information is collected from the sound samples, but the sensitivity to noise increases. Therefore, the optimum K must be found. The best result was obtained when K=50 was selected.

The second step is to extract the text from these clusters. For this, the Hidden Markov Model (HMM) was used. In this article, the feedback-based training process was run over and over again. The output of the previous iteration was used as the labeled data of the next.

Three different machine learning models were used and the results were compared. The best method has been linear classification, then Gaussian mixtures, and then neural networks.

Berger et al. (2006) calculates a binary called constraint, which expresses the relationship between two keystrokes. A similarity matrix was used to calculate these constraints. A word has a certain set of constraints; however, there can be more than one word with this set of constraints. For example; the words help, "Iraq", "nose" and "path" all have the same set of constraints. The types and meanings of constraints are shown in Table 2.1.

Table 2.1 The four constraint types and their meanings. Adapted from Dictionary Attacks Using Keyboard Acoustic Emanations, by Berger et al., 2006, Proceedings of the 13th ACM conference on computer and communications security. Copyright (2006)

Type	*Meaning*
EQ	Ki=Kj means that the i'th keystroke and the j'th keystroke stem from the same key on the keyboard
ADJ	Ki~Kj means that the j'th keystroke stem from a key that is adjacent to tke key which the i'th keystroke stems from. For example, Q=W but not Q=E since E is located two positions away from Q on QWERTY keyboard.
NEAR	Ki~Kj means that Ki and Kj are at most two keys apart on the keyboard, e.g., keys NEAR G include R,D,N,J etc.
DIST	Distant keys are those that are not NEAR to each other.

Words suitable for these constraints were brought from the dictionary. Scanning the dictionary can be done quickly with suitable data structures. The critical point here is to be able to calculate the correct constraints. Even one incorrect constraint can result in completely different words. A set of candidate words was created by bringing words suitable for constraints from the dictionary. Performance evaluation was made by looking at the rank of the entered word in the set.

Compagno et al. (2016) tested a number of machine learning models in action. These models were Logistic Regression (LR) classifier, Linear Discriminant Analysis (LDA), Support Vector Machines (SVM), Random Forest (RF) and K-Nearest Neighbors (KNN). Extensive grid search method was preferred in order to optimize the models in the tests where MFCC features are used.

Slater et al. (2019), unlike previous studies, did not prefer a factored approach and developed an end-to-end model. In this model, detection, feature extraction and classification stages were resolved in a single block. At this stage, they benefited from the recurrent neural network architecture. In their work, they present a comparison of the factored approach and the end-to-end approach.

2.6. Keystroke Detection

Some studies have made corrections on the predictions and improved the predictions by using the features of the language of the text after the classification phase. At the most basic level, predictions are checked with the help of a dictionary and corrected if a word in the dictionary is reachable with a certain number of letter changes. This number usually varies according to the length of the word. For example, while the words that can be reached with only one letter change in a five-letter word are searched in the dictionary, up to 3 letter changes are allowed in a word with ten or more letters. This method is called "spell checking" (Levenshtein, 1966).

However, with spell checking, the word "fur example" will remain uncorrected. Spell checking does not take into account the relationship between words, the frequency of use of words, and sentence structures. The most appropriate candidate words in the dictionary can be decided by looking at the context of the sentence or the words before and after the related word. This method is called the "n-gram language model" (Zhuang et al., 2005).

In addition, some studies have calculated a probability for each letter in the next prediction based on the letters in the word using developed language models, and have given a certain weight to the probabilities coming from the language model while choosing the appropriate one among the possible predictions obtained using audio features.

Asonov and Agrawal (2004) did not attempt to improve their results with error checking or the language model. However, 88% of the keys pressed were predicted correctly in the first three attempts in the recordings obtained using the same keyboard and the typist.

Zhuang et al. (2005) performed error correction with the spell checking method. In addition, they improved the predictions with the trigram language model they created. When they took a 10-minute audio recording of an English text as input, they were able to return 96% of the entered text.

Berger et al. (2006) did not need to make improvements with the language model, as they assumed that the entered word was a word that exists in the dictionary. While they could find words with 10+ characters 90% correct in the first 50 tries, this rate was 73% for all words.

Cecconello et al. (2019) used an English dictionary to develop their results and selected each word by ordering them according to their probability of use. In tests using different laptops, it was stated that 98% of the 20 predictions for the MacBook Pro laptop were correctly predicted.

2.7. Keystroke Detection

There are various statistical and machine learning methods that can be explored for clustering (unsupervised) and classification (supervised) of keystrokes. Due to time constraints, only some of these methods were tested, which are briefly mentioned below.

2.7.1. Tree Models

Tree-based machine learning is one of the most widely used methods. The decision tree can be a classification tree or a regression tree. The former predicts a class, whereas the latter predicts a number. Tree learning can be carried out in various ways. Random Forest, Rotation Foresty and MARS are some of the notable algorithms. Each node of the tree created in the decision tree method has a feature statement (except for the last nodes that report the result). The tree is divided into branches according to these nodes. In this method, the aim is to assign the features that differentiate the cluster the most to the highest level nodes (Quinlan, 1986).

2.7.2. Linear Discriminant Analysis

Discriminant Classifier or Linear Discriminant Analysis is a model that works on numeric values. It makes the classification by looking at the statistical features of the classes. When determining the axis it will form between the classes, the LDA uses two criteria (Fisher, 1936):

1. Maximize the distance between the means of the two classes, and
2. Minimize the variation within each class.

2.7.3. Naive Bayes Classifiers

Naive Bayes is a probabilistic machine learning algorithm based on the Bayes Theorem (Hand & Yu, 2001).

$$P(Y|X) = \frac{P(X|Y)P(Y)}{P(X)} \tag{2.1}$$

The Bayes Theorem formula shows the reduction of conditional probability to set probabilities. The probability that the C character to be classified in the S class is the product of the probability that all the characters in the text are in the S class. (Hand & Yu, 2001).

2.7.4. Support Vector Machines

The purpose of the SVM classification method is to ensure that the boundary line to be drawn between the classes on a plane passes through the furthest point to the elements of both groups. Maximizing the margin distance provides some reinforcement so that future data points can be classified with more confidence (Cortes & Vapnik, 1995).

2.7.5. K-Nearest Neighbors

While classifying each new data in the KNN algorithm, it is decided which class it belongs to by looking at the K nearest neighbors. The value of K is variable and should be defined neither too small nor too large. There are different distance metrics that can be used when measuring the distance between data. Euclidean, Manhattan, Cosine distances are some of them.

2.7.6. Ensemble Models

It is a method that aims to achieve more successful results by using more than one technique together. There are two main varieties of the method. In the Bagging method, the classification of a data in all used machine learning models is checked and the class is assigned to whichever class the majority puts the data. In the Boosting method, it applies the same method, but not all models are given the same weight, their weights vary according to their individual success (Opitz & Maclin, 1999).

CHAPTER 3

DATASET CONSTRUCTION

While recording, a Dell brand KB212-B model keyboard and a Logitech brand C510 model microphone were used. Records were created with Matlab program in wav format with 24 bits and 48 kHz. sampling frequency. The recordings were taken in a room, similar to an office environment where sounds such as the street and air conditioning were present in the background with an average SPL of 61 dBA.

During the study, only 38 frequently used keys were considered, which are shown as boxed in red in Figure 3.1.



Figure 3.1. The keyboard showing the 38 keys (red boxed) considered in the study

The recording process was completed in three stages. In the first stage, a dataset was created by pressing each of the 38 keys for 100 times in succession. In this way, using one finger to press the same key multiple times is called "Straw Man Typing" in the literature (Halevi & Saxena, 2012). To reduce the file sizes 100 keystrokes were divided into four separate sound files, each one containing 25 keystrokes. The microphone was placed behind the keyboard in the same plane. There was a distance of approximately 20 cm between the microphone and the keyboard during the recording. From now on this dataset will be called “Hundreds”.

In order to test the effect of changing acoustic conditions due to the recording setting and the environment, two other datasets were also recorded. The aim was to investigate the

hypothesis that a dataset that is not prepared in the same environment as the test data can also improve the performance of the model when used in training.

In the second stage, each of the 38 keys in the scope were pressed 25 times in succession with the same “Straw Man Typing”. Each sound file contains 25 keystrokes and there is only one sound file for each of the 38 keys. From now on this dataset will be called the “TwentyFives”. There was a distance of approximately 50 cm between the microphone and the keyboard during the recording and the keyboard was on an office table. The image of the recording environment is shown in Figure 3.2.

The recording of the third stage was made in the same environment as the second stage. In this stage, unlike the previous stages, the writing of two texts were recorded instead of pressing the keys in succession. Typing is still done using one finger, but following a text so that the keys pressed in succession are different. This writing style is called "Hunt and Peck Typing" in the literature (Halevi & Saxena, 2012).

- The first text was a panagram for Turkish: “Vakfın çoğu, bu huysuz genci plajda görmüştü.”. In the spelling of the capital letter v, “caps lock” is turned on and off. When the sentence is completed, the “enter” and “backspace” keys are pressed, respectively. In this way, 35 of the 38 keys in the scope were obtained. This text contains 49 keystrokes and has been recorded by typing it 20 times. Note that in this panagram, not all of the 35 keys are pressed equal number of times. In order to prevent biased results, only the first press of each unique key was considered. From now on this dataset will be called “Panagrams”.

- The second text was “new york quebec texas” prepared to get the remaining 3 keys, namely w, q and x. This text contains 21 keystrokes and has been recorded by typing 20 times. From now on this dataset will be called “ThreeStates”.

The 35x80 matrix from the "Panagrams" records and the 3x80 matrix from the "ThreeStates" records were combined. As a result of the process, 20 matrices of 38x80 dimensions were obtained. From now on this dataset will be called the “Panagrams and ThreeStates”.

Looking at the whole datasets; there are 3800 keystrokes in total in 152 records in the "Hundreds" dataset, 950 keystrokes in total in 25 records in “TwentyFives” dataset, 760 keystrokes in total in 20 records in the "Panagrams and ThreeStates" dataset. The datasets are available at https://github.com/aozkantr/KeystrokeRecordings.



Figure 3.2. Recording setting where the datasets “TwentyFives”, “Panagrams” and “ThreeStates” were recorded, where the distance between the keyboard and the microphone was 50 cm

CHAPTER 4

KEYSTROKE TRANSCRIPTION

The block diagram of the proposed system is shown in Figure 4.1.

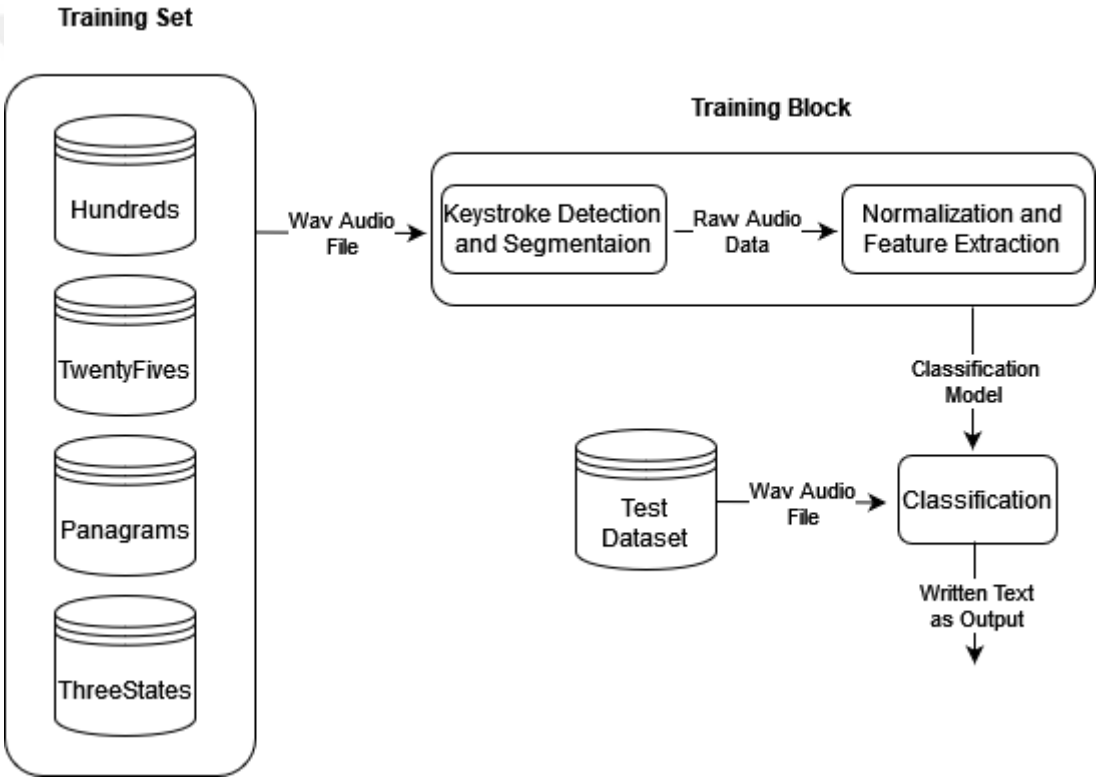


Figure 4.1. The block diagram of the proposed system.

Each one of these blocks are explained in detail in the next sections.

4.1. Continuous Wavelet Transform

Continuous Wavelet Transform (CWT) was used in the analysis of non-stationary signals. It was developed as an alternative to STFT to solve the resolution related problems. To achieve this aim, the continuous wavelet transform analyses the signal with different resolutions at different frequencies. The continuous wavelet transform was computed by changing the scale of the analysis window, shifting the window in time, multiplying by the signal, and integrating over all times (Torrence & Compo, 1998).

With CWT, high frequencies are better located in time, while low frequencies are better located in the frequency domain. It is suitable for the analysis of the signals that travel at low frequencies for a long time, while the high frequencies of the signal last for a short time. This is the nature of many signals in the real world. Impulsive signals such as keystrokes can be analyzed most accurately with CWT.

4.2. Keystroke Detection and Segmentation

The continuous wavelet transform of the raw signal was used in the keystroke detection stage. Detection stage is handled in the function named "KeystrokeExtractor". In order to prevent the noise and to detect the peak point more clearly, a narrower frequency range was preferred compared to the filterbank frequency range to be used in the segmentation stage. In the tests performed by trying different intervals, it was seen that the range from 400 Hz to 9000 Hz provided the most effective results for detection. Details of the trials are given under Chapter 5.

When the transform operation was performed with the created filterbank, the resulting signal for each scale had imaginary and negative values. In order to accurately detect the positive peaks pointing to the keystroke, the absolute value of the real part of the signal at each scale was taken. Then the mean value of the resulting signals were calculated. This process cleaned the signal and emphasized the peaks for keystroke detection.

In order to detect the peaks, a known peak detection algorithm was used, whose implementation in Matlab is known as the "findpeaks" function (Find Local Maxima - MATLAB Findpeaks, n.d.). Findpeaks function returns two vectors with the local maxima (peaks) of the input signal vector and the locations of these peaks. By using the parameters of the function, the minimum distance between the consecutive peaks (MinPeakDistance) or the smallest amplitude value that a peak should have (MinPeakHeight) can be configured.

In the study, the MinPeakHeight value was determined heuristically. This value is related to the amplitude of the signal and the amplitude value is affected by a number of factors, such as the quality of the microphone, microphone preamps, the distance between the microphone and the keyboard, and the force amount of keystrokes. However, due to CWT-based pre-processing, there was a significant difference in signal amplitudes between the times when keystroke was present and absent. Thanks to this difference, MinPeakHeight could be set to a value to cover all keystrokes. Since the keystroke signal is an impulse-like signal, it has high amplitude values in a narrow time interval, while it has relatively lower values in the remaining times.

While determining the MinPeakDistance, another limitation, which is the typist's use of one finger while typing, makes our work easier. Since one finger is used and the keystrokes do not overlap each other, the distance specified as MinPeakDistance can cover the entire keystroke quite safely.

After the keystroke onset time was detected, the raw audio data of this keystroke must be isolated from the record. At this stage, called segmentation, it was preferred to receive all of the keystroke data, including the press and release phases.

In order to get all the keystroke data, a total of 300 ms long blocks (50 ms before the peak point and 250 ms after the peak point) were taken for each keystroke. These values correspond to 2400 and 12000 samples, respectively, with the sampling frequency of 48 kHz. The created segments were combined in a matrix named RawKeystrokes. Each row of the RawKeystrokes matrix holds 14400 samples for a separate keystroke. Code for keystroke detection and segmentation is given in Appendix A as "Keystroke Extractor".

4.3. Normalization and Feature Extraction

The data obtained after the detection and segmentation stages is still raw audio signal data. Energy normalization is required to prevent the force differences applied to the keys during the recording from misdirecting the model. A function (RawAudioNormalizer function) was written for this purpose, which takes the raw signal data and normalizes the values between -1 and 1.

Normalized segments were then used in feature extraction (FeatureExtractor function) which also utilizes continuous wavelet transform. Different from the filterbanks used in the keystroke detection stage, in the filter bank specified for feature extraction, the frequency range was kept wider, i.e., from 400 Hz to 24 kHz. The reason for this is that data at different frequencies are wanted to be evaluated with minimum loss. While choosing the 24 kHz value, the sampling frequency of 48 KHz was taken into account, due to the Nyquist Theorem (Lévesque, 2014). The lower bound of 400 Hz was selected because keystrokes' data start from this frequency as reported in the study (Kelly, 2010). For this frequency range, the CWT has 80 scales.

After calculating the CWT of the keystrokes, we obtain a data matrix of size N x T x S, where N is the number of keystrokes, T is the segment size, which is 14400, and S is the number of CWT scales, which is 80 for the selected frequency range.

A 3-dimensional matrix makes it difficult to work with most of the machine learning models. The solution here is to reduce the dimensions of the matrix. After various trials detailed in Chapter 5 Results, the most appropriate method was found to take the standard deviation value of each segment, resulting in a feature vector of size 1xS for each keystroke. Then, these vectors were combined under a single matrix of size N x S to produce the data matrix. Code for normalization and feature extraction is given in Appendix B and C as "Normalizer" and "Feature Extractor", respectively.

4.4. Keystroke Classification

"Classification Learner", an application offered by Matlab, was used in the implementation of machine learning models. In Classification Learner app, one can explore his/her data, select features, specify validation schemes, train models, and assess results (Train Models to Classify Data Using Supervised Machine Learning - MATLAB, n.d.).

Working with the Classification Learner app starts by creating a new session from the workspace. In the classification stage, after selecting the data matrix resulting from the feature extraction stage as the Data Set Variable, the label prepared for this matrix was added to the response header from the workspace. Default values were preferred in the validation section. For validation 5-fold cross validation was applied.

Due to the high speed and accuracy it offers, the most suitable model was found to be the “Optimizable Discriminant” model. "Grid Search" was determined as the optimizer. Grid Search is an approach, which tests every unique combination of hyperparameters in the search space and returns the best performance (Liashchynskyi & Liashchynskyi, 2019). The number of grids was left at the default value of 10. For validation, 5-fold cross validation was applied. In the configuration of the optimizer parameters, the default values “Number of Grid Divisions” = 0 and “Training Time Limit” = Unchecked were selected. PCA option was not used as they did not improve performance drastically.

CHAPTER 5

RESULTS

5.1. Segmentation Results

One of the parameters affecting the keystroke detection and segmentation stages is the "frequency range of filter bank". In order to accurately detect the keystroke peak, the signal must be cleared of noise and the peaks must be made clear.

CWT processing was applied to the raw audio signal using filter banks configured with different frequency ranges. While lower frequency value candidates are selected as 100 Hz, 400 Hz and 1000 Hz; higher frequency candidates were determined as 6000 Hz, 9000 Hz, 12000 Hz and 24000 Hz. Trial results are plotted in Figures from 5.1 to 5.4.

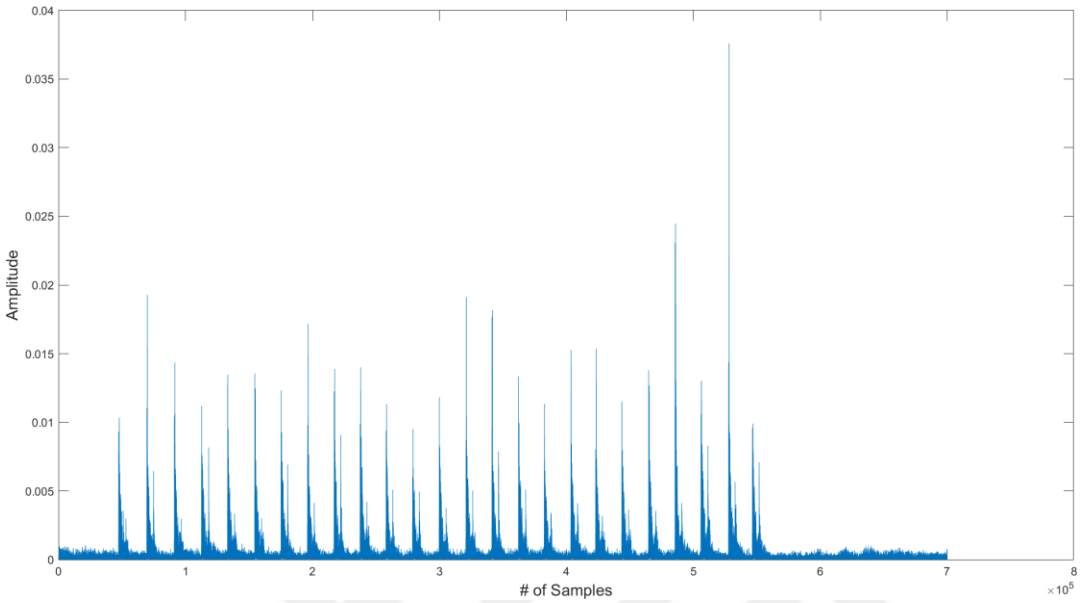


Figure 5.1. Plot of MeanRealCWTForDetection with filter bank frequency range from 100 Hz to 9000 Hz

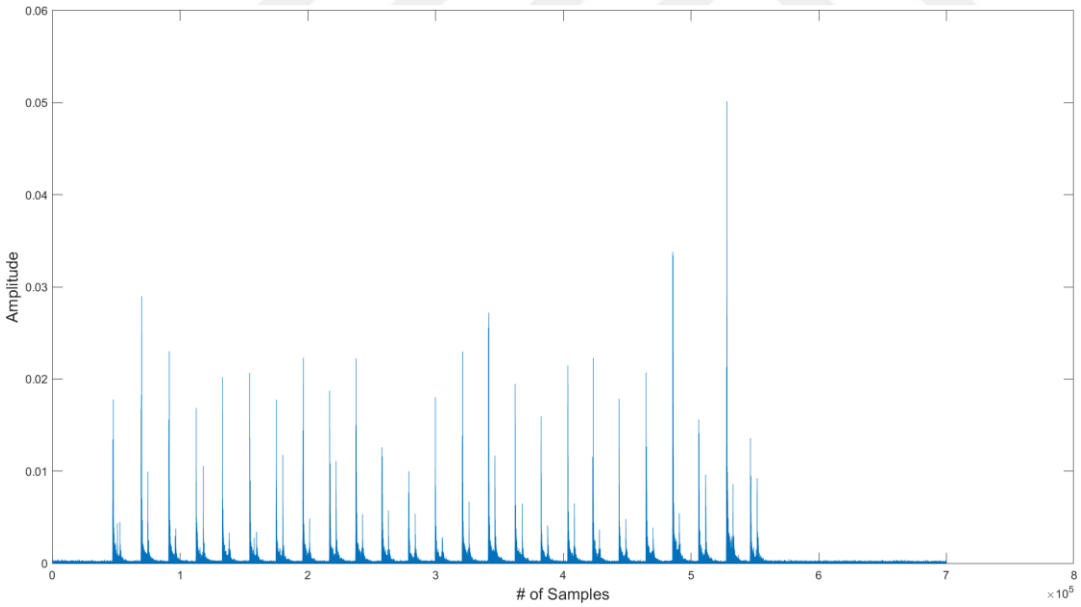


Figure 5.2. Plot of MeanRealCWTForDetection with filter bank frequency range from 1000 Hz to 9000 Hz

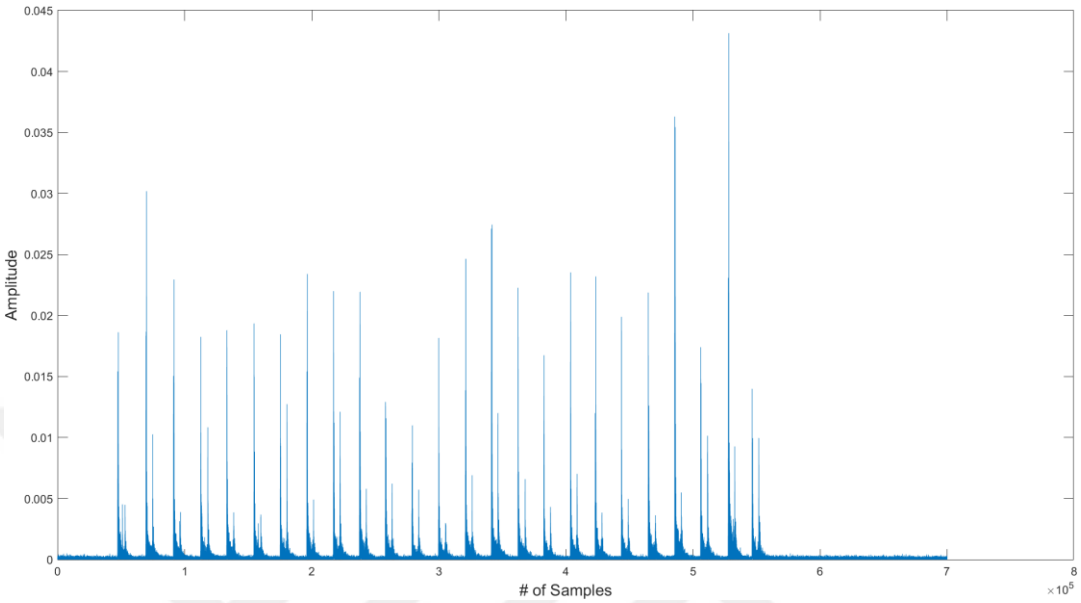


Figure 5.3. Plot of MeanRealCWTForDetection with filter bank frequency range from 400 Hz to 6000 Hz

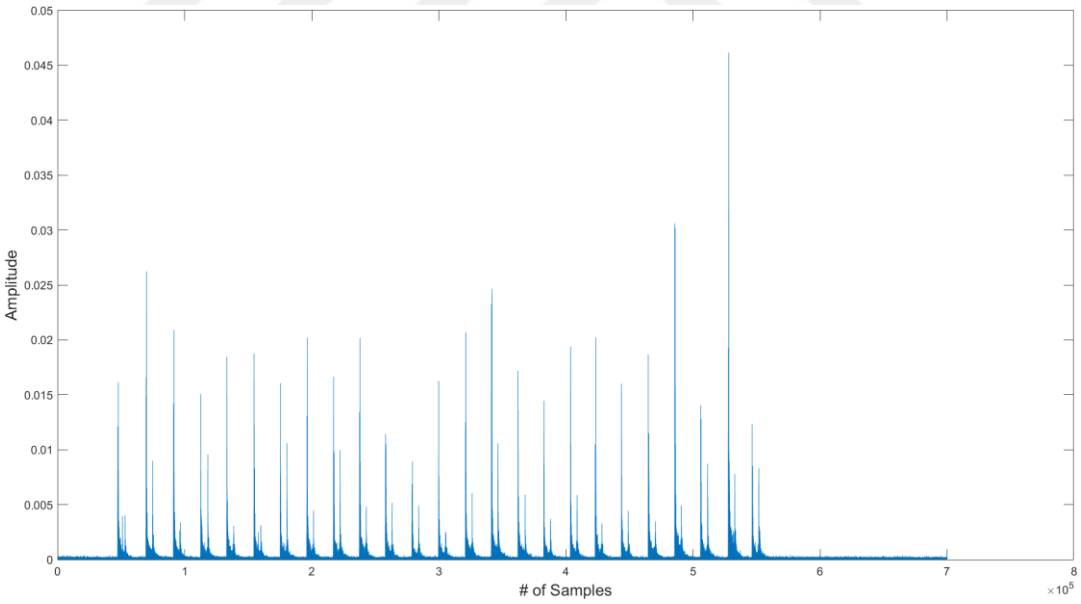


Figure 5.4. Plot of MeanRealCWTForDetection with filter bank frequency range from 400 Hz to 12000 Hz

When the lower frequency value for the filter bank drops below 400 Hz, it will be seen that noise starts to occur in the signal, and when the upper frequency value is above 9000 Hz, there will be a decrease in the peak amplitude value. Therefore, the optimum frequency range was found to be between 400 Hz and 9000 Hz.The plot of this trial is given in Figure 5.5.

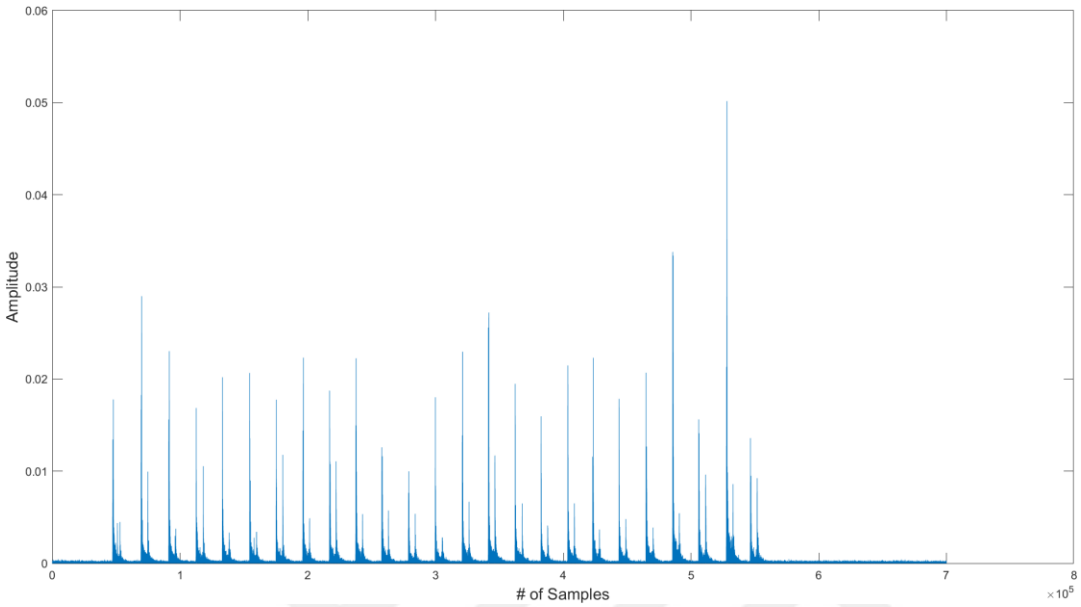


Figure 5.5. Plot of signal obtained after processing with CWT with frequency range from 400 Hz to 9000 Hz

Only the real values of the CWT output were used, and then their absolute values were calculated. In order to reduce the dimension, the resulting values were averaged across the dimension of scales.

This processing made it easier to detect the keystrokes with the peak detection algorithm mentioned in Chapter 4. Using this technique all the keystrokes in the datasets were successfully detected, which was verified by manual checking.

5.2. Feature Extraction Results

As we stated earlier, machine-learning models have difficulty working with 3D matrices. Our solution at this point is to reduce the size of the matrix. It was decided to reduce it by taking either the mean, maximum or standard deviation values across the time dimension. The dimension reduction technique can affect the classification performance obtained at the end of the day. Indeed, taking the standard deviation has been found to give the best performance among the three.

The records in the TwentyFives dataset were divided into train/test datasets by splitting the keystrokes by 20/5 for each character.

When standard deviation was used to reduce dimension, the validation accuracy was 89.3%, and the test accuracy was 94.7%. The confusion matrices are given in Figure 5.6 and Figure 5.7 for validation and test evaluations, respectively.

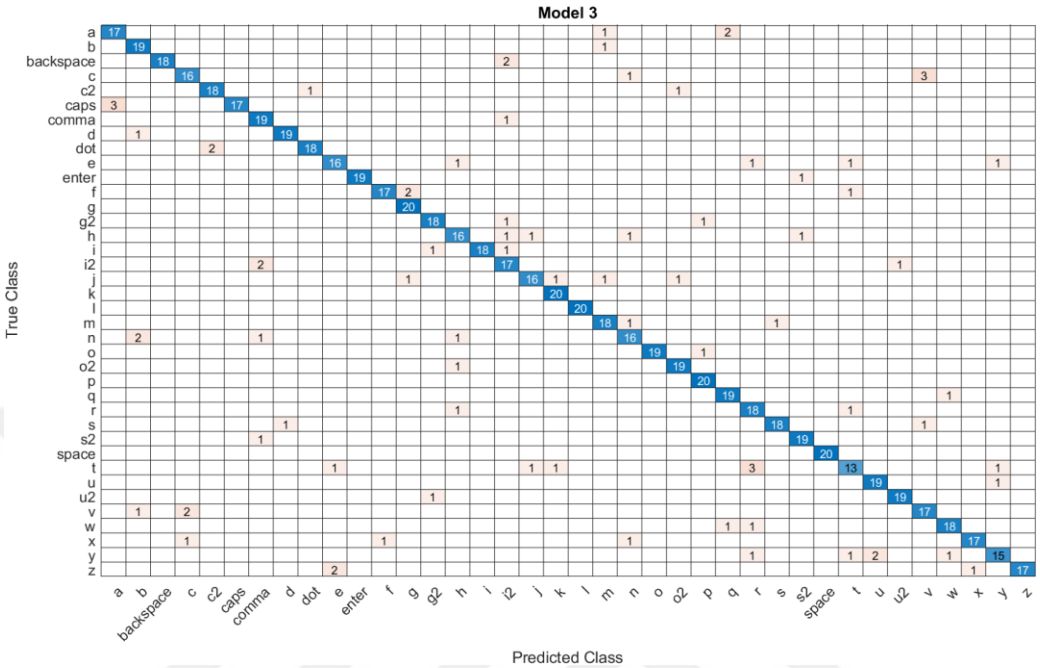


Figure 5.6. Confusion matrix of validation while using the standard deviation to reduce dimensionality

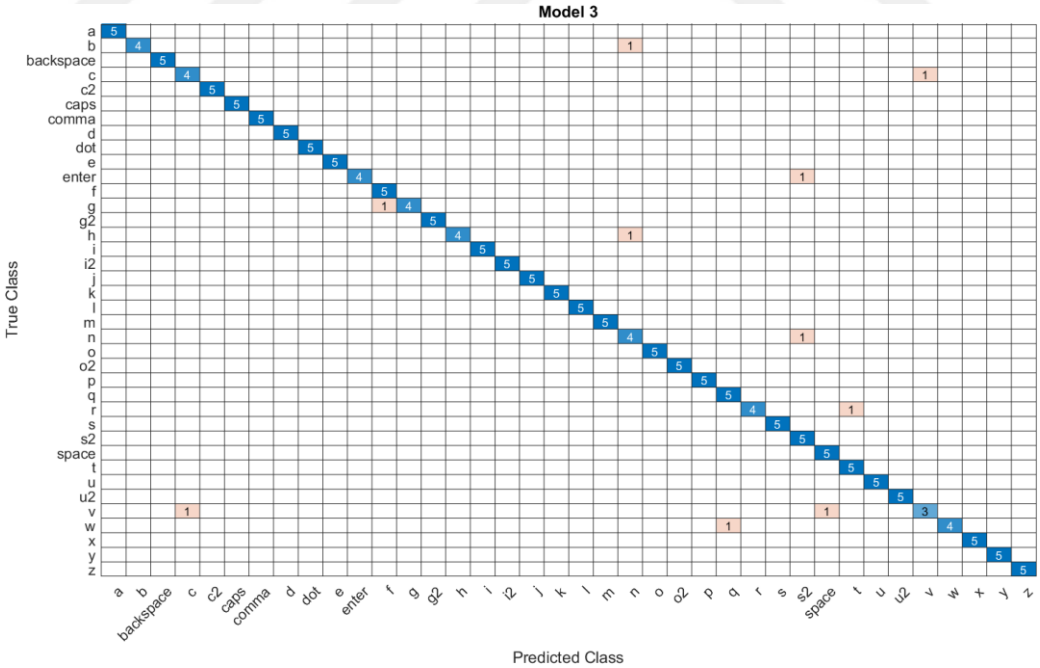


Figure 5.7. Confusion matrix of test while using the standard deviation to reduce dimensionality

When mean was taken to reduce dimension, the validation and test accuracies were 83.0% and 86.3%, respectively.

When maximum values were used to reduce dimension, the validation and test accuracies were 78.2% and 81.6%, respectively. The confusion matrices of these two statistical features were shared in figures from 5.8 to 5.11.

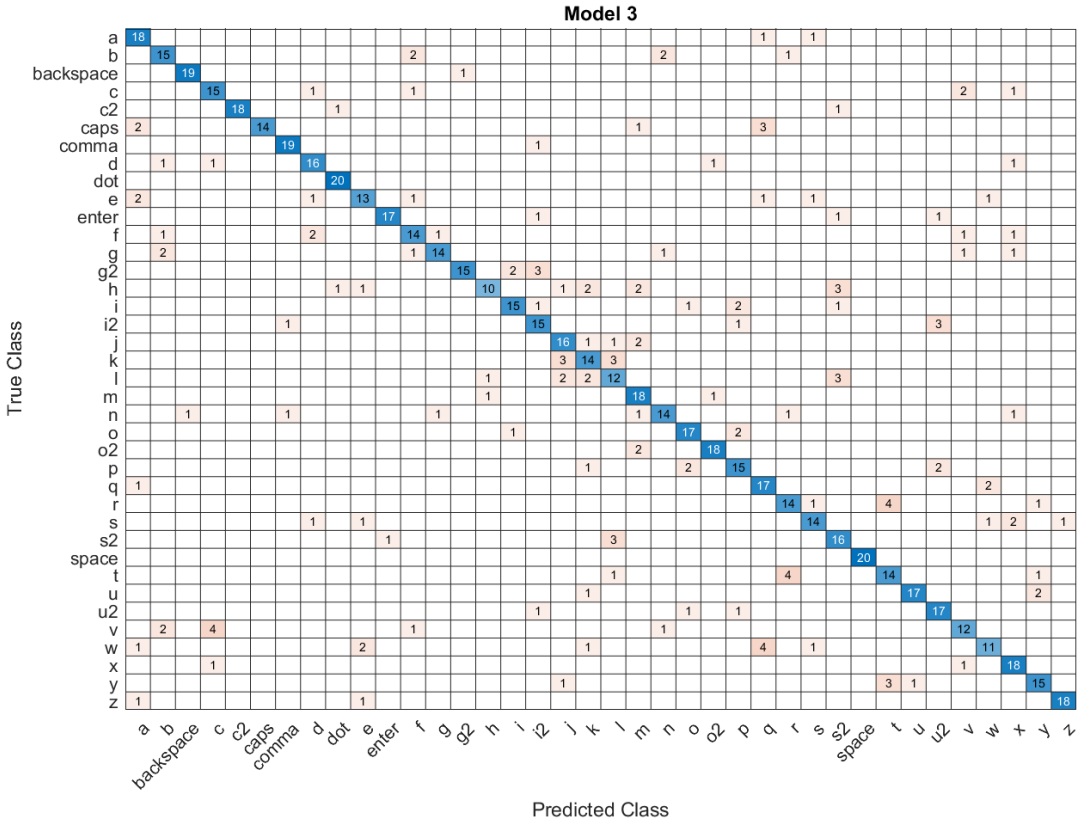


Figure 5.8. Confusion matrix of validation while using the maximum to reduce dimensionality

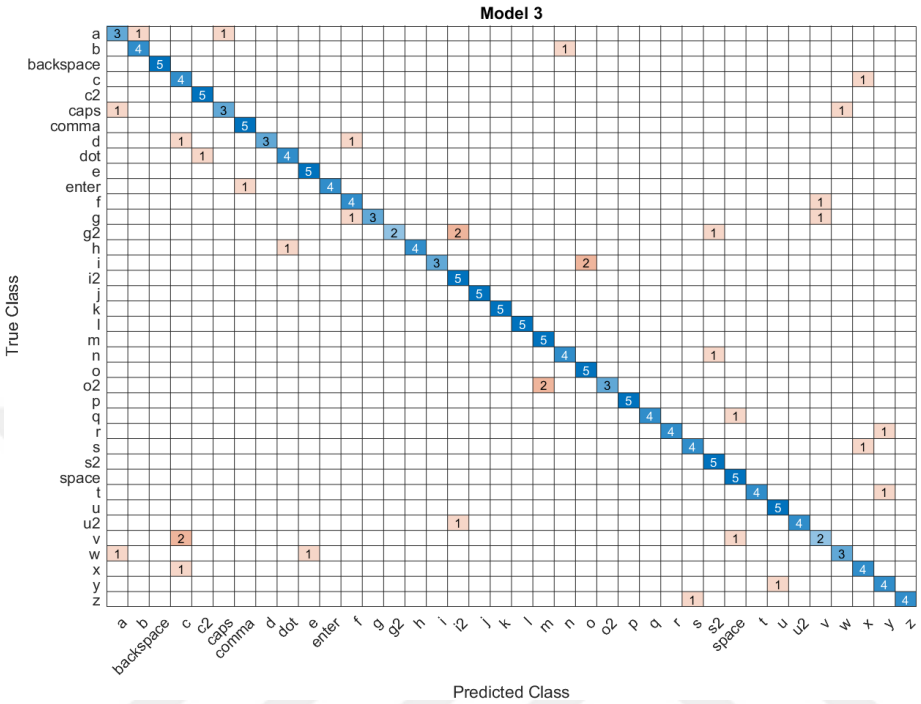


Figure 5.9. Confusion matrix of test while using the maximum to reduce dimensionality

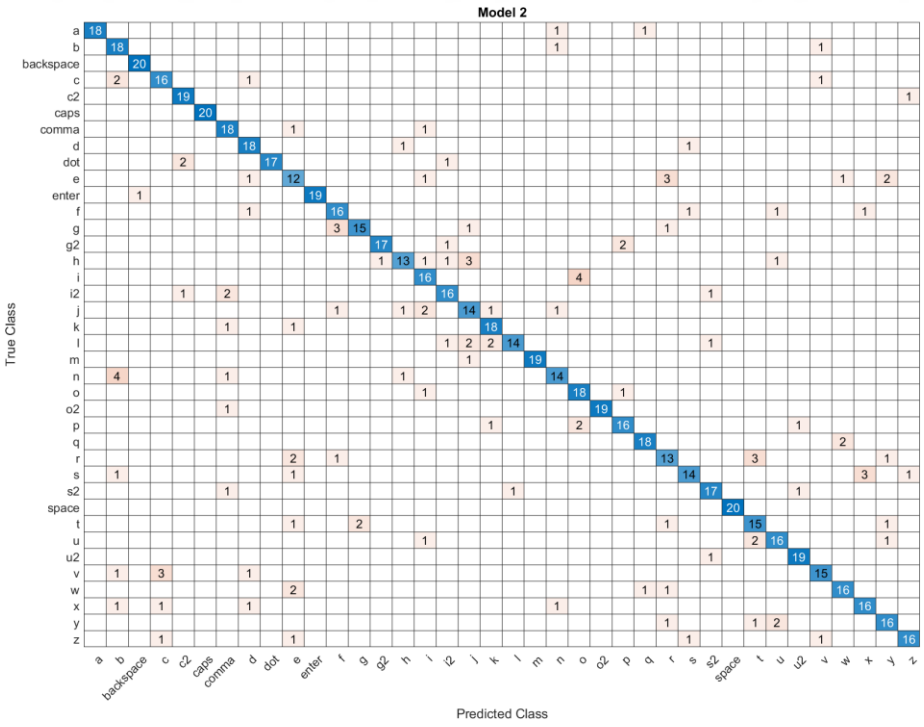


Figure 5.10. Confusion matrix of validation while using the mean to reduce dimensionality

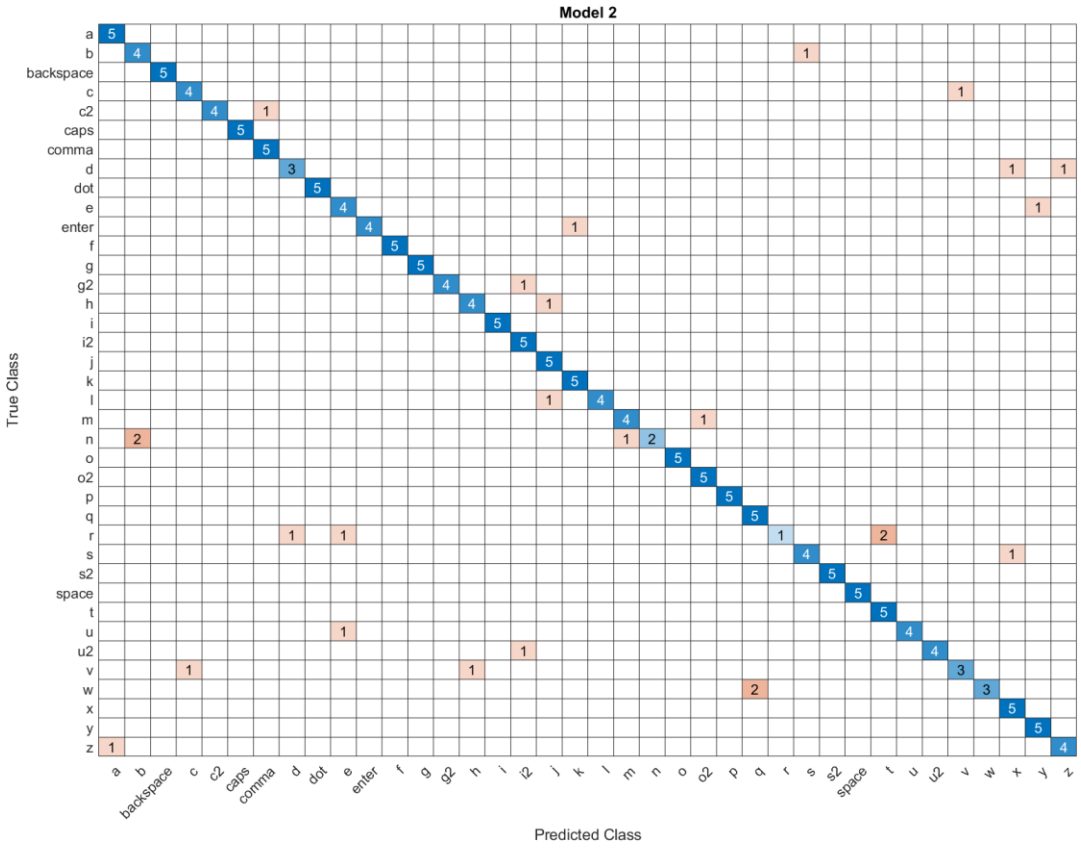


Figure 5.11. Confusion matrix of test while using the mean to reduce dimensionality

5.3. Classification Results

There are many models offered by the Classification Learner app. It was decided to work with “Optimizable” variants of all models. Optimizable models aim to achieve the highest performance by configuring the model's hyperparameters (Hyperparameter Optimization in Classification Learner App - MATLAB & Simulink, n.d.). Tested models were Tree, Discriminant, Naive Bayes, SVM, KNN, Ensemble and Neural Network. In order to compare the performances of these models, the datasets specified in Part 5 in Table 5.1 were used. Results are shared in Figure 5.12 and 5.13.

The datasets were used in five different ways in order to test the effect of the recording setup and the key press method. In the first part of the study, the records in the "TwentyFives", “Panagrams and ThreeStates” datasets were used. All of the “TwentyFives” records and 50% of the “Panagrams and ThreeStates” records were combined to form the training set. The remaining 50% of the “Panagrams and ThreeStates” records were reserved as the test set. When the training dataset was trained with Optimizable Discriminant method, the validation accuracy of 80.1% was found. Then, the test dataset was loaded, and the test accuracy was found to be 72.9%. Confusion matrices of all the phases for both training and test stages are in figures from 5.14 to 5.19.

Table 5.1 Validation and test accuracies of all optimizable models in classification learner for part 5

Models	*Validation Accuracy*	*Test Accuracy*
Optimizable Tree	35.8	27.6
Optimizable Discriminant	79.6	80.3
Optimizable Naïve Bayes	50.2	42.8
Optimizable SVM	72.8	67.1
Optimizable KNN	57.9	52.0
Optimizable Ensemble	50.4	38.8
Optimizable Neural Network	68.6	66.4

Table 5.2 Training and test datasets of the classification phases

Part	Training Dataset	Test Dataset	Validation Accuracy	Test Accuracy
1	TwentyFives + 50% of (Panagrams and ThreeStates)	Other 50% of (Panagrams and ThreeStates)	80.1	72.9
2	Hundreds + TwentyFives + 50% of (Panagrams and ThreeStates)	Other 50% of (Panagrams and ThreeStates)	72.4	22.4
3	80% of (Panagrams and ThreeStates)	20% of (Panagrams and ThreeStates)	71.5	70.4
4	TwentyFives	Panagrams and ThreeStates	91.3	35.7
5	TwentyFives + 80% of (Panagrams and ThreeStates)	20% of (Panagrams and ThreeStates)	79.6	80.3

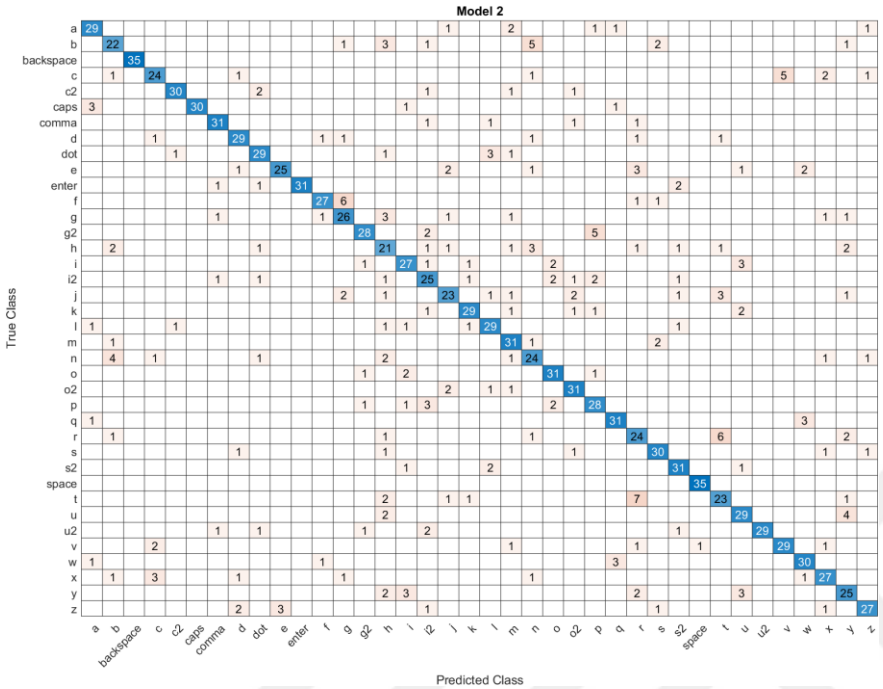


Figure 5.12. Confusion matrix of validation for part 1

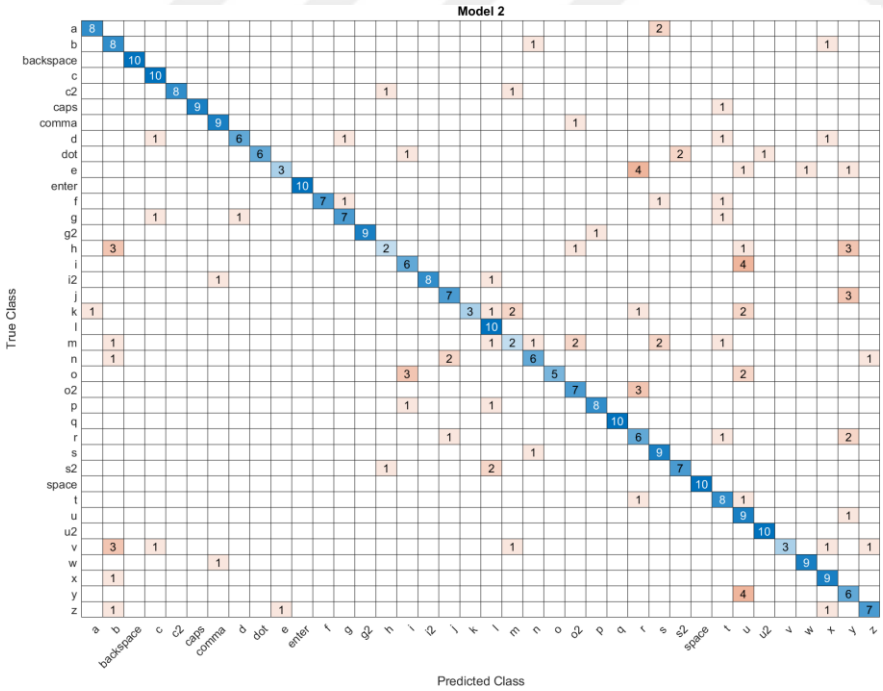


Figure 5.13. Confusion matrix of test for part 1

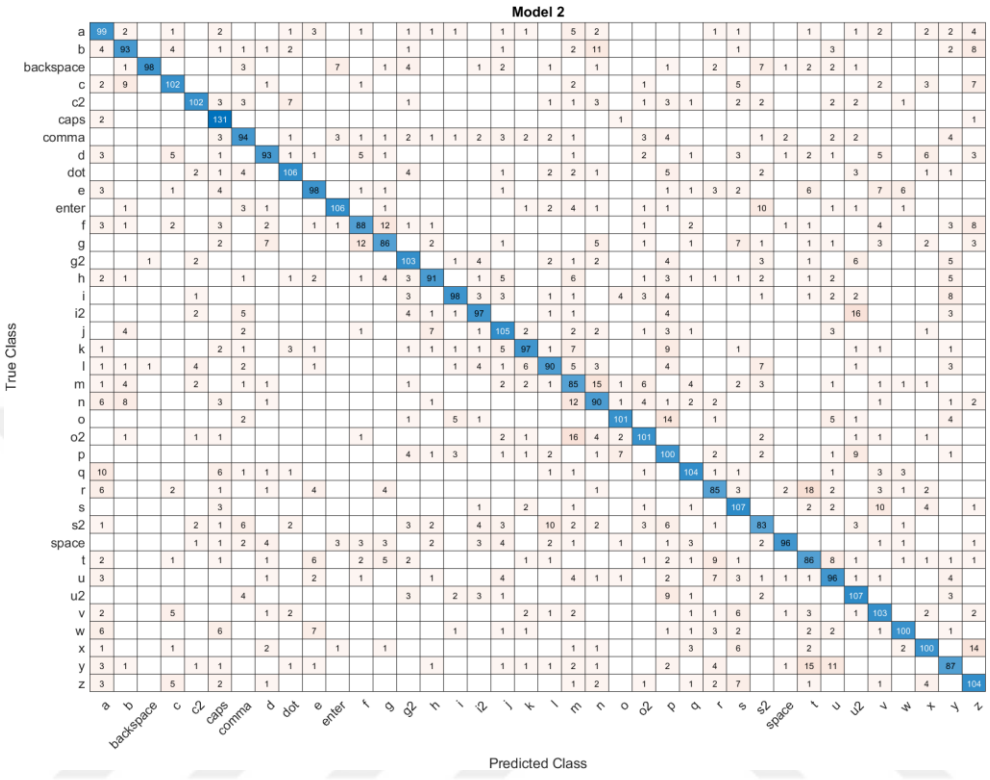


Figure 5.14. Confusion matrix of validation for part 2

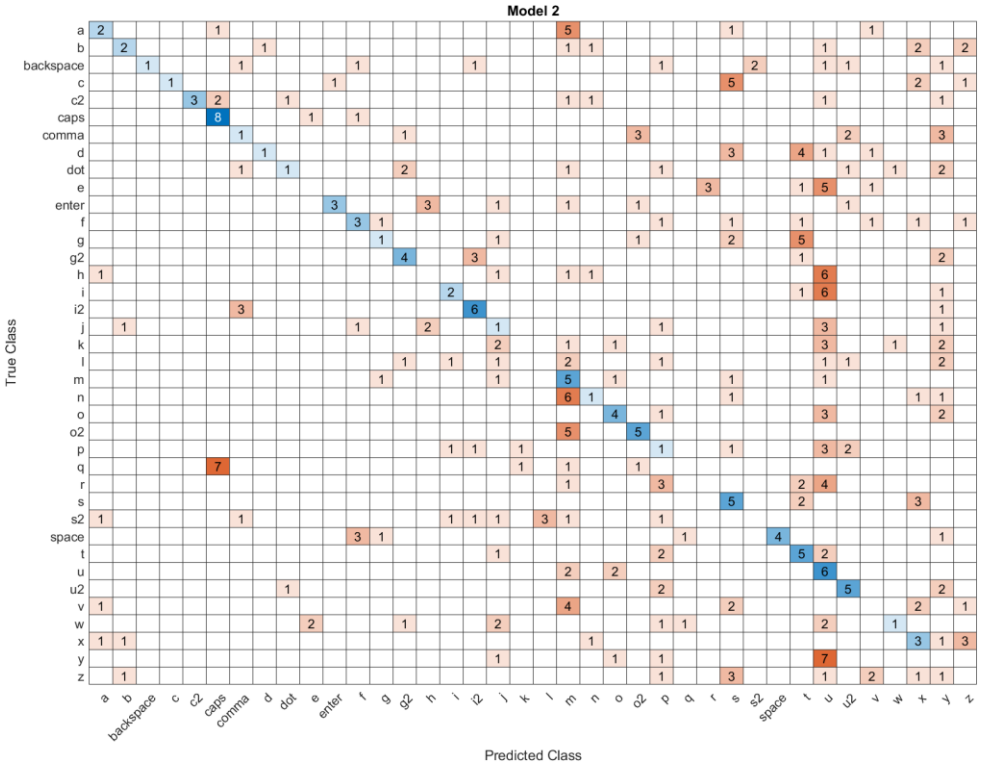


Figure 5.15. Confusion matrix of test for part 2

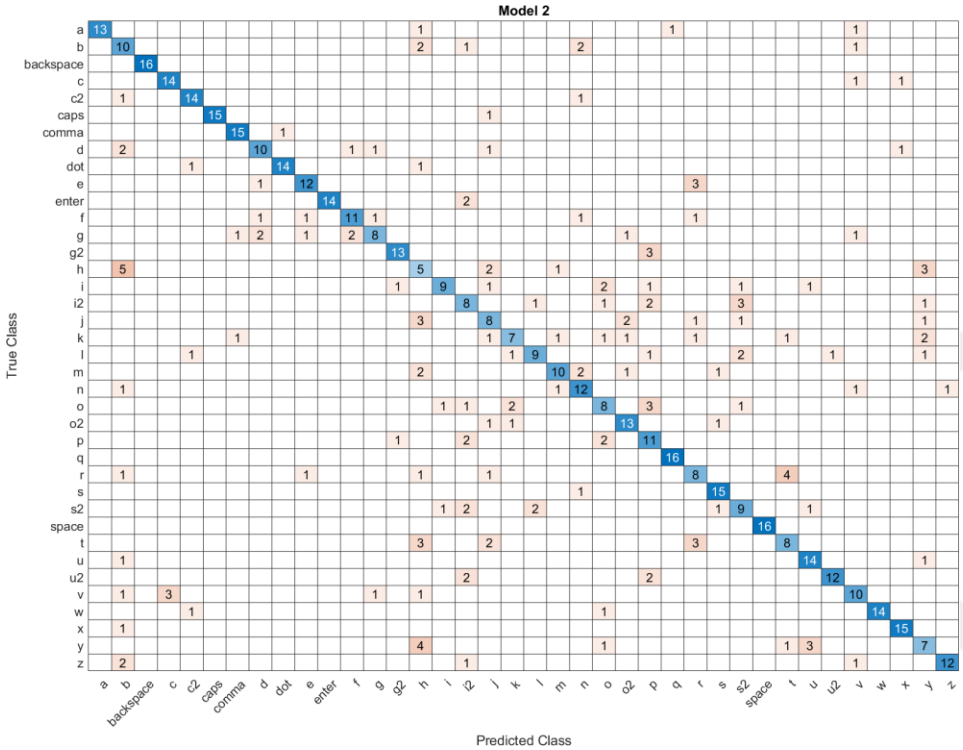


Figure 5.16. Confusion matrix of validation for part 3

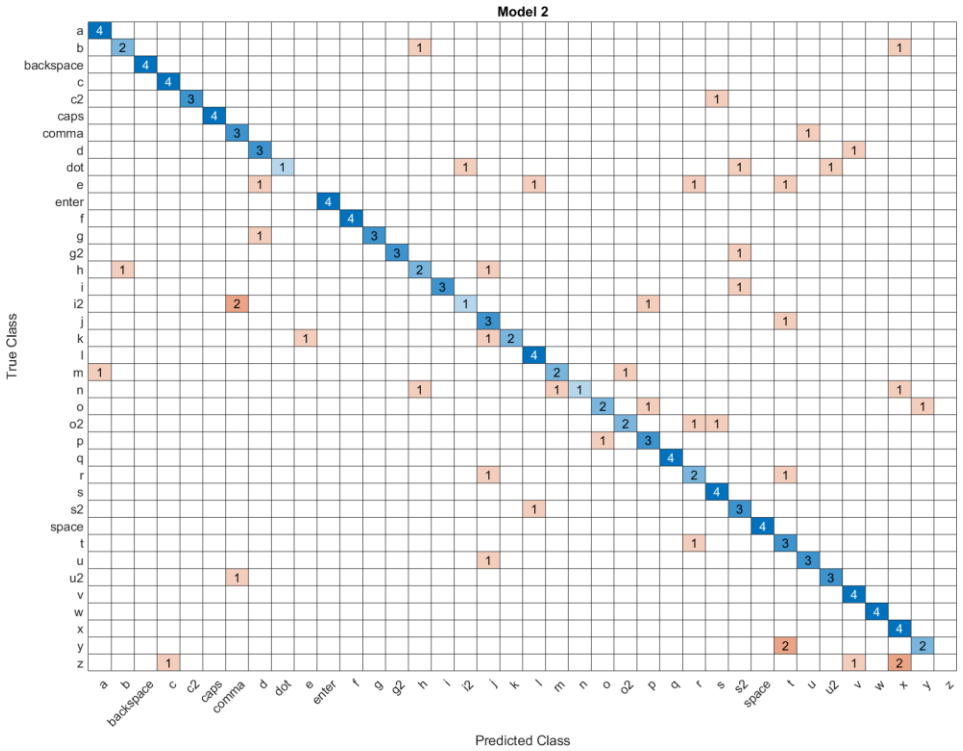


Figure 5.17. Confusion matrix of test for part 3

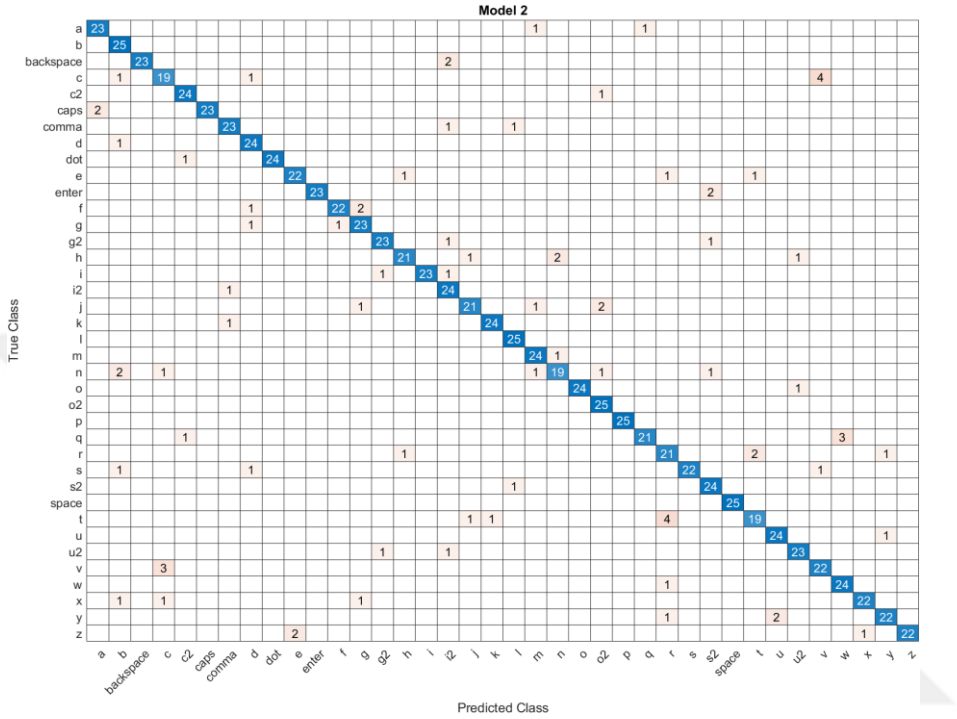


Figure 5.18. Confusion matrix of validation for part 4

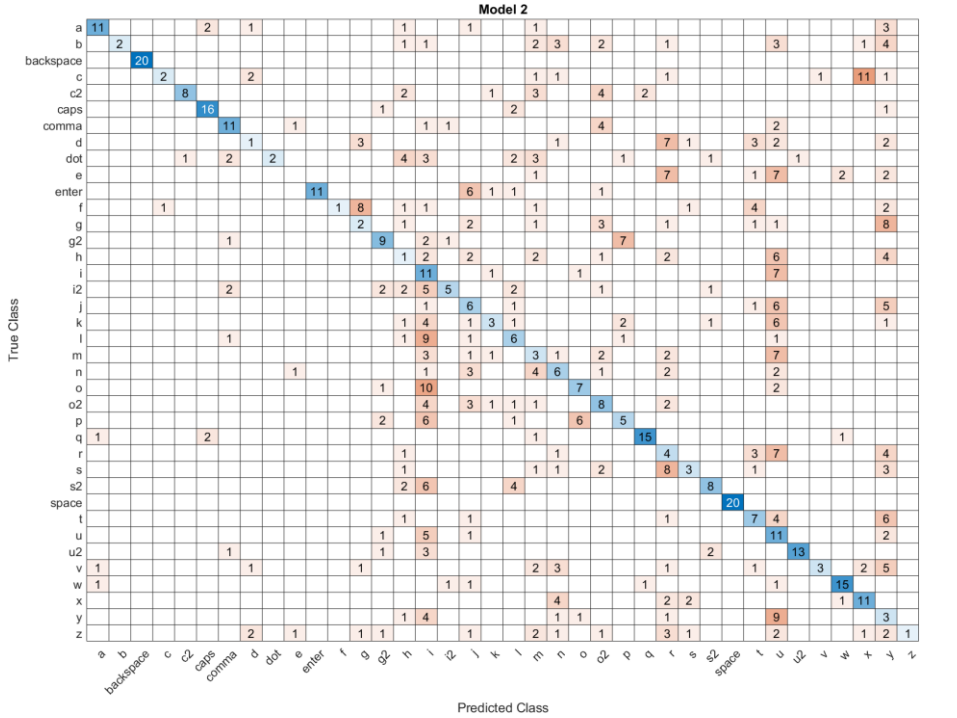


Figure 5.19. Confusion matrix of test for part 4

In the second part, all the records were used together. This time, all the records in the "Hundreds" dataset have been added to the training set created in the first stage; while the test dataset was used unchanged. When the training dataset was trained with Optimizable Discriminant, validation accuracy was found to be 72.4%. Then, by loading the test dataset, the test accuracy was found to be 22.4%.

In the third part of the study, the records in the "Panagrams and ThreeStates" dataset were used together. They were randomly divided into two as training and test datasets, at a ratio of 16/4. When the training dataset was trained with Optimizable Discriminant, the validation accuracy was 71.5%. Then, the test dataset was loaded, and the test accuracy was found as 70.4%.

For the fourth part, the training was made with only the “TwentyFives” dataset, while the testing was made with the “Panagrams and ThreeStates” dataset. Validation accuracy was 91.3%, while test accuracy was 35.7%.

In the last part of the study; the record in the “TwentyFives” dataset and 80% of “Panagrams and ThreeStates” dataset were combined for training. Test dataset consisted of the remaining 20% of the “Panagrams and ThreeStates” dataset. In this case, the validation accuracy was 79.6%, while the test accuracy was 80.3%.

CHAPTER 6

DISCUSSION

When the classification results performed with the optimizable discriminant model in Table 5.1 are examined, it can be seen that the model achieves higher accuracy in the records taken under the same conditions. The accuracy value obtained with part 1 datasets, all of which were recorded under the same conditions, decreased when a dataset recorded under different conditions was added to the training phase in part 2. Especially, it drastically reduced the accuracy obtained during the test phase.

Another conclusion that can be drawn from the table is that the typing style also has an effect on the results. In Part 3, both the training and test datasets consisted of key sounds typed in the "Hunt and Peck Typing" style. However, in part 4, the training dataset was in the style of "Straw Man Typing" and the test dataset was in the style of "Hunt and Peck Typing". For this case, the test accuracy decreased significantly compared to part 3.

Finally, although the model trained with one typing style does not show great success in the test of a different typing style; the presence of both typing styles at the training phase affects the results positively. In Part 3, the training and test datasets consisted of the same typing style. When a new dataset recorded with a different typing style was added to the training datasets in Part 5, it is seen that the accuracy values increase for both the training and test phases.

The amount of accuracy increase is also related to the split ratio of the dataset between training and testing. While the increase in test accuracy in part 1 was limited compared to part 3, the increase in test accuracy was higher in part 5, where most of the dataset was allocated to the training stage, compared to part 3.

6.1. Mitigations

Anand and Saxena (2018) made two suggestions against keyboard acoustic emanation attacks. The first of these, white noise, was produced with matlab and added to the entire recording for signal masking in the 400 Hz – 12 kHz frequency range. As a result of the study, they showed that white noise had no effect on the success of the model. While many

microphones and audio recording applications even offer background noise suppression by default, this result is not surprising.

The second recommendation is to use fake keystrokes. Fake keystrokes are real keystroke sounds that have been previously recorded. Anand and Saxena stated that fake keystrokes are not filtered in their study on Skype. Fake keystroke sound cannot be distinguished from real keystroke sound. This will cause the attacker model to identify keys that are not included in the text, and the language modeler enhancements will not work properly.

It can be protected from attack not only by adding artificial sounds, but also by removing the noises made by the keys. In this context, there is a need to test the model with the use of quiet keyboards. Quiet keyboards do not cause a mechanical release when used. Thus, the entered text will remain secure.

Finally, one of the most effective solutions is to leave the recording devices out, not to take them into places where sensitive data is transferred using the keyboard. While applying this solution, it should not be forgotten that the laptop or webcam microphones are also disabled.

CHAPTER 8

CONCLUSIONS

In the thesis study, our first aim was to present the academy with a systematically prepared and labeled keystroke recordings dataset that can be used in keystroke transcription from acoustic emanations research. We believe that the datasets we have prepared in this direction are successful both because they contain different typing styles and because they have a very wide scope.

The work is also valuable because it demonstrates that continuous wavelet transform features can also be used in keystroke analysis, while presenting an original model for keystroke detection and feature extraction.

We have also shown the detrimental effect of using a training dataset that is not prepared in the same way as the test data on the performance. When the first and second stages of the classification study are examined, it can be seen that the inclusion of "Hundreds" records in the training dataset significantly reduces both validation and test accuracy. At this point, we can say that although the same keyboard and the same typist are used, the change in the typing style will negatively affect the performance of the attack. However in practice the training data can be collected while a typist is typing a known text, such as a known website address, their names, email addresses, etc., which would be matching the same typing style and environment for transcribing an unknown typed text.

Since the considered keys also include “backspace”, it is also possible to see the deleted characters and texts, during text editing, for example while a person is composing an email. This may be useful for analyzing the typist’s intentions and thoughts, even if they do not appear in the final text.

As a future work, the transcribed texts can further be corrected or improved by using dictionaries and language models. The number of letters in a written word is clearly understood, especially because of the high distinguishability of the "space", "enter", "caps lock" and "backspace" keys. The inclusion of the Turkish alphabet in the scope allows this study to be carried out for Turkish, a language that has not been considered in previous keystroke transcription studies.

It has also been observed in the study that certain keys can be confused with each other. In particular, this takes place between keys with close positions on the keyboard. The knowledge of frequently confused keys can be exploited together with a language model to increase the accuracy of transcribed text.

Shift key is not included in the scope of the study. Including this key in the scope is of great importance, especially for successful password attacks. The shift key is used when writing special characters and even when entering capital letters for some users. Cecconello et al. (2019) shared a tip for distinguishing the shift key they gave as future work. Accordingly, in order to detect the use of the shift key, the sections where a new key is inserted between the press and the release peak can be examined. In addition, our recommendation is to treat the keys pressed while the shift key is pressed as a separate key group from the keys pressed alone. The key tones are distinguishable because they vibrate the keyboard base differently due to their position. Since the vibration of the keyboard will change while the shift key is pressed, we anticipate that the same key pressed alone and the same key pressed while the shift key is pressed will produce different sounds.

The use of different microphone types and different numbers of microphones is also worth examining. In the study, one and only one modeled microphone was recorded. Asonov and Agrawal (2004) tried different microphone models but used this to see the effect of distance on the model's success. The effects of recordings taken from various devices with microphones used in attack scenarios such as laptop, webcam, mobile phone on the model can be examined. The increase in the performance of the model with the use of a large number of microphones is also one of the possible results.

The future works proposed so far have been to increase the success of the attack. Another point of view that is at least as important as this one is on how to minimize being affected by this attack. At this point, since we share the codes and dataset produced in the study; proposed mitigations can be tested directly as a continuation of this study. The results produced in the current study showed how easily keystroke emanations can be used and how large data leaks they can cause. These results are an important source of motivation for working on mitigations. We hope that the mitigations to be determined by future works will be made available to make the data more secure.

REFERENCES

Anand, S. A., & Saxena, N. (2018). Keyboard emanations in remote voice calls: Password leakage and noise(less) masking defenses. *CODASPY 2018 - Proceedings of the 8th ACM Conference on Data and Application Security and Privacy*, *2018-January*, 103–110. https://doi.org/10.1145/3176258.3176341

Asonov, D., & Agrawal, R. (2004). Keyboard acoustic emanations. *Security and Privacy, 2004. Proceedings. 2004 IEEE Symposium on Security and Privacy,* 3-11.

Berger, Y., Wool, A., & Yeredor, A. (2006). Dictionary attacks using keyboard acoustic emanations. *Proceedings of the 13th ACM conference on computer and communications security,* 245-254.

Bhunia, S., & Tehranipoor, M. (2019). Side-channel attacks. *Hardware Security* (p. 193–218). Morgan Kaufmann. https://doi.org/10.1016/b978-0-12-812477-2.00013-7

Cecconello, S., Conti, M., Lain, D., Zurich, E., & Gene Tsudik, S. (2019, November). Skype & Type: Keyboard Eavesdropping in Voice-over-IP. *ACM Transactions on Privacy and Security*, *22*, 1-34.

Compagno, A., Conti, M., Lain, D., & Tsudik, G. (2017). Don't Skype & Type! Acoustic Eavesdropping in Voice-Over-IP. *ASIA CCS '17: Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security,* 703-715.

Cortes, Corinna; Vapnik, Vladimir (1995). Support-vector networks. *Machine Learning. 20 (3),* 273–297. *doi:10.1007/BF00994018*

Fisher, R. A. (1936). The use of multiple masurements in taxonomic problems. *Annals of Eugenics, 7 (2).* 179–188. doi:10.1111/j.1469-1809.1936.tb02137.x

Find local maxima - MATLAB findpeaks. (n.d.). Retrieved August 5, 2022, from https://www.mathworks.com/help/signal/ref/findpeaks.html

Fit binary decision tree for multiclass classification - MATLAB fitctree. (n.d.). Retrieved August 16, 2022, from https://www.mathworks.com/help/stats/fitctree.html

Fit discriminant analysis classifier - MATLAB fitcdiscr. (n.d.). Retrieved August 16, 2022, from https://www.mathworks.com/help/stats/fitcdiscr.html

Fit ensemble of learners for classification - MATLAB fitcensemble. (n.d.). Retrieved August 16, 2022, from https://www.mathworks.com/help/stats/fitcensemble.html

Fit k-nearest neighbor classifier - MATLAB fitcknn. (n.d.). Retrieved August 16, 2022, from https://www.mathworks.com/help/stats/fitcknn.html

Halevi, T. & Saxena, N. (2012). A closer look at keyboard acoustic emanations: random passwords, typing styles and decoding techniques. *Proceedings of the 7th ACM Symposium on Information*, *2012-May*, 89-90.

Halevi, T., & Saxena, N. (2015). Keyboard acoustic side channel attacks: exploring realistic and security-sensitive scenarios. *International Journal of Information Security*, *14*(5), 443–456. https://doi.org/10.1007/s10207-014-0264-7

Hand, D. J., & Yu, K. (2001). Idiot's Bayes—not so stupid after all?. *International statistical review, 69(3),* 385-398.

Hyperparameter Optimization in Classification Learner App - MATLAB & Simulink. (n.d.). Retrieved August 8, 2022, from https://www.mathworks.com/help/stats/hyperparameter-optimization-in-classification-learner-app.html

Kelly, A. (2010). *Cracking Passwords using Keyboard Acoustics and Language Modeling* (Corpus ID: 14079461) [Master's thesis, University of Edinburgh].

Lévesque, L. (2014). Nyquist sampling theorem: Understanding the illusion of a spinning wheel captured with a video camera. *Physics Education*, *49*(6), 697–705. https://doi.org/10.1088/0031-9120/49/6/697

Levenshtein, V. (1966). Binary codes capable of correcting deletions, insertions, and reversals. In *Soviet physics doklady*, *10.* 707–710.

Mneney, S. H. (2008). *An introduction to digital signal processing: A focus on implementation*. River Publishers.

Opitz, D., & Maclin, R. (1999). Popular ensemble methods: An empirical study. *Journal of Artificial Intelligence Research, 11,* 169–198. doi:10.1613/jair.614

Rao, K. S., & K E, M. (2017). *Speech recognition using articulatory and excitation source features*. Springer International Publishing. https://doi.org/10.1007/978-3-319-49220-9

Shumailov, I., Simon, L., Yan, J., & Anderson, R. (2019). *Hearing your touch: A new acoustic side channel on smartphones*.

https://doi.org/10.48550/arXiv.1903.11137

Side-Channel Attack - Glossary | CSRC. (n.d.). Retrieved August 11, 2022, from https://csrc.nist.gov/glossary/term/side_channel_attack

Slater, D., Novotney, S., Moore, J., Morgan, S., & Tenaglia, S. (2019). Robust keystroke transcription from the acoustic side-channel. *ACM International Conference Proceeding Series*, 776–787. https://doi.org/10.1145/3359789.3359816

Teo, K. R., Balamurali, B. T., Ming, C. J., & Zhou, J. (2021, January 30). Retrieving Input from Touch Interfaces via Acoustic Emanations. *2021 IEEE Conference on Dependable and Secure Computing, DSC 2021*. https://doi.org/10.1109/DSC49826.2021.9346271

Torrence, C., & Compo, G. (1998). A practical guide to wavelet analysis. *Bulletin of the American Meteorological Society. 79 (1),* 61–78.

Train models to classify data using supervised machine learning - MATLAB. (n.d.). Retrieved August 5, 2022, from https://www.mathworks.com/help/stats/classificationlearner-app.html

Train multiclass naive Bayes model - MATLAB fitcnb. (n.d.). Retrieved August 16, 2022, from https://www.mathworks.com/help/stats/fitcnb.html

Train support vector machine (SVM) classifier for one-class and binary classification - MATLAB fitcsvm. (n.d.). Retrieved August 16, 2022, from https://www.mathworks.com/help/stats/fitcsvm.html

Wang, J., Ruby, R., Wang, L., & Wu, K. (2016). Accurate Combined Keystrokes Detection Using Acoustic Signals; Accurate Combined Keystrokes Detection Using Acoustic Signals. *2016 12th International Conference on Mobile Ad-Hoc and Sensor Networks (MSN)*. https://doi.org/10.1109/MSN.2016.9

Zhuang, L., Zhou, F., & Tygar, J. D. (2005). Keyboard Acoustic Emanations Revisited. *Proceedings of the 12th ACM Conference on Computer and Communications Security,* 373-382.

Zhu, T., Ma, Q., Zhang, S., & Liu, Y. (2014). Context-free attacks using keyboard acoustic emanations. Proceedings of the ACM Conference on Computer and Communications Security, 453–464. https://doi.org/10.1145/2660267.2660296

APPENDICES

APPENDIX A

KEYSTROKE EXTRACTOR

```
function [RawKeystrokes, Fs] = KeystrokeExtractorV03 (AudioPath)
    % Reading Audio
    [Audio, Fs] = audioread(AudioPath);
    SignalLength = length(Audio);
    % Keystroke Detection
    FbDetection = cwtfilterbank('SignalLength', SignalLength, 'SamplingFrequency', Fs,
'FrequencyLimits', [400 9000], 'wavelet', 'amor');
    [CWTAudioForDetection, ~] = cwt(Audio, 'Filterbank', FbDetection);
    MeanAbsRealCWTAudioForDetection = mean(abs(real(CWTAudioForDetection)));
    [~, Locations] = findpeaks(MeanAbsRealCWTAudioForDetection,
"MinPeakHeight",1/300 , "MinPeakDistance", Fs/4);
    % Segmentation
    SegmentSize= 1920;
    SegmentLengthBeforePress = 5*SegmentSize/4;
    SegmentLengthAfterPress = 25*SegmentSize/4;
    RawKeystrokes = zeros(length(Locations),
SegmentLengthBeforePress+SegmentLengthAfterPress);
end
```

APPENDIX B

NORMALIZER

```
function [NormKeystrokes, EnergyLevels] = NormalizeRawAudioV03
(RawKeystrokes)
   NormKeystrokes = zeros(length(RawKeystrokes(:,1)), length(RawKeystrokes(1,:)));
   EnergyLevels = zeros(length(RawKeystrokes(:,1)), 1);
   for i = 1:length(RawKeystrokes(:,1))
      TempRaw = RawKeystrokes(i,:);
      TempNorm = TempRaw/sqrt(mean(TempRaw.^2));
      TempEnergy = sqrt((mean((TempNorm).^2)));
      NormKeystrokes(i,:) = TempNorm;
      EnergyLevels (i) = TempEnergy;
   end
end
```

APPENDIX C

FEATURE EXTRACTOR

```
function [MaxFeatureAggregated] = FeatureExtractionV03 (NormKeystrokes, Fs)
   Size = size(NormKeystrokes);
   SignalLength = Size(2);
   FbSegmentation = cwtfilterbank('SignalLength', SignalLength, 'SamplingFrequency',
Fs, 'FrequencyLimits', [100 24000], 'wavelet', 'amor');
   for i = 1:Size(1)
      [Feature, ~] = cwt(NormKeystrokes(i,:), 'Filterbank', FbSegmentation);
      Feature = abs(Feature);
      MaxFeature = std(Feature');
      if i==1
         ScaleNumber = length(Feature(:,1));
         MaxFeatureAggregated = zeros(Size(1),ScaleNumber);
      end
      MaxFeatureAggregated(i,:) = MaxFeature;
   end
end
```