Final Year Project Report
on

Authentication using Keystroke Dynamics

submitted in fulfillment of
the requirement of the Degree of Bachelor of Technology

by

Sahil Pavaskar
(171090070)

Abhishek Myana
(171090023)

Omkar Chavan
(171090067)

Ashutosh Srivastava
(171090003)

Sagar Udasi
(171060088)

Under the guidance of

Dr. Rohin D Daruwala

DEPARTMENT OF ELECTRICAL ENGINEERING
VEERMATA JIJABAI TECHNOLOGICAL INSTITUTE
(An Autonomous Institute Affiliated to Mumbai University)
Matunga, MUMBAI – 400019

A.Y. 2020-2021

DECLARATION OF STUDENT

I declare that the work embodied in this Project titled **"Authentication using Keystroke Dynamics"** comes from our group's contribution under the guidance of Dr. Rohin D Daruwala at the Department of Electrical Engineering, Veermata Jijabai Technological Institute, Mumbai. Wherever we have used the work of any existing resource, the citations and the corresponding references are mentioned.

–

Sahil Pavaskar
ID: 171090070
Date:
Place: VJTI, Mumbai

Abhishek Myana
ID: 171090023
Date:
Place: VJTI, Mumbai

Omkar Chavan
ID: 171090067
Date:
Place: VJTI, Mumbai

Ashutosh Srivastava
ID: 171090003
Date:
Place: VJTI, Mumbai

Sagar Udasi
ID: 171060088
Date:
Place: VJTI, Mumbai

ACKNOWLEDGEMENT

Taking the challenging task and then finishing it satisfactorily requires the support and faith in the team from the mentor. This project would not have been finished without the help and support of the below-mentioned people who aided us with their efforts and blessings.

We start by showing our gratitude to our project mentor and the dean of student activities, Dr. Rohin D Daruwala. He has been a constant force in guiding us throughout the project. He provided us with the suggestions that kept our project develop seamlessly.

We would like to thank the B.Tech EXTC students of 2017-2021 batch for being supportive and helping us in making the dataset of the typing patterns required. As we know that without the data the entire pattern recognition and analysis is of no use, we are eternally owed to our batchmates who bridged the gap of the theoretical algorithms and the practical results by providing us with the required data.

Thank you all for making this journey so memorable.

APPROVAL SHEET

The credit report ***"Authentication using Keystroke Biometrics"*** by Sahil Pavaskar, Abhishek Myana, Omkar Chavan, Ashutosh Srivastava and Sagar Udasi is found to be satisfactory and is approved for the Degree of Bachelor of Technology.

Dr. Rohin D Daruwala
Supervisor

Examiner

Examiner

Examiner

Place: *Veermata Jijabai Technological Institute, Mumbai*
Date: / /2021

CERTIFICATE

This is to certify that Sahil Pavaskar (171090070), Abhishek Myana (171090023), Omkar Chavan (171090067), Ashutosh Srivastava (171090003) and Sagar Udasi (171060088), students of B.Tech (Electronics and Telecommunication), Veermata Jijabai Technological Institute, Mumbai have successfully completed the project titled ***"Authentication using Keystroke Biometrics"*** under the guidance of Dr. Rohin D Daruwala.

Dr. Rohin D Daruwala
Supervisor

Dr. Faruk Kazi
Head of Electrical Department

Place: *Veermata Jijabai Technological Institute, Mumbai*
Date: / /2021

Contents

Chapter 1

SYNOPSIS

1.1 Abstract

This paper discusses various aspects of authenticating a person using *keystroke dynamics.* The methods proposed by earlier literary works are revisited and suggested parameters are studied. Support Vector Machine (SVM) and Recurrent Neural Network (RNN) models are trained and compared on the *CMU keystroke dynamics benchmark dataset.* The similar analysis is then done on the custom dataset. Also, the idea of classifying the users based on *FAR (False Acceptance Rates) and FRR (False Rejection Rates)* is discussed.

1.2 Technical Keywords

Keystroke dynamics, Biometric Authentication, Behavioral Biometrics.

Chapter 2

ABOUT THE PROJECT

2.1 Introduction

Authentication is defined as an act of verifying the validity of the user on a computer-system. This authentication can be done using one or more of the following factors [1]:

a) *Knowledge-based factors:* The identification method is based on *what user knows.* This involves asking user some form of password or answer to the secret question, etc.
b) *Possession-based factors:* The identification method is based on *what user has.* This involves sending an OTP to ensure that the user is with their own device.
c) *Inherence-based factors:* This identification method is based on *what the user is.* This may involve checking of some physiological traits such as fingerprints, iris, DNA, etc., or some behavioral traits such as signatures, voice or typing patterns.

2.1.1 Biometrics

Biometrics refer to measurable human characteristics to define and identify the person uniquely. It can be broadly classified into two types: *physiological* and *behavioral. Physiological biometrics* involve the physical traits like fingerprints, iris, DNA, etc., whereas *behavioral biometrics* are based on human habits. Signature recognition, Voice recognition and Keystroke dynamics are types of behavioral biometrics. The use of behavioral biometrics, in conjunction with the password systems, not only improves the authentication mechanism but also reduces the need of long and unmemorable passwords.

According to the biometric recognition study [2], certain properties must be present in biometric features (also called biometric characteristics) in order to use those features in biometric systems and to be practical:

- **Universality:** each person should have the characteristics.
- **Distinctiveness:** different persons should be sufficiently different in terms of the characteristics.
- **Permanence:** stability of the characteristics over a period of time.
- **Collectability:** the characteristics can be measured quantitively.

The above properties are needed to be able to use the system and to make sure that the performance of distinguishing between different persons is achieved. Three properties are needed to make the system more practical and secure.

- **Performance:** refers to accuracy, speed and robustness in technology used.
- **Acceptability:** the degree of acceptance to use such a particular biometric identifier in the daily life.
- **Circumvention:** resistance of the system against fraudulent methods.

In general, a practical system should have all of these properties to ensure high accuracy and speed in accepting legitimate users and rejecting impostors. It is also necessary to prevent that multiple people use the same identity (positive recognition) and also prevent the same person from using multiple identities (negative recognition).

2.1.2 Biometric Systems

A biometric system is *"the automated identification or verification to human identity through repeatable measurement of physiological and/or behavioural characteristics"*.

It consists of two phases or subsystems. The first one is the enrolment phase where the biometric features are extracted from the user and converted into a template that will be stored in the system database in order to be used in the second phase. The second phase is the verification phase where the identity of the user is checked against the data obtained during the enrolment phase.

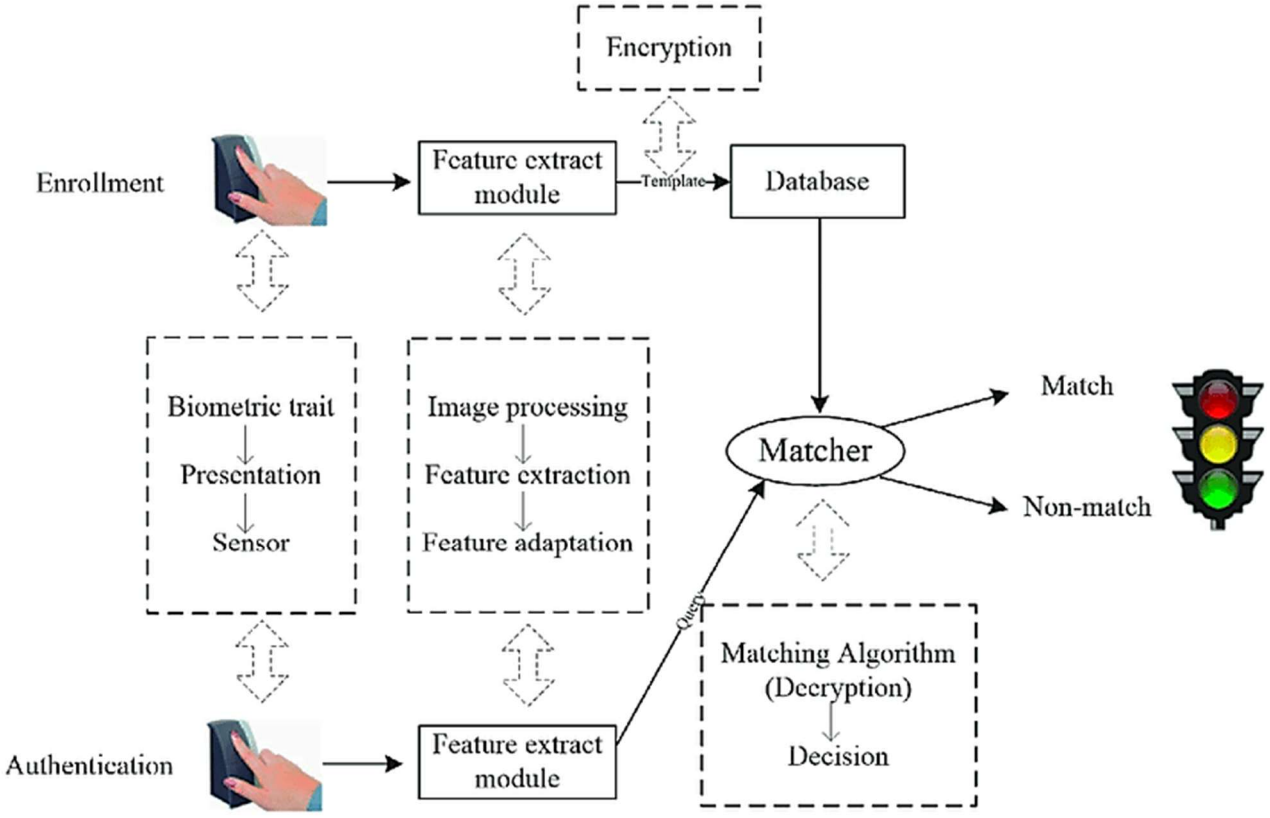


Fig. 2.1 – A Biometric System

During the enrolment phase the biometric features of the user are transformed into a template. This transformation is necessary for two reasons. First it is due to legal aspects as it is not expected to reveal significant information about the original data of the user. The second reason is that most of the biometric systems do not store raw biometric data because it can be unpractical since the template is used for comparison. In the authentication phase the user again presents his/her biometric characteristics which are extracted and then matched against the template(s) that correspond to the claimed identity of the user. A distance metric should be used to know how far/close the extracted features are from the template. Finally, a decision rule should be used to determine whether the user is rejected or accepted to the system. This decision depends on a predefined threshold value which is important to calculate some of the error types related to the biometric system.

2.1.3 Calculations and Error Types

Biometric systems certainly offer alluring advantages over other factors of authentication. While keys and passwords can be replicated or stolen, there is such a limited possibility in biometrics systems. Furthermore, the credit cards, ATM cards and other such provisions can't be misused, when accompanied with biometric test. Sharing of biometrics characteristics is not possible and thus, they can be used to prevent the same user from using two different identities (negative recognition). However, there is also a specific drawback attached to the biometric authentication. With something you know, either you know the secret or you do not know it. With something you have either the hardware fits or it does not fit, so it is 100% correct or 100% wrong. However, with something you are, the biometric features can never match 100%. For example, looking at a finger print, it will be some sample features that matches the template and other features that do not match the template. The more matches we find the more convinced we are that a legitimate user tries to authenticate himself to the system. The differences between the extracted sample for authentication and the template sample is giving by the distance value. This value is calculated by using a distance metric for example the sum of absolute distances between corresponding values in two sets (extracted set and template set).

The idea behind a distance metric is to give in principle a small intra-class value, meaning that sets from the same user have a low distance value, and a larger inter-class distance, meaning that sets from different users should give a high distance value. A decision rule, which depends on a predefined threshold should be used to decide whether a person is accepted or rejected. During this matching some errors might occur. Two types of important errors are:

- **False Match Rate (FMR):** This happens when a biometric system measures two different persons to be the same person. Obviously, an imposter wrongly will be accepted by the system. Also known as **False Positive Rate (FPR).**
- **False Non-Match Rate (FNMR):** This happens when a biometric system measures two different samples from the same person to be from a different person. A legitimate user is wrongly rejected by the system. Also known as **False Negative Rate (FNR).**

The trade-off between FMR and FNMR can be illustrated using the Receiver Operating Characteristics (ROC) or Decision Error Trade off (DET).

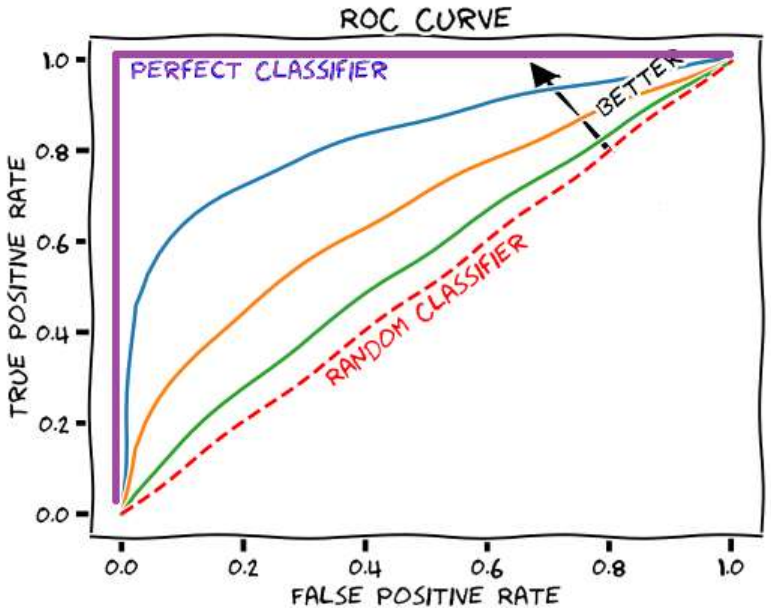


Fig. 2.2 – ROC curve

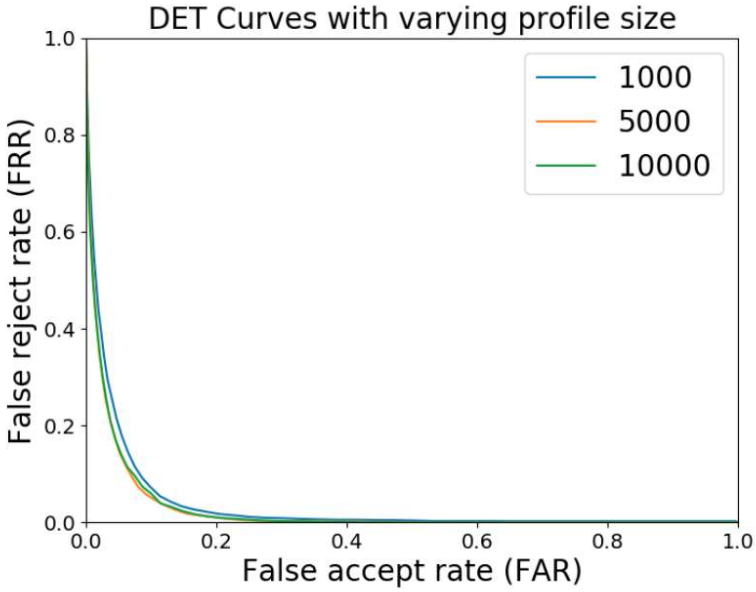


Fig. 2.3 – DET curve

Both curves show the system performance at different threshold values and the trade-off between FMR and FNMR. The difference between ROC and DET curves is that the DET curve plots false negatives (FNMR) on the Y-axis instead of true positives. While the ROC curve plots true positives (1-FNMR) instead of false negatives (FNMR).

Another important point is the Equal Error Rate (EER). This rate is used to compare different systems against each other and can give a brief idea about the performance of the biometric system. However as mentioned before accuracy of the system depends on the two errors FMR and FNMR which can be calculated as given.

$$FMR = \frac{Number\ of\ accepted\ impostor\ attempts}{Total\ number\ of\ impostor\ attempts}$$

$$FNMR = \frac{Number\ of\ rejected\ legitimate\ attempts}{Total\ number\ of\ legitimate\ attempts}$$

2.1.4 Keystroke Dynamics

Keystroke dynamics refers to the unique patterns of rhythm and timing-based features that are created when a user types on the touchscreen or on the traditional keyboard. The biometric system uses a pattern recognition system to classify the users based on their physical and behavioral characteristics. The typing dynamics gives the detailed timing information of when exactly each key was pressed and when it was released. These details form the primary features for the pattern recognition.

The major advantage of keystroke-based authentication is that they do not require any special device, as only a keyboard is enough. Moreover, the required keystroke information can be captured by the program running in the background without causing any inconvenience to the user by forcing them to do any additional task.

According to Hocquet & Cardot[3], several timings can be calculated from capturing the time when key is pressed down and the time when key is released up. For example, time between two key-downs or time between two key-ups or even time between key down and key up for the same key or time between key up for the first key and key down for the second key. The last two timings are the most popular among keystroke literatures. They are also known as duration and latency respectively.

Duration is the time a key is held down; it can be calculated by subtracting a key-down time from the key-up time for the same key. Latency is the time between two consecutive keystrokes which can be calculated by subtracting the key-up time for the first key from the key-down time for the second key.

The latency between two keys can be negative. For example, if the user presses the letter B before releasing the letter A. There are other definitions of latency, some use the key-down to key-down time. Our choice is to use the last approach of latency to assure that all latencies have positive values. The second part is needed to generate the templates or reference profiles. A user needs to create a template before he/she can use the system. This template contains a subset of all features which the user provided. Duration and latency are the basic to build such templates. Mean, standard deviation and the number of occurrences of the key is used in this template.

Finally, a distance metric is needed to make the comparison between the template and the new provided input data of the user. There are many different distance metrics that can be used. A property of a good distance metric is that it has a large inter-person distance and a small intrapersonal distance. One of a basic distance metric is the Euclidean distance, while Manhattan or Mahalanobis distances are the sophisticated metrics.

2.1.5 State of the art: State verification and Continuous verification

There are two types of keystroke dynamics. The first one is static keystroke dynamics in which the keystrokes are analysed only at specific times e.g., during login. The second one is continuous keystroke dynamics in which the typing characteristics are analysed during a complete session. Static approaches provide more robust user verification than simple passwords. However static methods do not provide continuous security, specifically they cannot detect substitution of the user after the initial verification.

2.1.6 Classification

Classification is a *machine learning/pattern recognition problem* that aims to differentiate and categorize the entities as they are recognized. It is the process of getting to know the records by the training data to identify the data points and determine to which set of categories it belongs. According to machine learning terminology, this problem falls under *supervised learning* where the model is trained on the labelled dataset and then tested on the dataset of same kind.

The dataset usually contains some features that are redundant or irrelevant for the classification process. Thus, they can be filtered out or removed after pre-processing the data without the loss of information required for classification. This process is called *feature selection.* After feature selection, the aim is to reduce the generalization error, training time, over-fitting and the dimensionality of the feature vector. The classification algorithm performs an exhaustive search in the space to find a new feature subset and scores them based on an evaluation measure.

Some examples of the classification algorithms based on learning methods are given below.
(denotes the algorithms that are discussed in this report.)*

Supervised Learning	**Unsupervised Learning**
Support Vector Machines*	*K*-means Clustering
K-Nearest Neighbors	Hidden Markov Models
Neural Networks*	Expectation-Maximization (EM) algorithm
Decision Trees	Principal Component Analysis
Random Forest	Singular Value Decomposition
Regression	Generative Adversarial Networks

Table 2.1 – Different classification algorithms based on learning methods

2.2 Motivation

2.2.1 Problem Description

Many experiments which are done to investigate keystroke dynamics as an authentication method have a low error rate between 1.17% and 5% showing that we can rely on such kinds of techniques to authenticate people.[4][5][6] Some of the previous studies have proven that keystroke dynamic authentication is resistant to some type of attacks like shoulder sniffing but still weak against some attacks in which the attacker has the feedback about the typing characteristics of the legitimate person.[8]

A lot of this research is being done on the CMU's benchmark dataset in which the subjects, people at CMU, typed a fixed password 400 times.[9] Some other works tried to capture the keystroke pattern over a long free-flow text.[10] However if this method is supposed to be used for authenticating users in real world systems, getting a training data from the user of this much size is not feasible.

2.2.2 Research Questions

So, in this project we are trying to investigate the following research questions:

- Does the model that gave the best results on the CMU's benchmark dataset also give the similar accuracy on the custom dataset of the VJTI, Mumbai students?
- What accuracy can we aim to reach by training the model with only 10 samples instead of 400 that of CMU's dataset?
- Is it possible to authenticate a person based on one general template or we need a set of application dependent templates?

2.3 Literature Survey

We review some of the previous literary works about keystroke dynamics for authentication purposes that uses various feature selection and classification algorithms.

2.3.1 Keystroke Dynamics on 'Fixed Text'

Livia C.F.Araujo, Luiz H.R.Sucupira Jr., M. G. L. L. & Yabu-Uti[5] achieved an FMR of 1.9% and an FNMR of 1.45%. In the same paper they also tried to mimic legitimate users by showing invalid users (impostors) how a legitimate user was typing and then try to mimic this user. They achieved an FMR of 3.66% which is worse than the previous result but still acceptable. They state that the keystroke dynamics is strong against mimicking using a shoulder sniffing attack. Furthermore, they show that familiar text passwords give better results than random passwords.

Peacock, A., Ke, X., & Wilkerson, M.[7] compared the results from fifteen scientific papers. Many authors reported FMR and FNMR less than 2.5%. However, many of the good performers require users to write long text before they are authenticated. This is unacceptable in case of static authentication since it would be a very costly solution for a company, if their workers had to spend several minutes a day only to be authenticated. Other suggested solutions in this paper required the authentication system to be updated every time a new user is added, or when user typing behaviour change over time. Those systems that have to be updated are almost unusable for large organizations. Only two of the fifteen reports had more than 50 participants. The majority had less than 25 participants.

In 2009, Kevin S. Killourhy and Roy A. Maxion of Dependable Systems Laboratory, Carnegie Mellon University (CMU) published a paper on keystroke dynamics titled '*Comparing Anomaly-Detection Algorithms for Keystroke Dynamics*' [9]. As per the paper, they collected the dataset from 51 subjects (typists), each typing a password '.tie5Roanl' 400 times. On this *benchmark dataset,* 14 various anomaly-detection algorithms (detectors) were implemented and were compared based on their error rates. The methodology followed was:

a) Run the training phase of the detector on the timing vectors from the first 200 password repetitions typed by the genuine user. The detector builds the model of the user's typing behavior.
b) Then run the test phase of the detector on the timing vectors from the remaining 200 password repetitions typed by the genuine users. The number of users accepted is the '*true accepted/true positive*' count and the number of users rejected is the '*false rejected/false negative*' count.
c) Then run the test phase of the detector on the timing vectors from the first five repetitions typed by each of the 50 imposters. The number of users accepted is the '*false accepted/false positive*' count and the number of users rejected is the '*true rejected/true negative*' count.

To measure the accuracy rates of the detectors, *equal-error-rate (EER) and false positive rate (FPR)* were measured from the *receiver-operating-characteristic (ROC)* curve. Some results are tabulated as follows.

Detector	EER (std. deviation)	FPR (std. deviation)
1. Manhattan (scaled)	**0.096 (0.069)**	0.601 (0.337)
2. Nearest Neighbor (Mahalanobis)	0.100 (0.064)	**0.468 (0.272)**
3. SVM (one-class)	0.102 (0.065)	0.504 (0.316)
4. Mahalanobis	0.110 (0.065)	0.482 (0.273)
5. Neural Network (auto-assoc)	0.161 (0.080)	0.859 (0.220)

Table 2.2 – Average EER and FPR values of detectors (2009)

The paper concluded with the remark that the detectors that have considerably great performance over the benchmark dataset are Nearest Neighbor (Mahalanobis), Manhattan, Mahalanobis and SVM (one-class). The more advanced models such as auto associative neural network performs poorly in comparison to the statistical learning models.

In 2012, Y. Zhong, Y. Deng and A. K. Jain published a paper under IEEE titled *'Keystroke dynamics for user authentication'* [11]. The paper continued the research made by Kevin and Roy in 2009 on the *benchmark dataset* by explaining why Manhattan and Mahalanobis distance classifiers beat traditional machine learning models and neural networks. The paper discussed the issue that Manhattan is more robust to the outliers and hence has a reduced EER as compared to Mahalanobis, whereas Mahalanobis has reduced FPR. To get the best of both worlds, the paper proposed and used a new distance metric which is the combination of Manhattan and Mahalanobis distance metrics. The results were as follows:

Detector	EER (std. deviation)	FPR (std. deviation)
Nearest Neighbor (new distance metric)	**0.087 (0.056)**	**0.423 (0.269)**

Table 2.3 – Average EER and FPR values of detectors (2012)

In 2019, Madhuri Ghorpade from California State University titled *'User Authentication using Keystroke Dynamics'* [12] addressed the problem highlighted by the previous studies that Neural Network anomaly detectors compared to other anomaly detectors like Manhattan or Nearest Neighbor. The paper aimed to increase the accuracy rate of NN based detectors. For this, instead of training the neural network over 33 available features, the author trained the neural network on 21 components obtained from Principal Component Analysis (PCA). Along with FCNN (fully connected neural network), author made Support Vector Machine (SVM) and K Nearest Neighbor (KNN) models. This is by far the best results obtained on CMU's benchmark dataset.

Detector	EER	FPR
1. SVM (one-class)	0.037	0.0256
2. KNN	0.039	0.0254
3. FCNN	**0.034**	**0.0241**

Table 3.3 – Average EER and FPR values of detectors (2019)

2.3.2 Keystroke Dynamics on 'Free-flow Text'

In 2016, Hayreddin Çeker and Shambhu Upadhyaya from University at Buffalo in their report titled *'User Authentication with Keystroke Dynamics in Long-Text Data'* [10] proposed that keystroke dynamics can be used for continuous authentication of users while working at the terminal. Instead of using for one-time password based short-text authentication, the SVM model can be easily extended for the long-text continuous authentication. They collected their own data at Clarkson University Lab. 39 subjects were enrolled to type for half an hour session and for two sessions within a period of 11 months. The features chosen were the flight and dwell times of 12 most common digraphs. They were able to successfully distinguish 34 subjects with an appropriate scale of RBF Kernel in one-class SVM. The EER in their study varied from 2.94% (0.0294) to 0% (0.0000) when 4 and 12 (or more) number of diagraphs are used, respectively.

In 2018, Lu Xiaofeng, Zhang Shengfei and Yi Shengwei published a paper titled *'Continuous Authentication by free-text keystroke based on CNN + RNN'* [13]**.** This paper proposes to divide the user keystroke data into a fixed-length keystroke sequence and convert the keystroke sequence into a keystroke vector sequence according to the time feature of the keystroke. A model of RNN + CNN is used to learn the sequence of individual keystroke features for user authentication. In order to improve the performance of the network model, the keystroke sequence is firstly processed by CNN to extract a higher-level keystroke feature sequence. Then the keystroke feature sequence inputs into the RNN. The model was tested using enhanced dataset of University at Buffalo, involving 157 subjects, participating for 3 sessions with each participant making an average 5,700 keystrokes in each session. The results showed that the value of EER was 3.04% (0.0304).

In 2020, Alejandro Acien, Aythami Morales, Ruben Vera-Rodriguez and Julian Fierrez, from Universidad Autonoma de Madrid, Spain, in their paper titled *'TypeNet: Scaling up Keystroke Biometrics'* [14] addressed the problem of *upto what limit of users, keystroke dynamics is able to distinguish the users.* The chose the Siamese Recurrent Neural Network and made the following conclusions. For 1K users, TypeNet obtains an EER of 4.8% using only 5 enrollment sequences and 1 test sequence per user with 50 keystrokes per sequence.

Using the same amount of data per user, as the number of test users is scaled up to 100K, the performance in comparison to 1K decays relatively by less than 5%, demonstrating the potential of TypeNet to scale well at large scale number of users. Alto University's dataset, which contains 5GB of keystroke dataset obtained from 1,68,000 participants over three months of time was used. The participants were asked to memorize English sentences from a set of sentences and type it. The length of each sentence ranged from 3 words to 70 characters. This model completely outperforms both POHMM (Partially Observable Hidden Markov Models) based models and digraphs and SVM based models which were trained on the same dataset. The results ranged from 9.53% to 3.33% EER for a user-set of size 100K.

2.3.3 Outcome of the Literature Survey

For keystroke dynamics over fixed set, statistical learning methods (e.g., distance classifiers) or neural networks, both are fine. As neural networks tend to capture a lot of patterns (also the noise), it is necessary that only the principal components are fed to them for training purposes.
For keystroke dynamics over free-flow text, recurrent neural networks in conjunction with additional techniques is the best choice.

2.4 Action Plan

In the literature survey, it is observed that most of the work is done on the CMU benchmark dataset. The plan is to reiterate over the most recent work on CMU's dataset done by Madhuri Ghorpade. In this reiteration, we aim to create SVM and FCNN models for the benchmark dataset. So, we first propose to train different models on the benchmark dataset so that the results can be compared with the previous works in this field. It is also observed that, although there are many studies on the use of RNN models for free-flow text, the performance of RNN models on benchmark dataset have not been extensively studied. So, we propose a RNN based binary and multiclass classification models and Siamese RNN models to be trained on the benchmark dataset. The comparison of these RNN models would then be made with other models trained on the same dataset.

After testing different models on the benchmark dataset, we intend to explore the real-life use of our models and the possibility of them being deployed for practical authentication purpose. To achieve this, we propose to create a system to record the keystroke latencies and then deploy so that people can access easily access it. Using this system, we can collect the typing patterns of subjects in a format similar to the benchmark dataset. The collected dataset can then be compared to the benchmark dataset in order to note the similarities and differences.

It is to be noted that the collected dataset shall not be as extensive as the benchmark dataset in terms of samples per user. To overcome this limitation, we propose to use the Siamese RNN architecture for this dataset. Since Siamese RNN network

Chapter 3

DATASET

3.1 Benchmark Dataset Description

The data are arranged as a table of 34 columns. Each row of data corresponds to the timing information for a single repetition of the password by a single subject.

The first column, *subject,* is a unique identifier for each subject (e.g., s002 or s057). Even though the data set contains 51 subjects, the identifiers do not range from s001 to s051; subjects have been assigned unique IDs across a range of keystroke experiments, and not every subject participated in every experiment. For instance, Subject 1 did not perform the password typing task and so s001 does not appear in the data set.

The second column, *sessionIndex*, is the session in which the password was typed (ranging from 1 to 8).

The third column, *rep*, is the repetition of the password within the session (ranging from 1 to 50).

The remaining 31 columns present the timing information for the password. The name of the column encodes the type of timing information. Column names of the form *H.key* designate a hold time for the named key (i.e., the time from when key was pressed to when it was released). Column names of the form *DD.key1.key2* designate a keydown-keydown time for the named digraph (i.e., the time from when key1 was pressed to when key2 was pressed). Column names of the form *UD.key1.key2* designate a keyup-keydown time for the named digraph (i.e., the time from when key1 was released to when key2 was pressed). Note that UD times can be negative, and that H times and UD times add up to DD times.

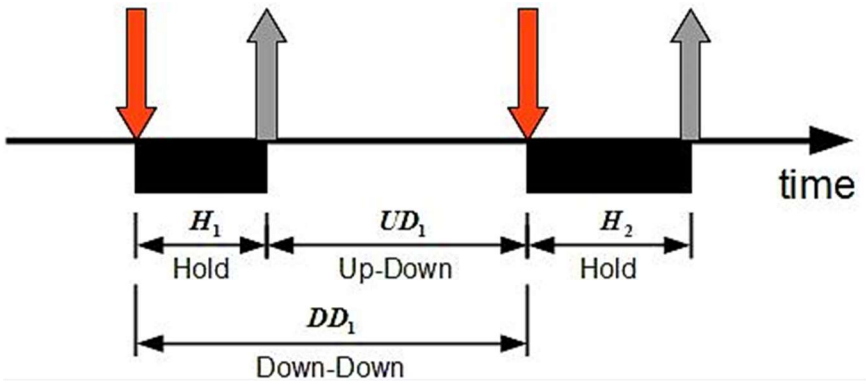
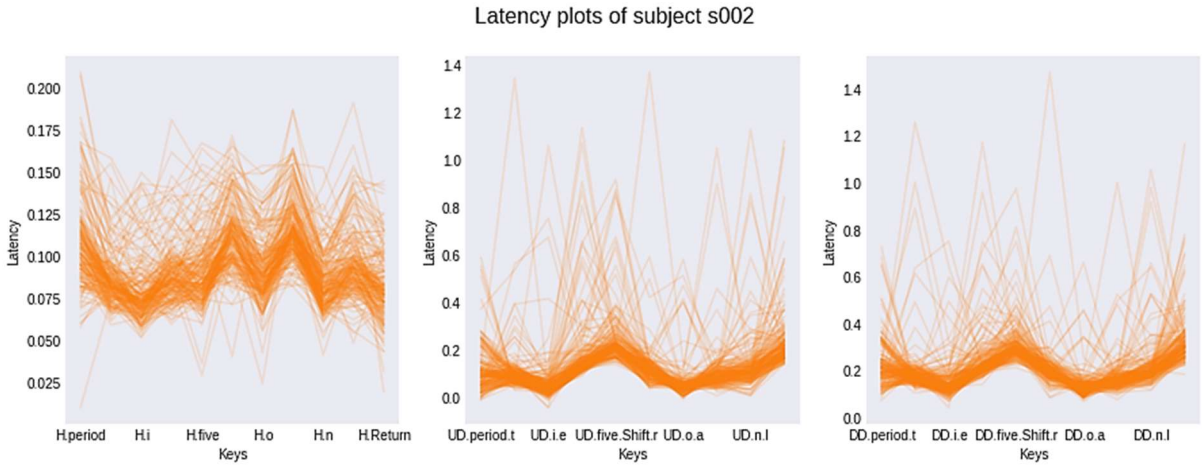
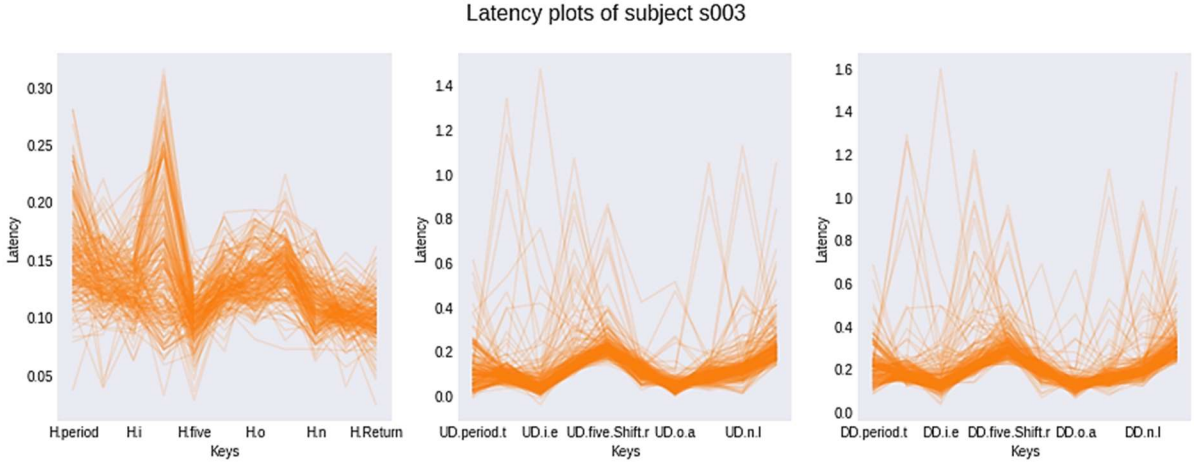


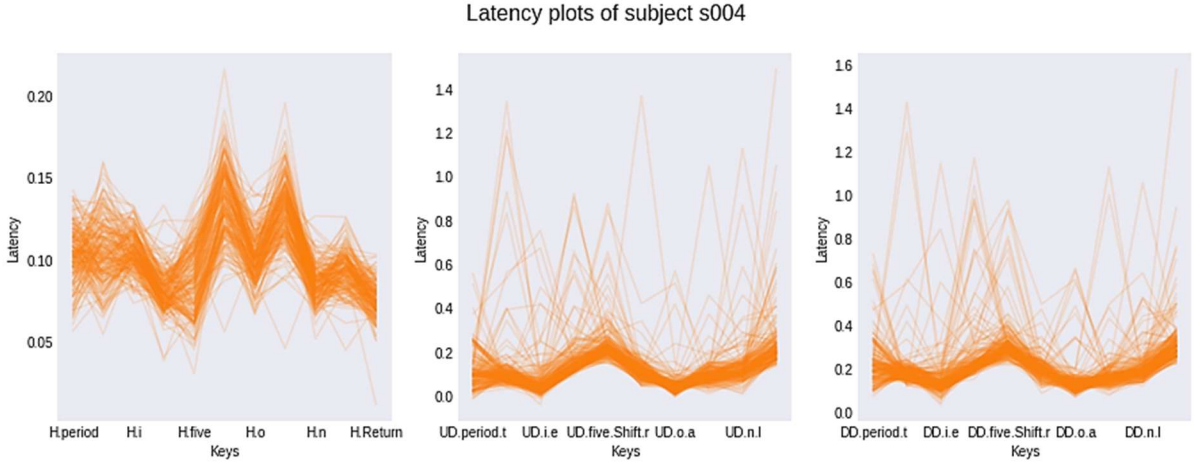
Fig. 3.1 – Keystroke Features

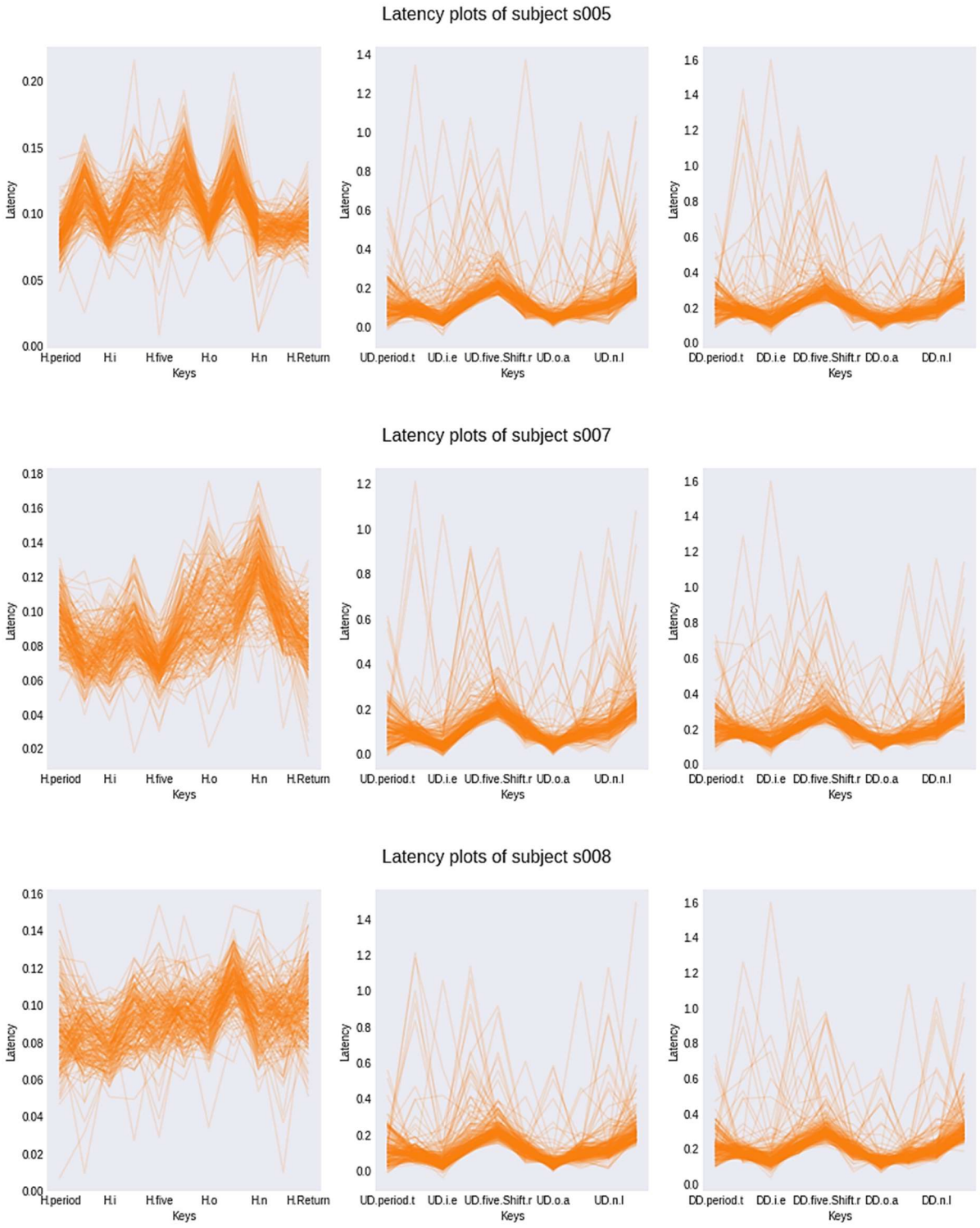
3.2 Visualizing Benchmark Dataset

Following are the latency plots (H-latency plot, UD-latency plot and DD-latency plot) for starting six subjects. In these plots, out of the 400 timing patterns of the password repetitions, 200 are chosen at random and plotted.









From above plots, we see that:

a) The *pressure pattern* (hold duration of the key) changes a lot from user to user.

b) The *DD-latency pattern* and *UD-latency pattern* more or less remains the same.

This can be more clearly seen in the following plots. Here, we have considered the mean durations of the timing patterns for the starting 6 subjects.

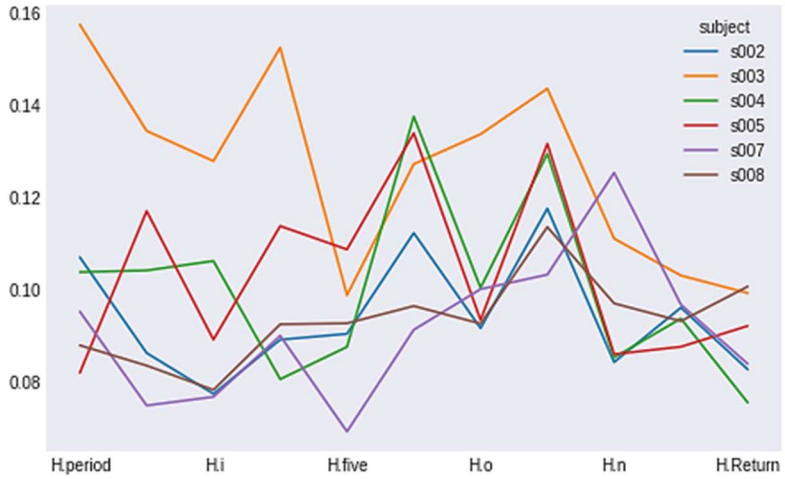


Fig. 3.2 – Plot of average H values of starting 6 subjects

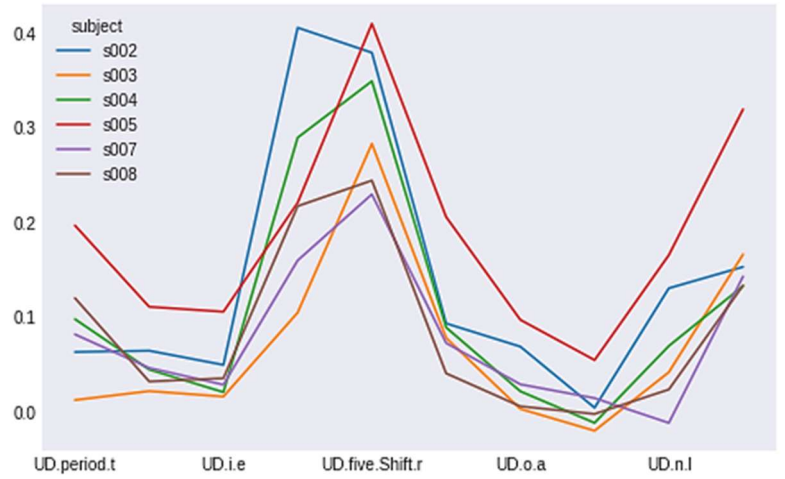


Fig. 3.3 – Plot of average UD values of starting 6 subjects

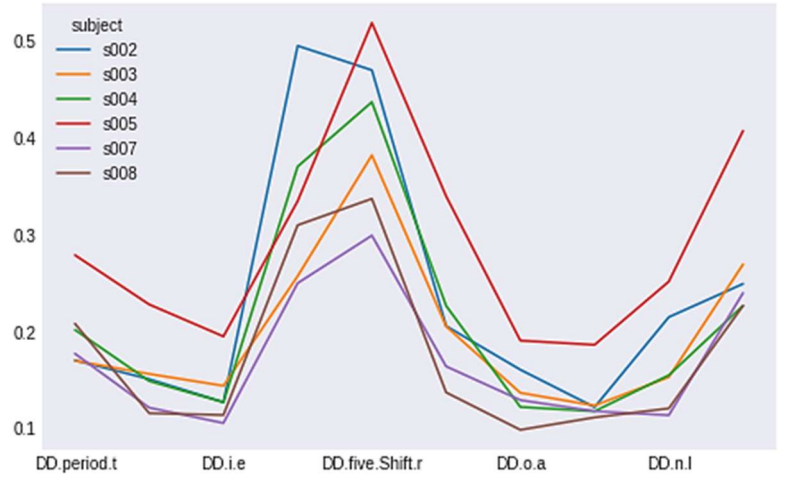


Fig. 3.4 – Plot of average DD values of starting 6 subjects

3.3 Collecting Custom Dataset

A website was intended to effectively collect the keystrokes of users. Users were expected to type a fixed sentence "5afe&5ecure" and their email id which will be of variable characters and size based on individual. Each User will type both of these keyword 12 times and considering any current entry (out of 12) will stand not valid in following:

- Character Mismatch.
- Delay in typing consecutive characters is more than 2.5 secs.
- The backspace key is pressed.

In all the above cases the user will have to re-type the keyword again. This ensures to maintain the integrity of the data. Below is the snapshot of the website designed,

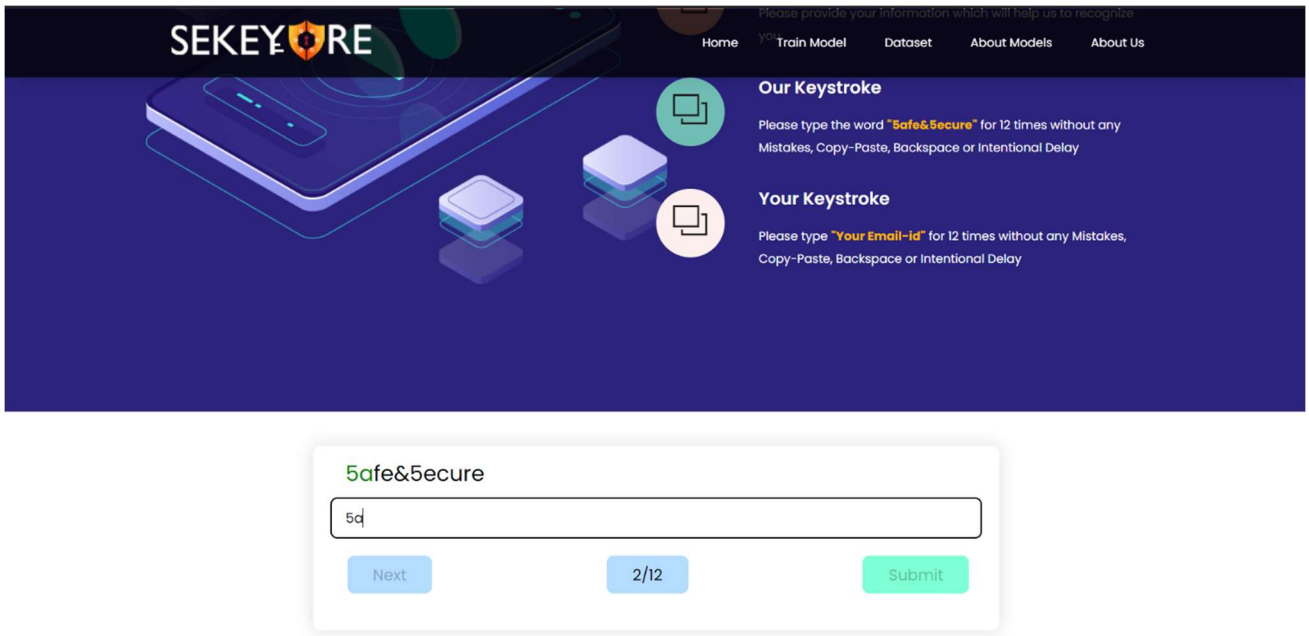


Fig. 3.5 – Website Demo

The technology stack used for the website: HTML, CSS, Bootstrap, Javascript, Firebase.
The website was hosted on firebase and the data collected from users were stored in real-time in Firebase's Cloud Firestore, a No-SQL database that allows to effectively store, sync, and retrieve data.

keyDown and keyUp timestamps are recorded for each key pressed, which is used to determine the latencies between consecutive keys. Latencies considered are key Down-Down time, Up-Down time, Up-Up time and Hold time. One thing to note here is that this custom dataset is considering one extra parameter than the CMU's Benchmark dataset, which is Up-Up time (i.e., the time from when key1 was released to when key2 was released).

Fig. 3.6. shows the timestamps recorded for key "5".

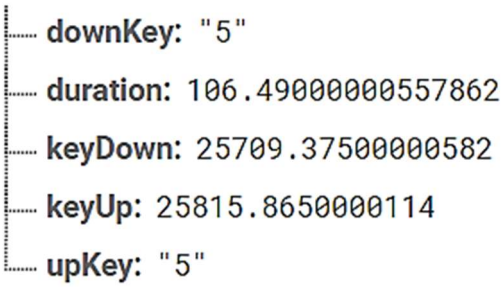


Fig. 3.6 – Data collected for key "5"

2_3_DD stored in Fig. 3.6, measures the Down-Down time between the second and third key which is well illustrated in Fig. 3.1.

The format of data stored in Firebase's real-time is JSON and was converted in excel for data exploration and building predictive models.

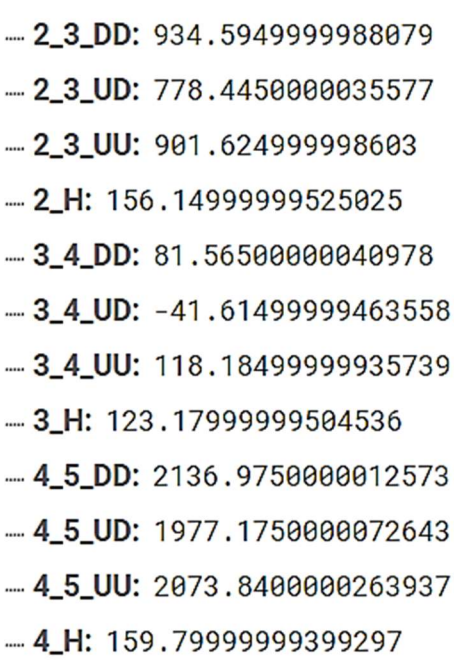


Fig. 3.7 – Latencies

The final dataset (Sekeyure Dataset) for the fixed keyword "5afe&5ecure" is organized as a table of 42 columns. Each row of data corresponds to the timing information for a single repetition of the password by a single user. The first column, the user conventionally referred to as a *subject,* is a unique identifier for each subject ranging from sub1 to sub50.

The rest of the 41 columns represents the timing information for the given keyword. The column with the title *key. H* designates a hold time for the named key (i.e., the time from when the key was pressed to when it was released). Also, *key1_key2_DD* (i.e., the time from when key1 was pressed to when key2 was pressed), *key1_key2_UD* (i.e., the time from when key1 was released to when key2 was pressed), and *key1_key2_UU* (i.e., the time from when key1 was released to when key2 was released) represents the key Down-Down time, Up-Down time and Up-Up time respectively.

Following chart shows the final collected dataset of 50 users.

#	subject	1_2_DD	1_2_UD	1_2_UU
0 – 599	50 unique values	0.09 – 2.39	0 – 2.28	0.13 – 2.36
0	sub1	1.7658850000007	1.53061999997589	1.75384999997913
1	sub1	0.996145000099204	0.902580000110901	1.0570300000254
2	sub1	1.44953000009991	1.3514900000300198	1.52746499993372
3	sub1	0.950539999990724	0.8646949999965721	1.04836000001523
4	sub1	0.570590000017546	0.481469999998807	0.681729999952949
5	sub1	0.434465000056661	0.345510000013746	0.528330000001005
6	sub1	0.38065499998629004	0.29651000001467703	0.46664500003680504
7	sub1	0.71393500000704	0.615544999949634	0.7928299999330191
8	sub1	0.631144999992102	0.543955000001005	0.709774999995715

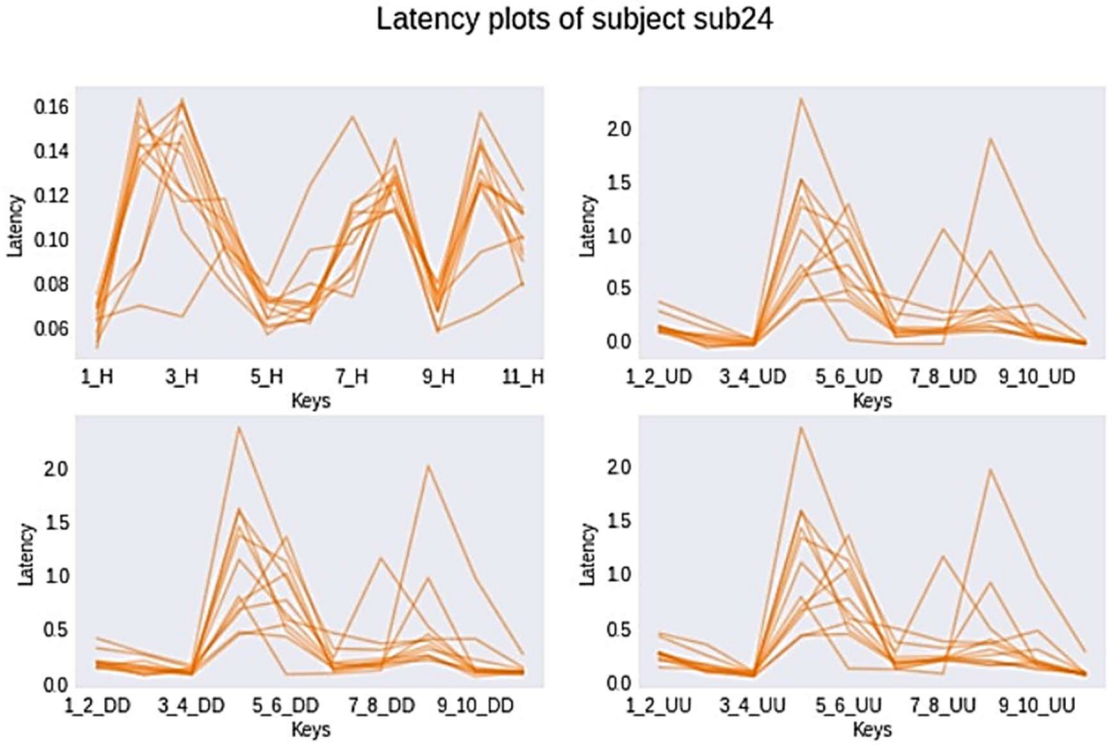
Fig. 3.8 – SEKEYURE – Custom Dataset

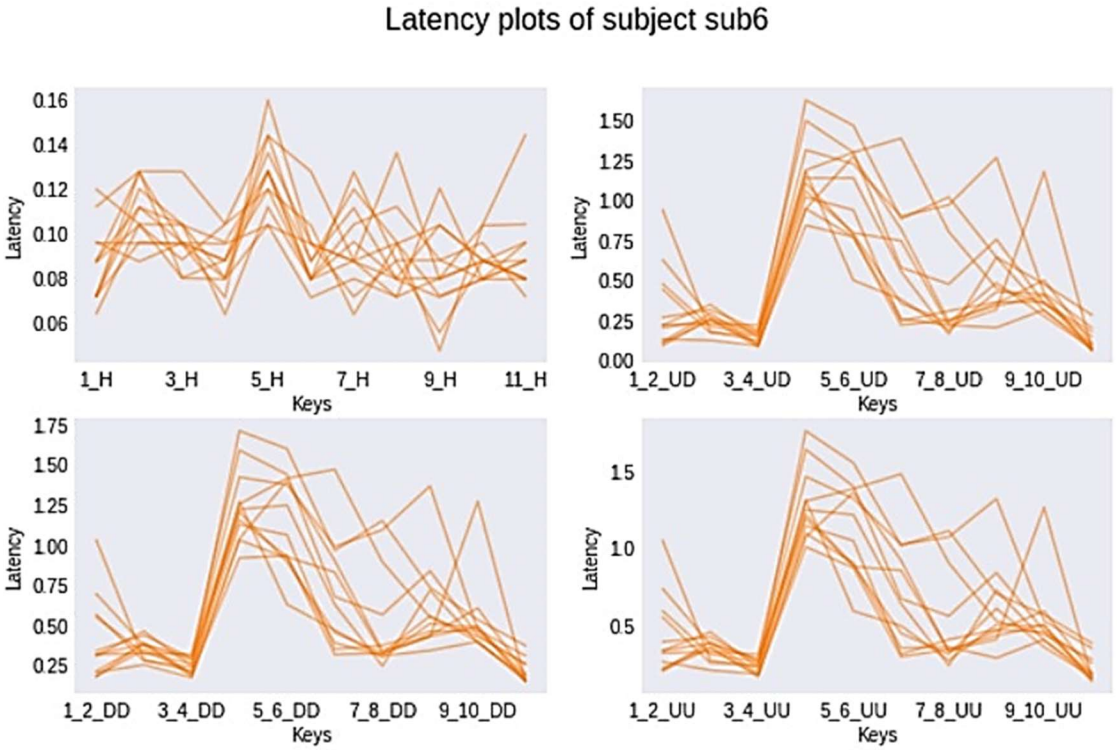
Why the word "5afe&5ecure" was chosen?

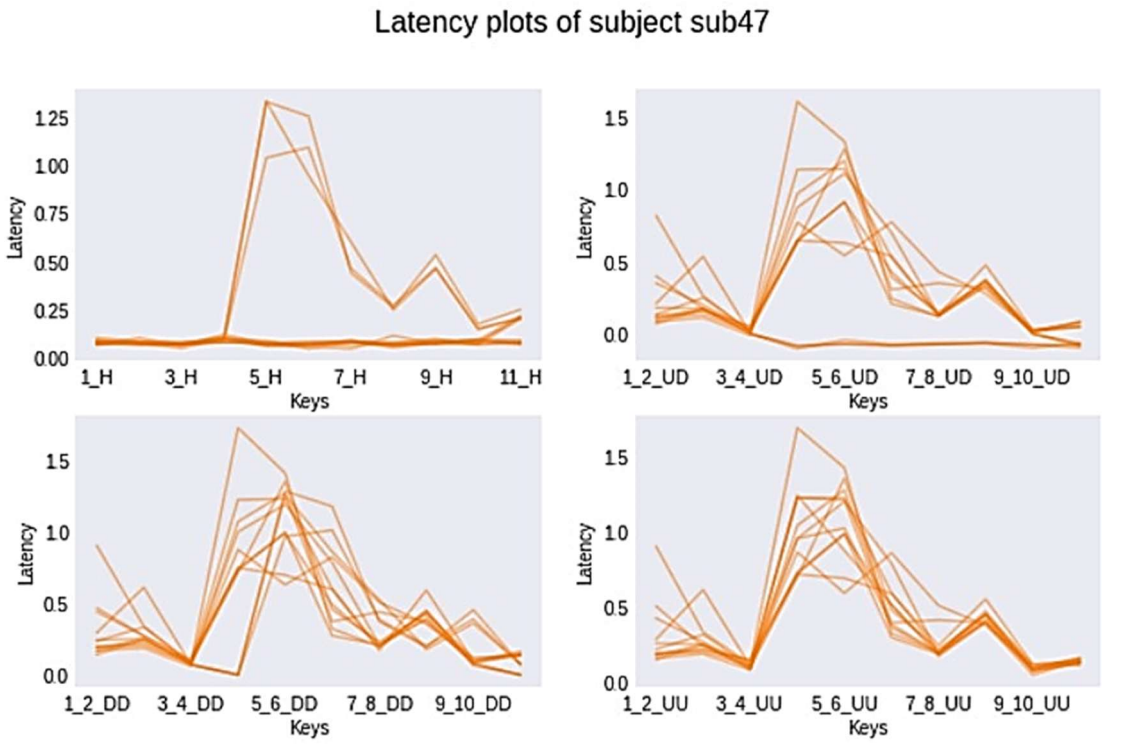
The word was chosen keeping in mind the passwords that are generally a combination of alphanumeric and special characters. The chosen keyword took inspiration from the benchmark dataset's keyword. Also, the word Safe and Secure aligned with our objective of making a secure and robust system for authentication.

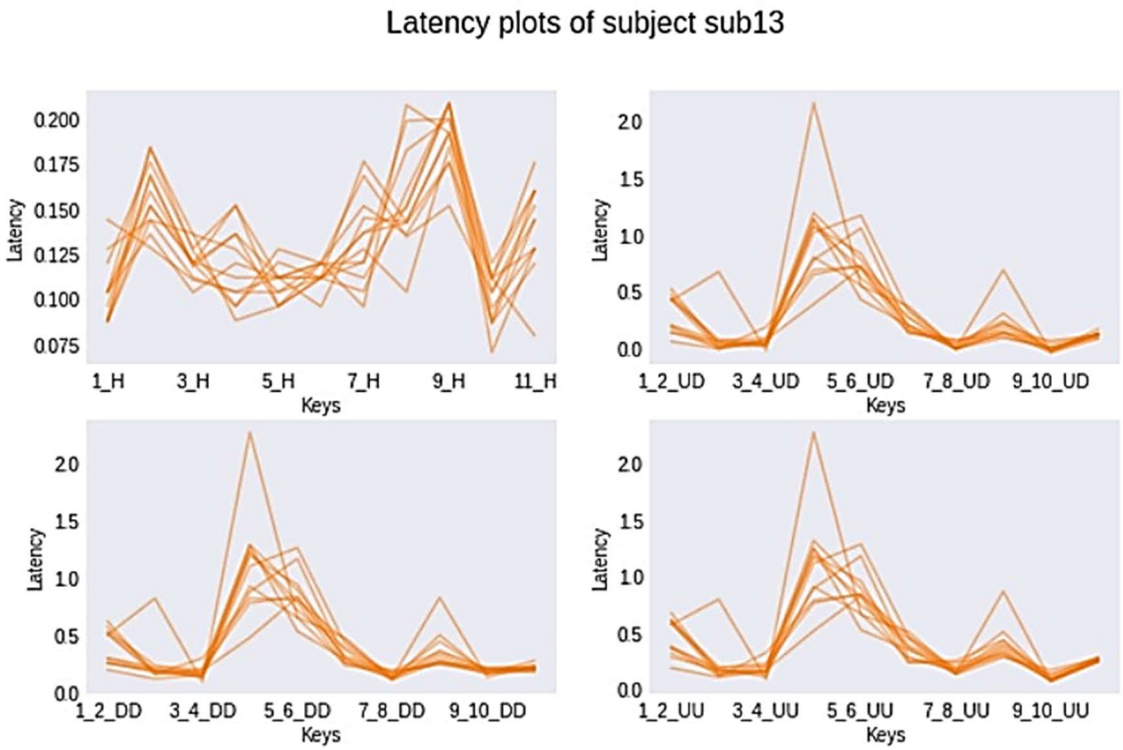
3.4 Visualizing Custom Dataset

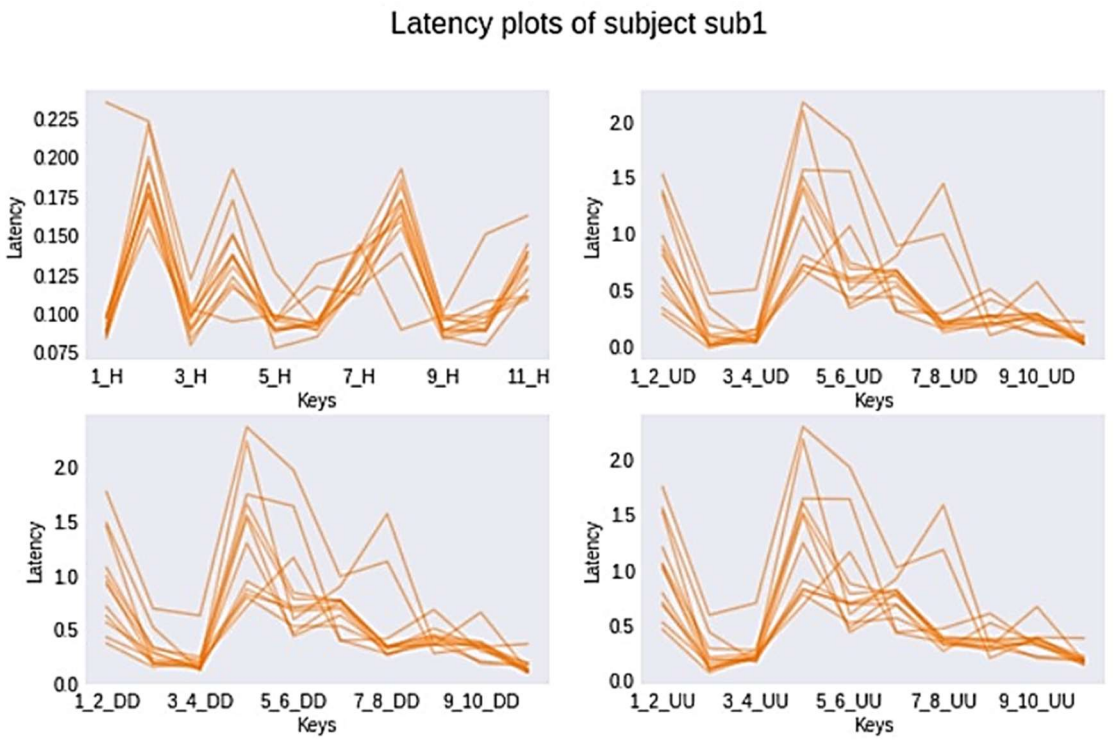
Following are the latency plots (H-latency plot, UD-latency plot, DD-latency plot and UU-latency plot) for random six subjects. In these plots, 16 typing patterns are plotted.

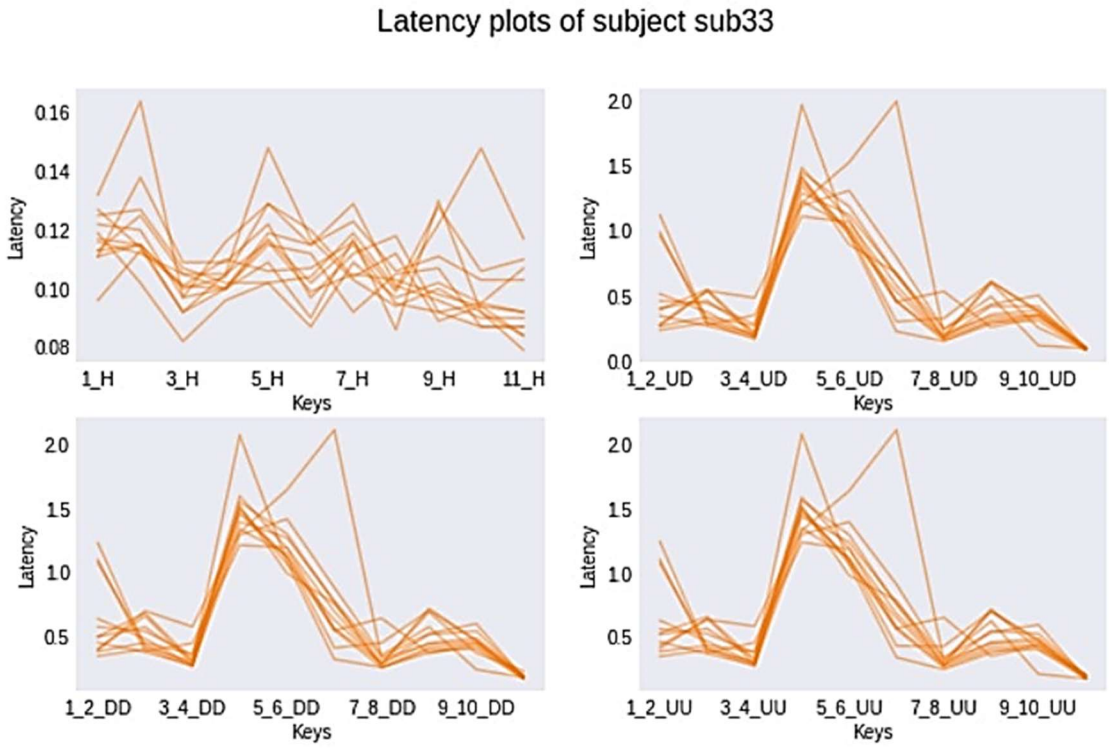





Latency plots of subject sub47
Latency
Keys
1_H
3_H
5_H
7_H
9_H
11_H
1_2_UD
3_4_UD
5_6_UD
7_8_UD
9_10_UD
1_2_DD
3_4_DD
5_6_DD
7_8_DD
9_10_DD
1_2_UU
3_4_UU
5_6_UU
7_8_UU
9_10_UU


Latency plots of subject sub13
Latency
Keys
1_H
3_H
5_H
7_H
9_H
11_H
1_2_UD
3_4_UD
5_6_UD
7_8_UD
9_10_UD
1_2_DD
3_4_DD
5_6_DD
7_8_DD
9_10_DD
1_2_UU
3_4_UU
5_6_UU
7_8_UU
9_10_UU





From above plots, we get two insights:

1) The H-latency plot doesn't have much discriminating information.
2) The UU-latency parameter which was missing in the CMU dataset, could also be a parameter that could help in the classification process.

This can be more clearly seen in the following plots. Here, we have considered the mean durations of the timing patterns for the starting 6 subjects of our custom dataset.

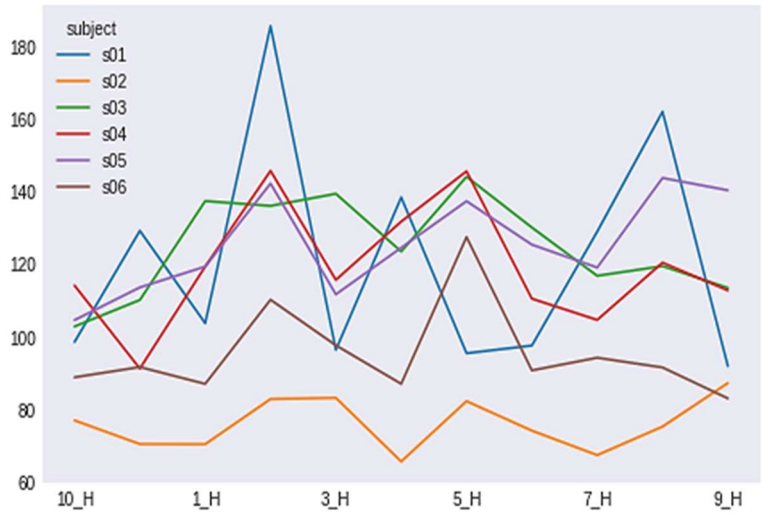


Fig. 3.5 – Plot of average H values of starting 6 subjects

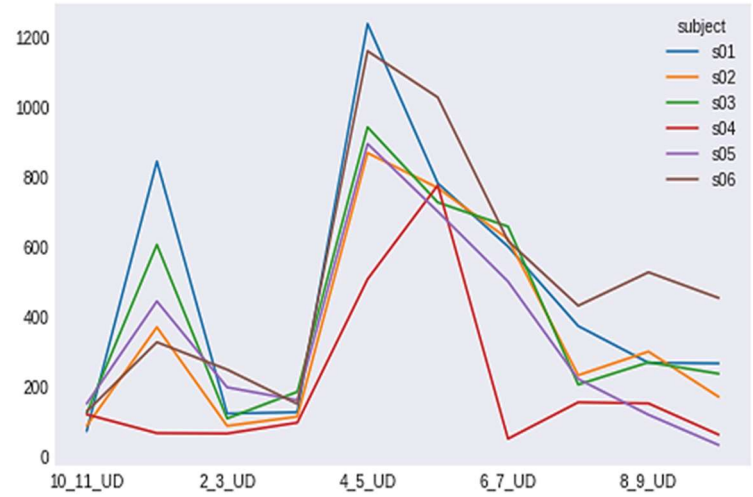


Fig. 3.6 – Plot of average UD values of starting 6 subjects

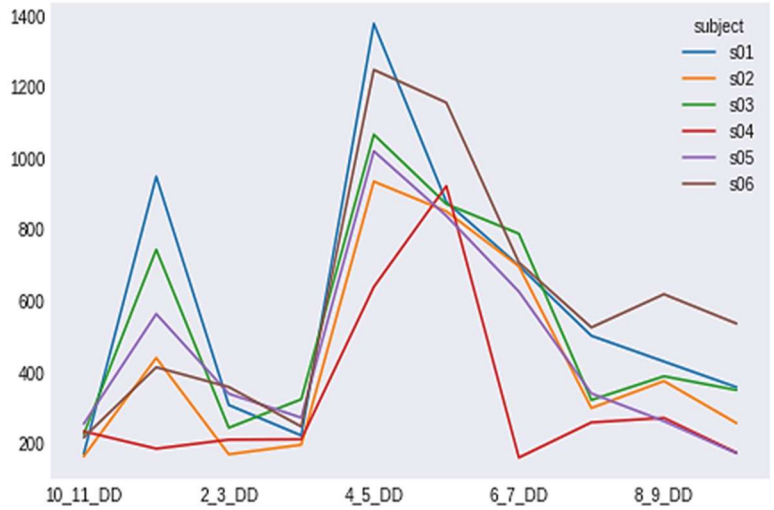


Fig. 3.7 – Plot of average DD values of starting 6 subjects

Chapter 4

MODEL TRAINING

4.1 Model Planning

As mentioned in the action plan, following is the list of models to be trained and studied depending upon the dataset.

Dataset	Models to be trained
CMU's benchmark dataset	SVM, FCNN, Binary LSTM, Multiclass LTSM, Siamese RNN.
Custom dataset	Siamese RNN

Table 4.1 – Dataset and the corresponding models to be trained

4.1.1 Activation Functions:

Activation Functions define the non-linear transformation of the weighted sum of a set of inputs which aids in finding the non-linear relation between inputs and outputs.
Following are the set of activation functions used during the course of this project:

Sigmoid:
Since binary predictive models have classification output as either 0 or 1 and sigmoid functions transform the input to a range of values between 0 and 1 which can be then threshold depending on the requirement. The sigmoid function is also differentiable and monotonic.

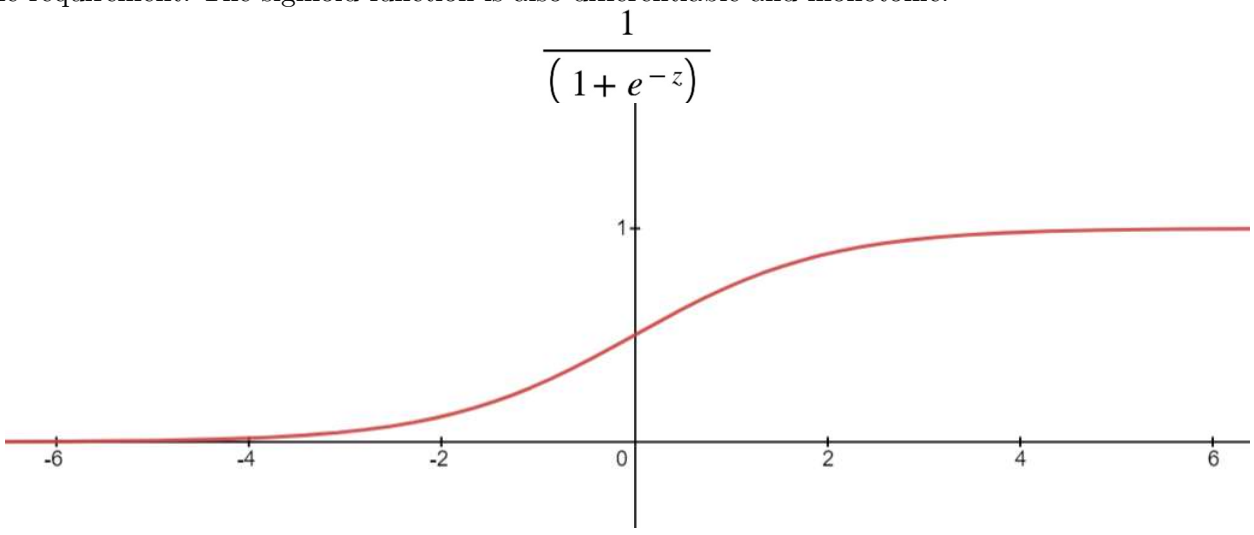


Fig. 4.1 – Sigmoid activation function

tanH:

This activation function is analogous to the sigmoid activation function and the only difference being the symmetry through the origin. The gradient of the tanH function happens to be steeper than the sigmoid function and its symmetry across origin makes it more preferable to use.

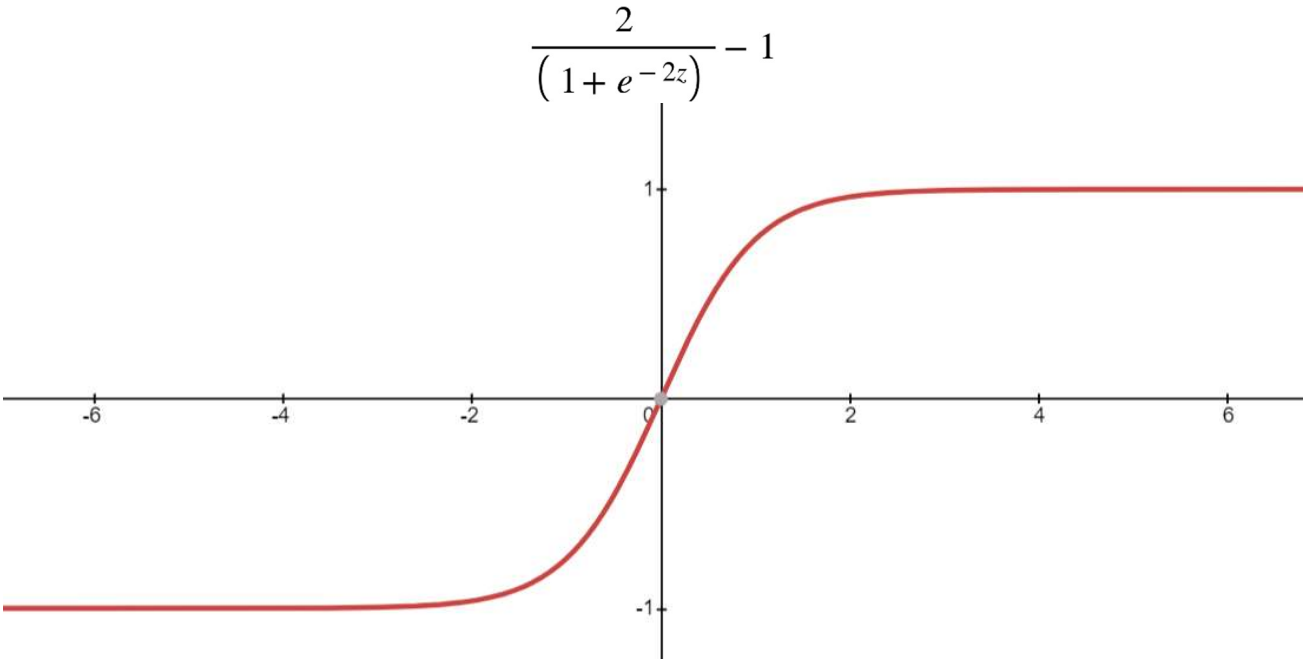


Fig. 4.2 – tanH activation function

ReLU:

Rectified Linear Unit(ReLU) maps weighted sum of a set of inputs to output directly if it is greater than or equal to zero, else it forwards zero. This makes some neurons get deactivated if their linear transformation sum is less than zero, which makes ReLU more computationally efficient than sigmoid or tanH.

$$f(x) = x, \; for \; x \geq 0$$
$$= 0, \; for \; x < 0$$

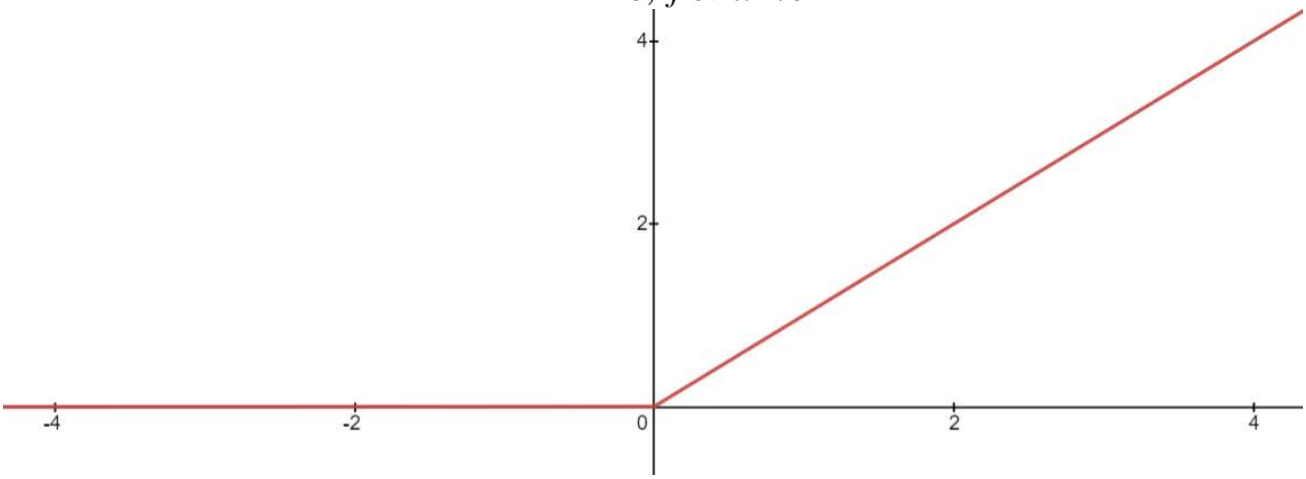


Fig. 4.3 – ReLU activation function

Softmax:
Multiclass classification will have many neurons in the final output layer, softmax activation function which is the combination of the sigmoid function returns the probabilities of a given set of inputs belonging to a particular class.

$$f(x)_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}}, for\ j = 1, 2, 3 \ldots . K$$

4.1.2 Loss Functions:

The choice of loss function determines the way in which the model learns to perform predictions over any given dataset. Hence, it is vital to choose an optimal loss function depending on the need in order to get optimal results. Here, the loss functions that we used are binary cross entropy, categorical cross entropy and contrastive loss.

Binary Cross Entropy:
It is also called as log loss. In binary classification, the model outputs a value between 0 and 1. If this value varies greatly from the expected value (which is 0 or 1), then the loss value will be high and the model will be highly penalised. If the difference between the predicted and expected values is low then the penalization will be less. In this way, the binary classification models were trained using binary cross entropy loss function. The loss value is calculated as follows

$$J(\theta) = -(y \log(p) + (1-y) \log(1-p))$$

where y: *expected outcome*

p: *predicted outcome*

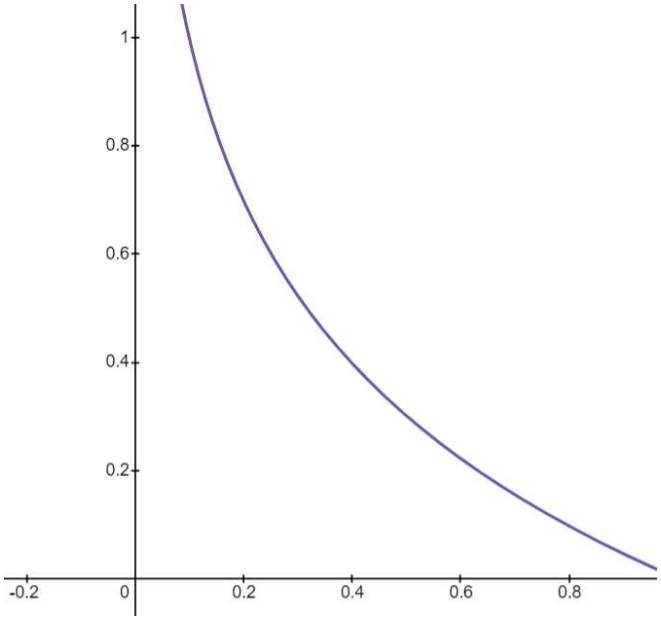


Fig. 4.4 – Binary Cross Entropy Loss function

Categorical Cross Entropy:

In multiclass classification, the output of the model will be a vector. The number of elements in the vector is equal to the number of classes. Each element in the output vector represents the probability of the input sequence belonging to that particular class (one hot vector). In our case, the expected vector will have a single 1, indicating that the input keystroke sequence belongs to that corresponding class and all other elements will be zero.

After obtaining the expected vector and predicted vector, the following summation is computed to calculate the loss value.

$$J(\theta) = -\sum_{i=1}^{number\ of\ classes} y_i \log(y_i)$$

Contrastive loss:

This loss function is used to learn the similarities between two keystroke sequences. It was used for Siamese RNN models. Using this loss function, the model is trained in such a way that similar keystroke sequences output a similar vector . Hence, The distance between the output vectors of similar keystroke sequences will be less i.e. they will be close to each other. Similarly, the distance between dissimilar keystroke sequences will be more, i.e. they will be far from each other. The contrastive loss is computed as follows

$$J(\theta) = (1-y)\,D^2 + y \cdot max(margin - D, 0)^2$$

$$where,\ y:\ actual\ outcome$$

$$D:\ Euclidean's\ Distance$$

4.1.3 Early Stopping:

A major task in training a Deep Learning based model is to decide the number of epochs required to efficiently train the model.

If the training is stopped before the optimal number of epochs, we are losing the capabilities of neural networks, which eventually results in the under-fitting of the model. If the model is trained for an exceptionally longer duration, then it'll result in the occurrence of over-fitting, due to which the model will perform poorly on the real-life external data sets.

One way to achieve optimal training of the model is to use the concept of early stopping. It employs monitoring of increment in validation loss after every epoch. Validation data is separated from training data and can be approximated to be a generalized dataset that resembles real-life data. It is observed that validation loss keeps on decreasing after every epoch as the model tries to fit over the data. As soon as the model starts overfitting on the training data, validation loss experiences a jump which can be treated as an indicator of the start of the overfitting of the model.

All the mentioned models used the early stopping with the **patience of 15**. This essentially means that if the validation loss doesn't decrease after 15 consecutive epochs the training stops and the previous epoch which resulted in the lowest validation loss is treated as the final epoch.

Models were trained using with and without early stopping and it turned out that early stopped models were giving **significant improvement in accuracy**.

4.1.4 Adam Optimizer:

Adam optimizer is an adaptive learning rate optimization algorithm that's been designed specifically for training deep neural networks. Adam is the combination of RMSprop and Stochastic Gradient Descent with momentum. It uses the squared gradients to scale the learning rate like RMSprop and it takes advantage of momentum by using the moving average of the gradient instead of the gradient itself like SGD with momentum.

4.2 Architecture of Models implemented over Benchmark Dataset

4.2.1 Support Vector Machine (SVM)

The preprocessing of the data for One-Class SVM involved following tasks:

For each subject, subsequent steps in sequential order:

- Select 'n' records from the dataset (in our case n = 400) of genuine user, genuine data.
- Select columns ranging from *H.period to H.return* from the genuine data.
- Training data is generated by selecting 300 samples randomly from genuine data. No negative samples are considered while training since One-Class SVM builds boundary across the positive data.
- Test data is generated by selecting remaining 100 samples of genuine data as test data for the genuine user and also select 5 samples of each subject, in total 250 samples test data of imposter.

For SVM, scikit-learn library is used. The kernels used are linear, RBG, poly and sigmoid. Out of these, the kernel for which the EER will be smallest would be chosen.

4.2.2 Fully Convolutional Neural Network (FCNN)

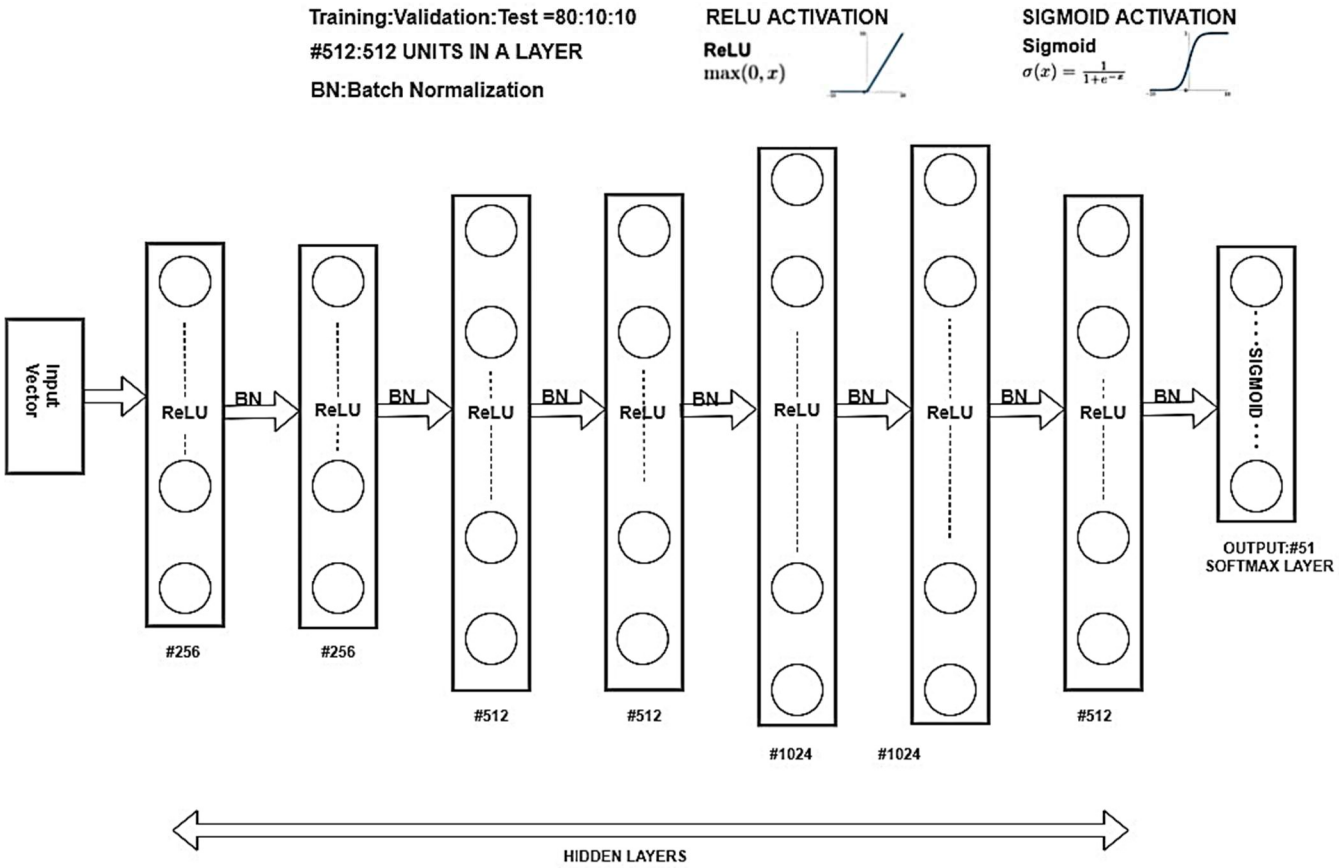


Fig. 4.5 – FCNN Architecture for Benchmark Dataset

Before training the neural network, the input data is divided into batches where each batch has a fixed number of training examples. The entire training dataset is randomly shuffled first. This is to prevent training examples of similar nature being present in a batch and thus giving false learning pattern to the algorithm or to prevent heavily biased data in the training set. Then the input dataset's features were normalized. The train-test split used is 75:25**.** In this algorithm batch size of 32 was used. The training set is now randomly assigned to batches using a pseudorandom algorithm. Examples in a particular batch are normalized by first subtracting from the batch mean and then dividing by the standard deviation of the batch by a process known as batch normalization. Batch normalization speeds up the training, smoothens the loss function. The normalized input data is now passed through two Dense layers consisting of 256 units each, followed by 3 more dense layers of 512 units each, followed by two dense layers of 1024 units, followed by a 512 unit dense layer and finally a 51 unit layer. For each layer other than the last, ReLU (Rectified Linear Activation Unit) was used as an activation function and batch normalization was applied to the output. Sigmoid activation function was used for the last layer. The loss function used in this algorithm was categorical cross entropy. The optimizer used in this model is Adam optimizer. The probability values for each user is indicated by the output (51) units in the case of benchmark dataset.

4.2.3 Binary Long Short-Term Memory RNN (Binary LSTM)

This requires the preprocessing of the data. For each subject, perform the subsequent steps in sequential order:

1. Select 'n' records from the dataset (in this case n = 400) of a genuine user, genuine data.
2. Select columns ranging from *H.period* to *H.return* from the genuine data.
3. Training data is generated by selecting 300 samples randomly from genuine data. No negative samples are considered while training since One-Class SVM builds boundary across the positive data.
4. Test data is generated by selecting the remaining 100 samples of genuine data as test data for the genuine user and also selecting 5 samples of each subject, in total 250 samples test data of imposter.

A best-fitting model to enable optimal performance was configured after a series of Hyperparameter tuning. The model which turned to be the finest had four layers of LSTM cells with each layer having 272 units. To avoid complex co-adaptations on training data and to reduce overfitting, a layer of dropout for regularization was added after every subsequent layer. The dropout layer for this model was Gaussian Dropout with a 20% dropout rate. Finally, a Dense layer with 272 neurons was added. Activation functions used were tanh and sigmoid, with Binary Cross-Entropy loss function and Adam optimizer.

Early stopping helped to prevent getting stuck at local minima.

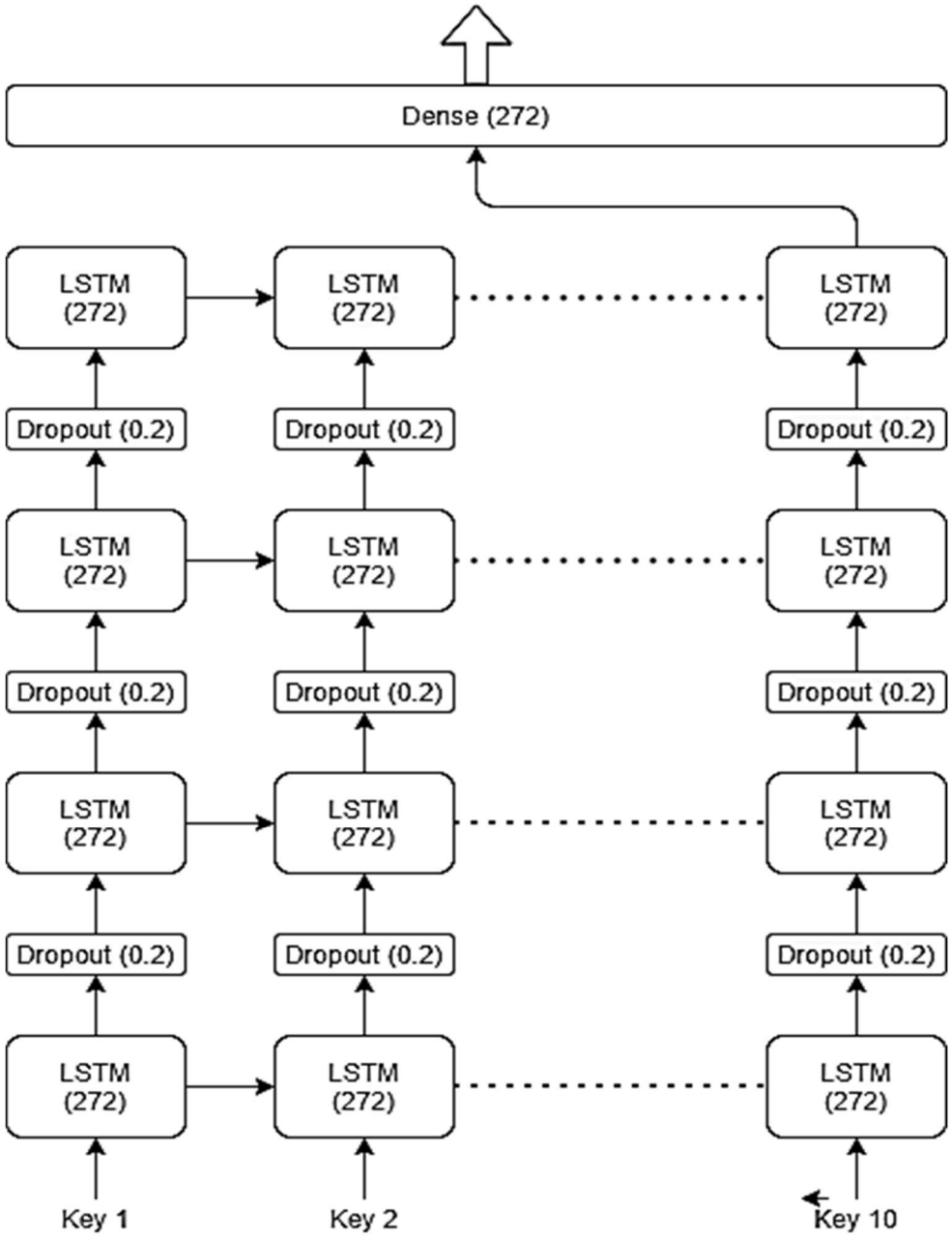


Fig. 4.6 – Binary LSTM Architecture for Benchmark Dataset

4.2.4 Multiclass Long Short-Term Memory RNN (Multiclass LSTM)

This requires the preprocessing of the data. For each subject, perform the subsequent steps in sequential order:

1. Since the dataset consists of 51 odd users conventionally referred to as subjects, One-Hot Encoding which happens to be more expressive for such a categorical dataset was performed.
2. Initially, the entire dataset was shuffled randomly. This helps in preventing the examples of identical nature being present in a batch and thus restraining false learning patterns to the algorithm. This also restricts heavily biased data in the training set.

3. Standard Scaling was performed to have the same range of values across the dataset to avoid imbalance and make computation more feasible.
4. Training and Testing split was 70% to 30% respectively.

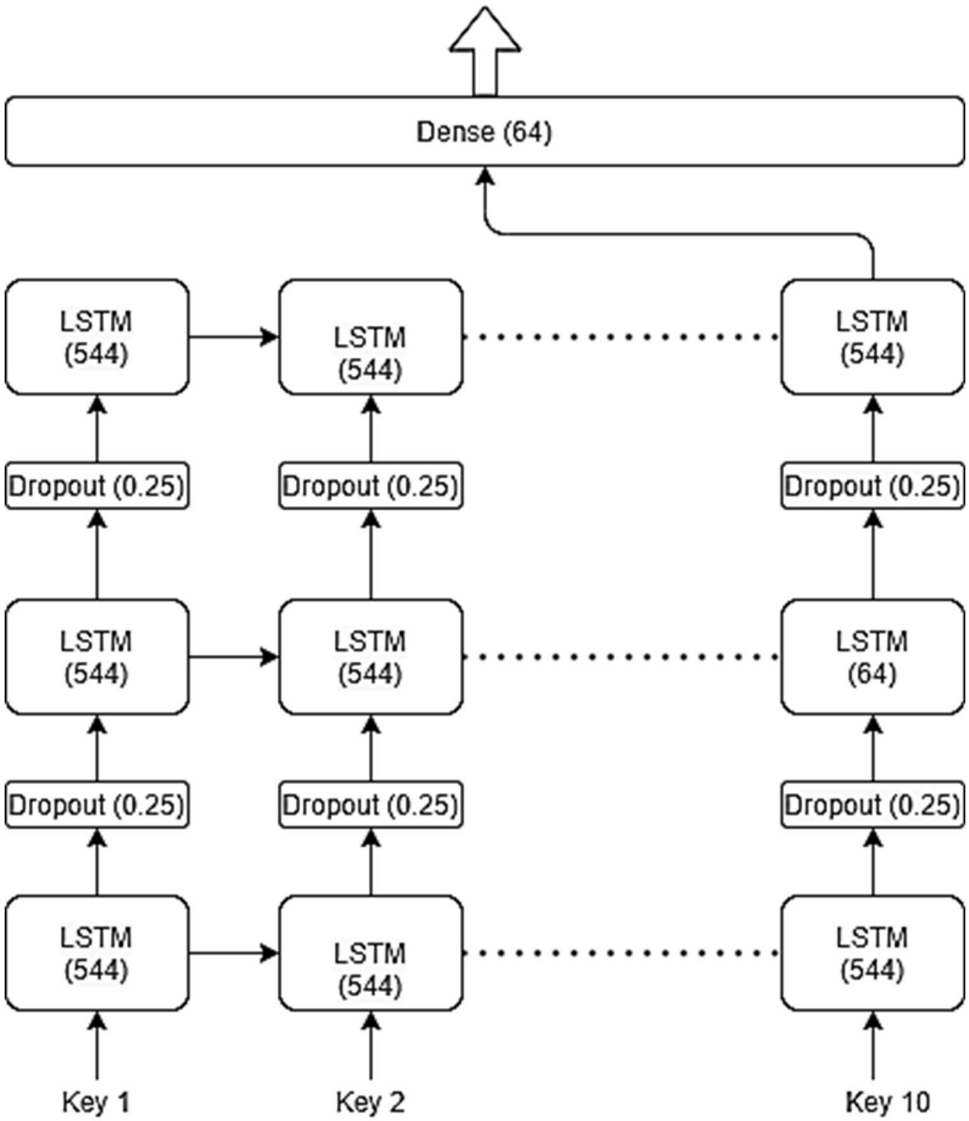


Fig. 4.7 – Multiclass LSTM Architecture for Benchmark Dataset

A set of model parameters that delivered the highest predictive accuracy was extracted through Hyperparameter tuning. The Multiclass RNN model had 3 layers of LSTM cells, having 544 units each. A dropout rate of 25% using Gaussian Dropout was introduced after each layer. Softmax and tanh were used as activation functions and since the output for multiclass classification was represented using one-hot encoding, the loss function used was Categorical Cross Entropy along with Adam Optimizer.

Early stopping helped to prevent getting stuck at local minima.

4.2.5 Siamese LSTM RNN

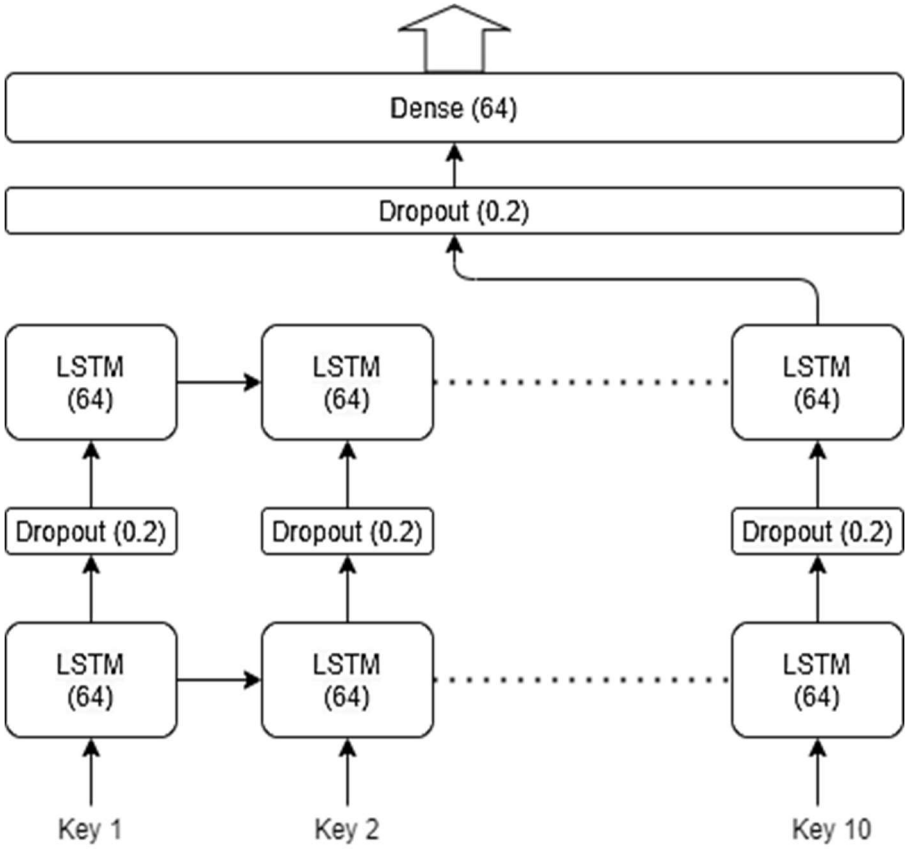


Fig. 4.8 – Siamese LSTM RNN Architecture for Benchmark Dataset

The above architecture was used for training the Siamese LSTM model. There are two LSTM layers followed by a Dense layer with each layer having 64 nodes. A Gaussian dropout layer is used between every two layers. The 'tanh' activation function is used for each layer. The loss function used here is contrastive loss.

For training the model, two keystroke sequences are passed through it to obtain two 64 parameter feature vectors, one for each input sequence. Next, we compare these to two vectors to find if they both belong to the same subject or not. Ideally, if both the samples belong to the same subject the distance between their feature vectors should be zero.
Similarly, the distance between them should be one if they belong to different subjects. So, the difference between two output vectors will be a float value between zero and one. To compute this distance, we use the cosine distance function which returns a value between 0 and 1. In this way, the Siamese RNN network is trained and subsequently evaluated.

We can train the Siamese RNN in two methodologies.

Method 1: Test Train Split: 40 train subjects & 11 test subjects

In this method, we split the available subjects. We used 40 subjects for training and the remaining 11 subjects for testing and using these subjects we created the train and test dataset respectively.

To create the training dataset, by including the positive and negative samples of each subject. For each subject, we separate that subject's data from the training dataset and perform a cross product of that data with itself. In this way, we generate the positive data for that subject and give it a label 0, since both the keystroke sequences in the dataset belong to the same person. To generate the negative data for each subject, we take the cross product of that subject's data with the data of other subjects (subjects other than the current subject). This data is labelled 1, since both the keystroke sequences in any row belong to different subjects. We then sample an equal number of positive and negative data. Here we sampled 10000 samples of positive and negative data for each user. In this way positive and negative samples are created for each user, they are merged and jumbled.

The same method is used to generate a test dataset using the test users.

Method 2: Test Train Split: 300 train samples & 100 test samples for each subject

In this method, we split the available samples of subjects. We split the data in 72:25 ratio, used 300 samples per subject for training and the remaining 100 subjects for testing and using these subjects, we created the train and test datasets respectively.

To create the training dataset, we create a training dataset by taking 300 samples of each subject. For each subject in the training dataset, we separate that subject's data and perform a cross product of that data with itself. In this way, we generate the positive data for that subject and give it a label 0 since both the keystroke sequences in the dataset belong to the same person. To generate the negative data for each subject, we take the cross product of that subject's data with the data of other subjects (subjects other than the current subject) in the training dataset. This data is labelled 1, since both the keystroke sequences in any row belong to different subjects. We then sample an equal number of positive and negative data. Here we sampled 10000 samples of positive and negative data for each user. In this way positive and negative samples are created for each user, they are merged and jumbled.

The same method is used to generate a test dataset using the testing dataset of 100 samples of each user.

4.3 Architecture of Models implemented over custom Dataset

4.3.1 Siamese LSTM RNN

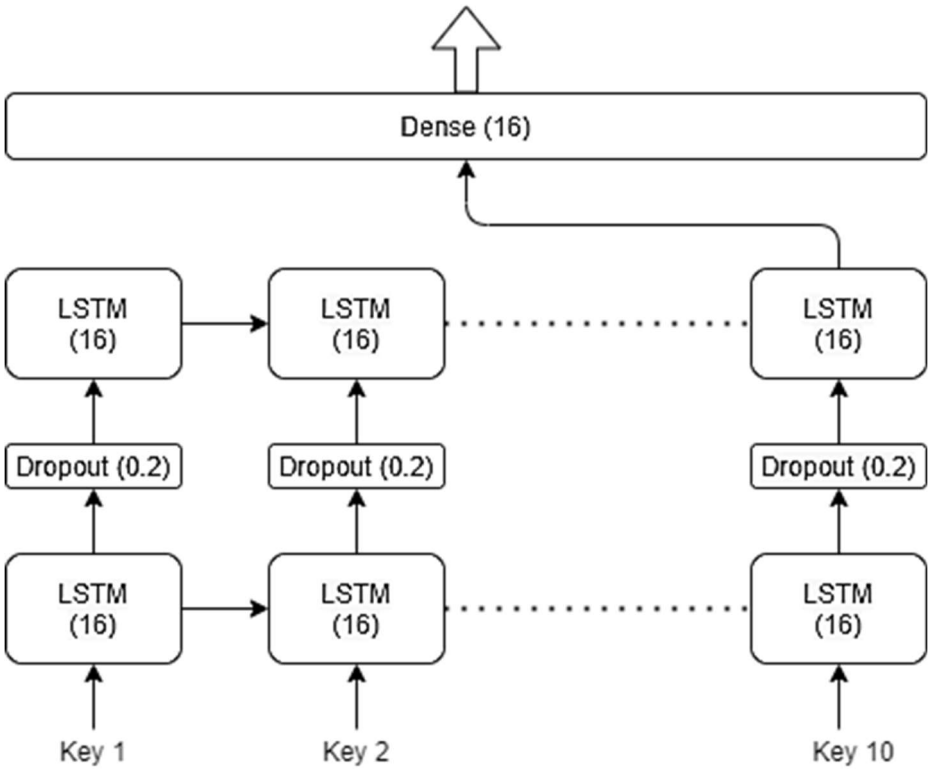


Fig. 4.9 – Siamese LSTM RNN for custom dataset

The above architecture was used for training the Siamese LSTM model. There are two LSTM layers followed by a Dense layer with each layer having 16 nodes. A Gaussian dropout layer is used between every two layers. The 'tanh' activation function is used for each layer. The loss function used here is contrastive loss.

For training the model, two keystroke sequences are passed through it to obtain two 16 parameter feature vectors, one for each input sequence. Next, we compare these to two vectors to find if they both belong to the same subject or not. Ideally, if both the samples belong to the same subject the distance between their feature vectors should be zero. Similarly, the distance between them should be one if they belong to different subjects. So, the difference between two output vectors will be a float value between zero and one. To compute this distance, we use the cosine distance function which returns a value between 0 and 1.

In this way, the Siamese RNN network is trained the custom dataset and subsequently evaluated.

Like Siamese LSTM RNN for benchmark dataset, in this case as well, the model could be trained in two ways.

Method 1: Test Train Split (40 train subjects & 10 test subjects)

In this method, we split the available subjects. We used 40 subjects for training and remaining 10 subjects for testing and using these subjects we created the train and test dataset respectively.

To create the train dataset, by including the positive and negative samples of each subject. For each subject, we separate that subject's data from the training dataset and perform cross product of that data with itself. In this way we generate the positive data for that subject and give it a label 0, since both the keystroke sequences in the dataset belong to the same person. To generate the negative data for each subject, we take the cross product of that subject's data with the data of other subjects (subjects other than the current subject). This data is labelled 1, since both the keystroke sequences in any row belong to different subjects. We then sample an equal number of positive and negative data. Here we sampled 144 samples of positive and negative data for each user. In this way positive and negative samples are created for each user, they are merged and jumbled.

The same method is used to generate test dataset using the test users.

Method 2: Test Train Split (9 samples of each subject for train and 3 for test)

In this method, we split the available samples of subjects. We split the data in 72:25 ratio, thereby using 9 samples per subject for training and remaining 3 samples per subjects for testing.

To create the train dataset, we create a training dataset by taking 9 samples of each subject. For each subject in the training dataset, we separate that subject's data and perform cross product of that data with itself. In this way we generate the positive data for that subject and give it a label 0, since both the keystroke sequences in the dataset belong to the same subject. To generate the negative data for each subject, we take the cross product of that subject's data with the data of other subjects (subjects other than the current subject) in the training dataset. This data is labelled 1, since both the keystroke sequences in any row belong to different subjects. We then sample an equal number of positive and negative data. Here we sampled all 10000 samples of positive and negative data for each user. In this way positive and negative samples are created for each user, they are merged and jumbled.

The same method is used to generate test dataset using the testing dataset of 3 samples of each user.

Dimensionality reduction using Hold Latencies:

Although there are more variations in hold latencies of users as seen in the latency plots which when compared with the other latencies Up-Up, Up-Down, and Down-Down doesn't considerably change with different users and follow a similar pattern. The keystroke dynamics problem can be approached using only hold latencies of the user, which will make the training and testing more optimized and make the model feasible to be deployed on the cloud.

The recent study on keystroke dynamics used Principal Component Analysis for dimensionality reduction showed that the most useful feature or in our case latencies had some of the Up-Down and Down-Down latencies and hence discarding these latencies would result in degradation of the model's performance.

Chapter 5

RESULTS AND ANALYSIS

5.1 Output Format

Presentation of output is very important in comparative studies. We output the results in following metrics:

- **ROC Curve**

  Receiver Operating Characteristics (ROC) curve is plotted with True Positive Rate (TPR) against the False Positive Rate (FPR) where TPR is on y-axis and FPR is on the x-axis. Area under this curve represents degree or measure of separability. It tells how much model is capable of distinguishing between classes.

- **Equal Error Rate (EER)**

  Out of the two errors, False Positive Rate (FPR) and False Negative Rate (FNR), the one can be reduced at the expense of the other, i.e., if we reduce one error, the other one increases. Hence, an appropriate middle point is usually used as a threshold based on the relative cost of the errors. At this appropriate point, the probability of occurrence of both the errors is equal and hence minimum. This point is called Equal Error Rate (EER).

5.2 Outputs of Models trained on benchmark dataset

5.2.1 One-class SVM

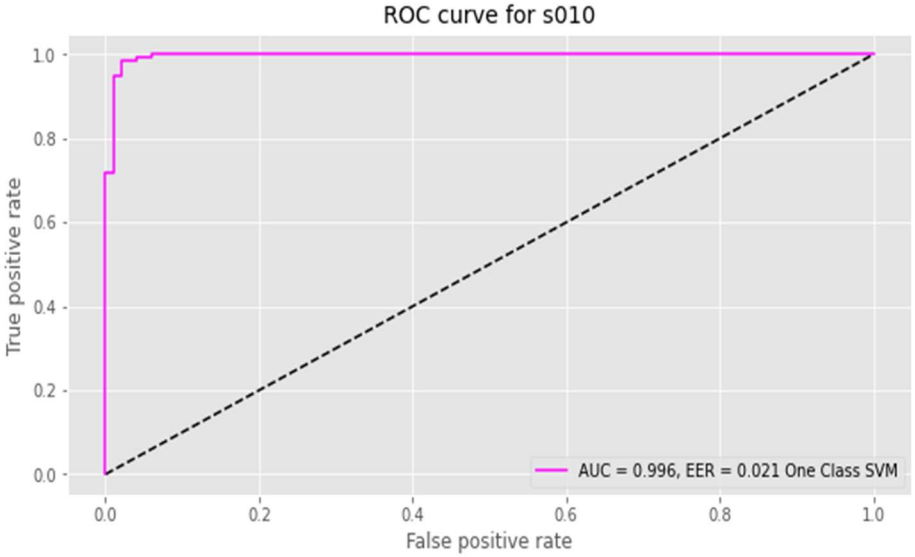


Fig. 5.1 – ROC curve for One-Class SVM model of subject S010

Subject Number	S10	S12	S22	S42	S56	S55
EER value	0.021	0.061	0.040	0.070	0.060	0.020

Table. 5.1 – EER values for One-Class SVM model

5.2.2 Multiclass FCNN

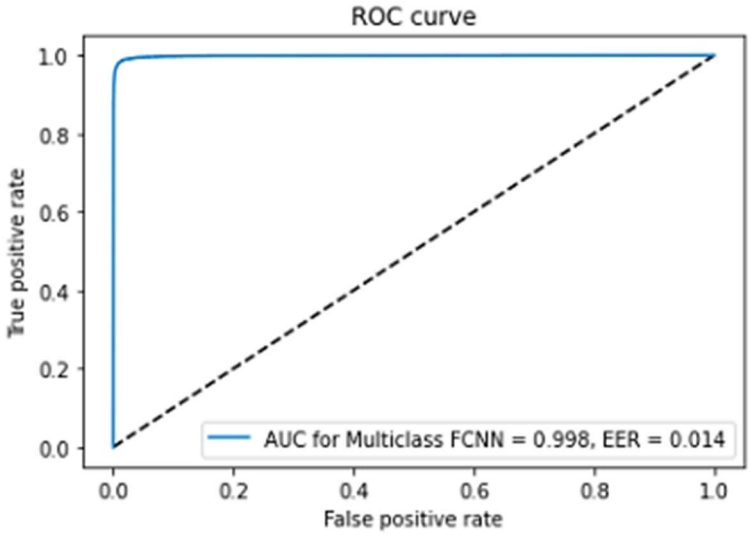


Fig. 5.2 – ROC Curve for Multiclass FCNN

EER	AUC	Test Set Accuracy
0.014	0.998	93.76%

Table 5.2 – EER value for multiclass FCNN

5.2.3 Binary LSTM RNN

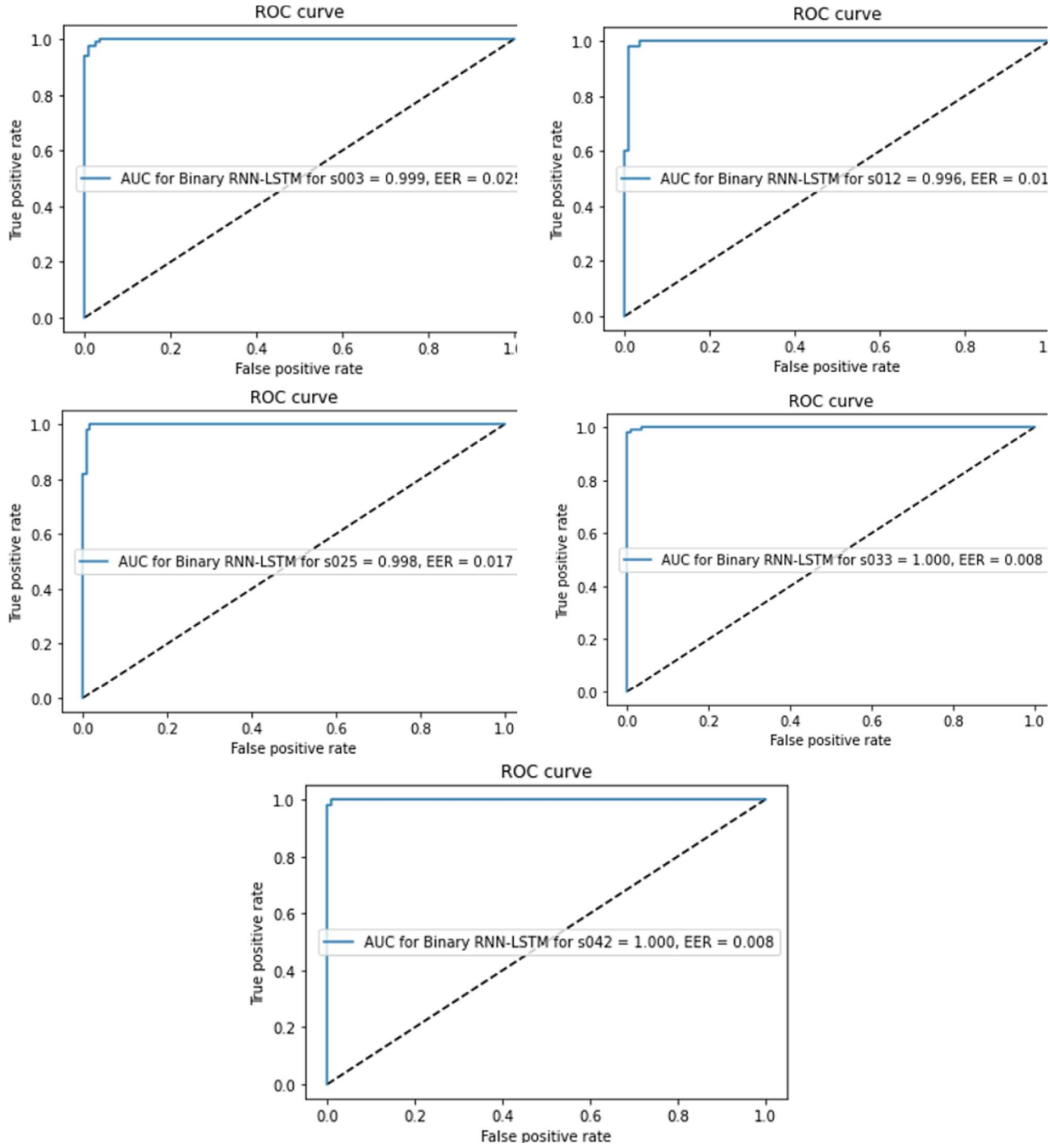


Fig. 5.3 – ROC curves for 5 subjects of Binary LSTM RNN

Subject	EER	AUC	Accuracy
s003	0.025	0.995	97.89%
s012	0.017	0.996	97.92%
s025	0.017	0.998	98.75%
s033	0.008	1.000	98.95%
s042	0.008	1.000	99.58%

Table 5.3 – EER values of 5 subjects under Binary LSTM RNN

5.2.4 Siamese LSTM RNN

Method 1: Test-Train Split: 40 train subjects & 11 test subjects

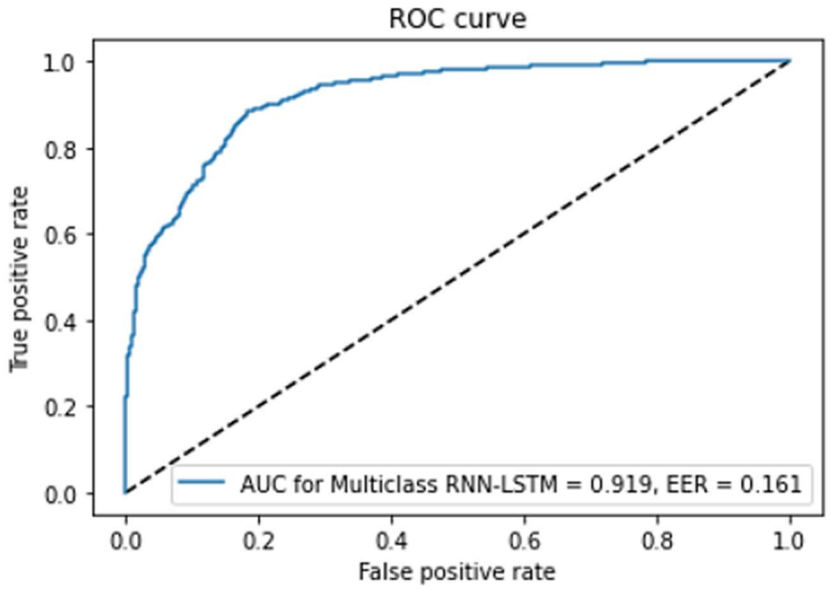


Fig. 5.4 – ROC curve for Multiclass Siamese LSTM RNN (method 1)

EER	AUC	Test Set Accuracy
0.161	0.919	83.59%

Table 5.4 – EER value for Multiclass Siamese LSTM RNN (method 1)

Method 2: Test-Train Split: 300 train samples & 100 test samples for each subject

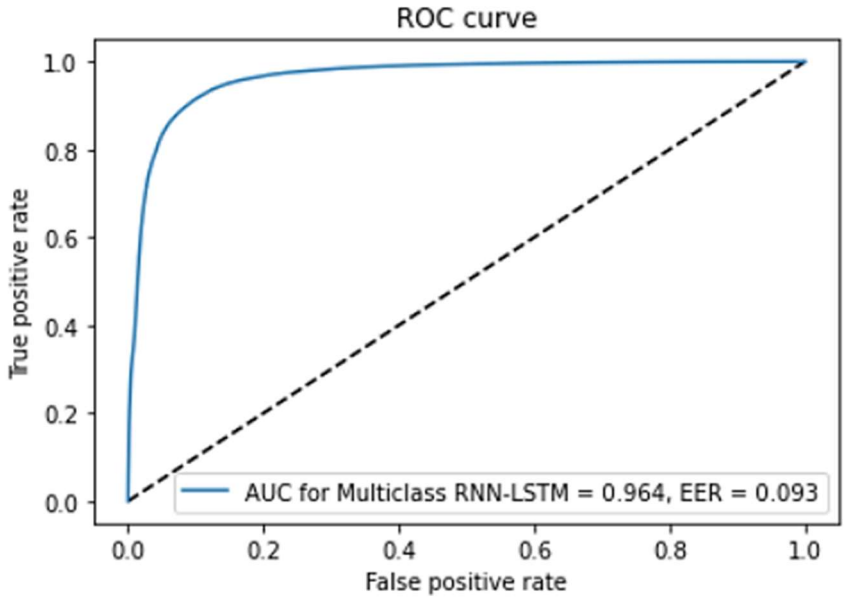


Fig. 5.5 – ROC curve for Multiclass Siamese LSTM RNN (method 2)

EER	AUC	Test Set Accuracy
0.093	0.964	90.71%

Table 5.5 – EER value for Multiclass Siamese LSTM RNN (method 2)

5.3 Outputs of Models trained on custom dataset

5.3.1 Siamese LSTM RNN

Method 1: Test Train Split (40 train subjects & 10 test subjects)

Trail	1	2	3	4	5
Accuracy	81.53%	85.28%	85.80%	83.47%	84.44%

Table 5.6 – Accuracy values of Multiclass Siamese LSTM RNN on trails over custom dataset (method 1)

Method 2: Test Train Split (9 samples of each subject for train and 3 for test)

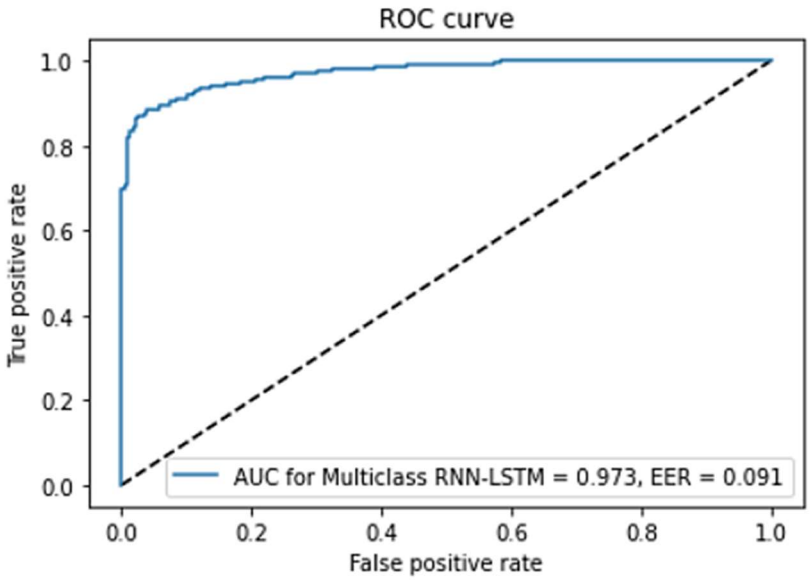


Fig. 5.6 – ROC curve for Multiclass Siamese LSTM RNN over custom dataset (method 2)

EER	AUC	Test Set Accuracy
0.091	0.973	92.22%

Table 5.7 – EER value for Multiclass Siamese LSTM RNN over custom dataset (method 2)

5.4 Comparison Table

Models\Dataset	Benchmark Dataset	Custom Dataset
One-class SVM	0.025	-
FCNN	0.014 (93.76 %)	-
Binary LSTM RNN	**0.010**	-
Siamese LSTM RNN	0.093 (90.71%)	**0.091 (92.22%)**

Table 5.8 – EER (Accuracy) Table of various models

Chapter 6

FUTURE SCOPE

The website which was intended to collect a custom dataset can be accompanied by the models which have been developed in this project. The Deep Learning model has to be deployed using any cloud service. User will have to type the same keyword which was used for data collection, now this data can be fed to the deployed trained model to predict the result.

Since keystroke dynamics deals with the timestamp of each key pressed and released. RNN-LSTM model acknowledges the complexity of the sequence dependence which is not addressed in regressive predictive modeling. Hence more focus can be on the RNN-LSTM model as there is always a possibility of achieving a better classifier by simply fine-tuning the hyper-parameters to fit the needs accordingly.

Using Siamese models better results with limited samples in the dataset which was otherwise not possible with other classification models. Hence increasing the number of users, i.e. expanding the custom dataset could help us achieve better results with the Siamese model.

Chapter 7

CONCLUSION

It is evident that for Binary Classification, the model Binary RNN-LSTM with four layers of LSTM cells outperformed all the existing models with the lowest EER and highest accuracy. For Multiclass classification, the FCNN models offered better results than all the existing models on multiclass classification.

Even though the custom dataset had only 12 samples for each user which in comparison with other datasets is too low, Siamese models perform with around 92% accuracy when trained on all users. An added advantage with Siamese models is that they can be used to test the users whose samples are not present in the training set, which was demonstrated by training on 40 users and testing on the remaining 10 users.

In this way, the use of different LSTM models for keystroke dynamics is found to be yield good results and offer other advantages as well.

References

[1] Sadikan, S. F. N., Ramli, A. A., & Fudzee, M. F. M. (2019). *A survey paper on keystroke dynamics authentication for current applications. ADVANCES IN ELECTRICAL AND ELECTRONIC ENGINEERING: FROM THEORY TO APPLICATIONS (SERIES 2): Proceedings of the International Conference of Electrical and Electronic Engineering (ICon3E 2019).*

[2] Jain, A., Ross, A., & Prabhakar, S. January 2004. An introduction to biometric recognition. IEEE Transactions on Circuits and Systems for Video Technology, 14(1), 4–20.

[3] Hocquet, S., Ramel, J.-Y., & Cardot, H. 2005. Fusion of methods for keystroke dynamic authentication. autoid, 0, 224–229.

[4] Gunetti, D. & Picardi, C. 2005. Keystroke analysis of free text. ACM Trans. Inf. Syst. Secur., 8(3), 312–347.

[5] Livia C.F.Araujo, Luiz H.R.Sucupira Jr., M. G. L. L. & Yabu-Uti, J. B. 2005. User authentication through typing biometrics features. *IEEE Transactions on Signal Processing,* 53(2), 851–855.

[6] Bleha, S. & Obaidat, M. 1991. Dimensionality reduction and feature extraction applications in identifying computer users. IEEE Transactions on systems, Man and cybernetics, 21(2), 452–456.

[7] Peacock, A., Ke, X., & Wilkerson, M. September-October 2005. Typing patterns: a key to user identification. IEEE Security and Privacy Magazine, 2(5), 40–47.

[8] Rundhaug, F. E. N. Can attackers learn someone's typing characteristics. Master's thesis, Gjovik University College, 2006.

[9] Killourhy, Kevin S. and R. Maxion. "Comparing anomaly-detection algorithms for keystroke dynamics." *2009 IEEE/IFIP International Conference on Dependable Systems & Networks* (2009): 125-134.

[10] H. Çeker and S. Upadhyaya, "User authentication with keystroke dynamics in long-text data," 2016 IEEE 8th International Conference on Biometrics Theory, Applications and Systems (BTAS), 2016, pp. 1-6, doi: 10.1109/BTAS.2016.7791182.

[11] Y. Zhong, Y. Deng and A. K. Jain, "Keystroke dynamics for user authentication", 2012 *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops, Providence, RI, 2012, pp. 117-123, doi: 10.1109/CVPRW.2012.6239225.*

[12] Ghorpade, Madhuri. User Authentication Using Keystroke Dynamics (2019), Computer Science Department, California State University.

[13] Lu Xiaofeng, Zhang Shengfei and Yi Shengwei, Continuous authentication by free-text keystroke based on CNN plus RNN, *International Conference on Identification, Information and Knowledge in the Internet of Things, IIKI 2018.*

[14] Alejandro Acien, Aythami Morales, Ruben Vera-Rodriguez and Julian Fierrez. TypeNet: Scaling up Keystroke Biometrics (2020).

Timeline

Task	**Date**
Project Commencement	30th August 2020
Literature Review	September and October 2020
Data exploration and Visualization	October 2020
Model building and Hyperparameter training	October and November 2020
Project Report	December 2020
Siamese Model building for benchmark dataset	December 2020
Building website for dataset collection	January 2021
Dataset Collection	February 2021
Data exploration and Visualization on custom dataset	February 2021 and March 2021
Model building and Hyperparameter training on custom dataset	March 2021 and April 2021
Project Report	May 2021
Report Submission	13th May 2021
Project Presentation	14th May 2021