Machine Learning for user Identification based on Text Typing Dynamics on Smartphones

Eustácio Domingos Muteia Cuatane
Department of Computer Engineering
Universidade Lúrio
Pemba, Mozambique
ecuatane@unilurio.ac.mz

Celso Vanimaly
Department of Computer Engineering
Universidade Lúrio
Pemba, Mozambique
celso.vanimaly@unilurio.ac.mz

Abstract— The increasing technological advancement and use of smartphones allows users to store a wide range of sensitive information, such as bank details, documents, social media data, passwords, contacts, among others. However, the loss or theft of these devices, along with vulnerabilities like password attacks, can seriously compromise the security of this data. Although traditional security mechanisms such as passwords and pattern locks have been widely used, their effectiveness has been questioned due to how easy they are to copy, forget, or share. Furthermore, they do not guarantee the link between an operation and the individual who carries it out. In response, many modern devices have adopted biometric technologies such as facial and fingerprint recognition, but their availability and cost can be limiting as their implementation requires special internal hardware which incurs more cost. Currently, machine learning has proven to be very effective in solving various problems in different areas. In this context, as another security alternative for these devices, we propose a low-cost and nonintrusive method using machine learning to identify users based on the dynamics of text typing on smartphones. To this end, we developed an Android application to collect typing data from 25 users, including the duration of each key pressed and the latencies between consecutive keys. We train and test models using algorithms such as Random Forest, XGBoost and LightGBM. The results show that the XGBoost model achieved the best performance, with an accuracy of 86.47% on the test set and 90.5% in a real environment using an API linked to the Android application.

Keywords— Typing dynamics, Biometrics, Smartphones, Machine Learning.

I. INTRODUCTION

The increasingly advanced technology of smart phones allows users to store sensitive data and private information, which therefore necessitates the need for better security mechanisms [1].As a way of protecting the information stored on these devices, most of them use authentication or identification techniques, usually through passwords or unlocking patterns. The purpose of such procedures is to provide additional evidence to authenticate the identity claim, i.e. to help confirm that the user is who they claim to be. The vulnerabilities presented by authentication mechanisms using passwords or unlocking patterns have led to that the use of physiological or behavioral biometrics is currently gaining ground. Among the various techniques, the dynamics of typing has been widely studied in various contexts. Each individual has a unique way of typing [2] . Thus, based on the idea that each individual's typing dynamics are unique, it is possible to identify them through this characteristic. This project implements machine learning techniques to help identify users based on the dynamics of typing text on the touchscreen of smartphones.

II. RELATED WORK

Various studies related to typing dynamics have been developed over time, most of them focused on desktop or personal computers. In recent years, this field has also been studied on mobile platforms such as smartphones. This section describes the work related to this study.

The authors in [3] develop biometric profiles using touch pressure, location and time. New interactions are authenticated in a profile using a metric. Classification achieves 100% accuracy in 3840.33 milliseconds on a Nexus 7 tablet mobile device. According to the authors, an Android program evaluates the classification time on a Nexus 7 tablet. The computation time increases with the total size of the model, requiring 1 second per 3333.33 interactions. Accuracy of 80% is achievable in less than 2 seconds - 90% in less than 3 seconds - 100% in less than 4 seconds.

The authors in [4] investigated whether a classifier can continuously authenticate users based on the way they interact on the touchscreen. They propose 30 features from raw data collected from the touchscreen of a smartphone. The proposed classifier achieves a median equal error rate of 0% for intrasession authentication, 2% - 3% for intersession authentication and below 4% when the authentication test was performed one week after the enrollment phase. They used two different classifiers: K-Nearest Neighbors and SVM Support Vector Machine. According to the authors, they obtained better results using the K-nearest neighbor classifier and the SVM also has a low error rate depending on the experiment.

In the authors' research [5]used machine learning algorithms to authenticate users through dynamic typing on the Android platform. Initially, the authors developed an Android application to collect the characteristics of 42 users, the application captures temporal and non-temporal characteristics (based on the touch screen), the users interacted with the application by typing the password (.tie5Roanl) 30 times in various sessions. Two Android devices were used to collect the data: a Nexus 7 tablet and an LG Optimus L7 II p710. The models were implemented using the Weka platform and for the temporal features the best performance was achieved using the random forest algorithm with 82.5% accuracy and when the touchscreen-based features were added the performance increased to 93% accuracy.

The authors in [6] developed an Android application, called iProfile, to collect keystroke events from Android devices, the application records all up and down events when the user

touches any key. The authors created their own keyboard and implemented the logic according to their needs. As in the work of [5] the password used in this research was (.tie5Roanl), according to the authors the choice of this password is justified by the possibility of forcing the user to navigate through different keyboard layouts. They used the SVM classifier and achieved a maximum accuracy of 97.4% for temporal, tactile and device-specific characteristics.

In the work carried out in [7]the authors developed an android application to collect the data and invited 30 subjects to participate in the process, the application collected temporal and non-temporal resources such as finger area and at the end of the experiment, using the Weka platform and the Random forest algorithm, the weighted results were true positive rate of 80.7%, false positive rate of 0.7%, precision of 81.3% and recall of 80.7%.

III. OVERVIEW AND OBJECTIVES

A. Identification by typing dynamics

The personal identification of users is associated with the possibility of accessing information, and is one of the main aspects to be considered in order to guarantee security. To guarantee this security, methods are adopted to authenticate users on desktop computers or mobile devices, such methods defining an entry point into the system. Normally, the user faces a password challenge and receives access only if they enter the correct password [4]Traditional user passwords and unlock screens are easily compromised as they are prone to fraud such as theft, forgetfulness, loss and copying. In this context, there is a notorious need to look for solutions to correctly identify the user when entering their credentials, however, user interactions with mobile devices provide robust human identity data because behavioral attributes during information entry constitute the password and are totally transparent to the user. This biometrics is referred to in the literature as keystroke dynamics. The term "typing dynamics" is also known in the literature as typing rhythm or typing pattern [8]It consists of establishing patterns based on the latency times between keystrokes and/or the duration of the pressure on each key, typing speeds, finger dimensions, frequency of typing errors, etc. These patterns can be obtained from a word, for example a password, or from continuous text.

B. Objectives of the study

The main aim of this study is to develop an automatic learning model capable of identifying users when they type text on a touchscreen. However, in order to achieve the main or general objective, we have defined some specific objectives:

- Develop a mobile application for collecting human characteristics when typing text;
- Build the Dataset;
- Implementing machine learning algorithms;
- Train and test the implemented algorithms;
- Evaluate the performance of the algorithms.

IV. METHODOLOGY

The methodology proposed in this study is applicable to all smartphones based on the Android system. Figure 1 illustrates the proposed architecture containing 4 (Four) main modules.

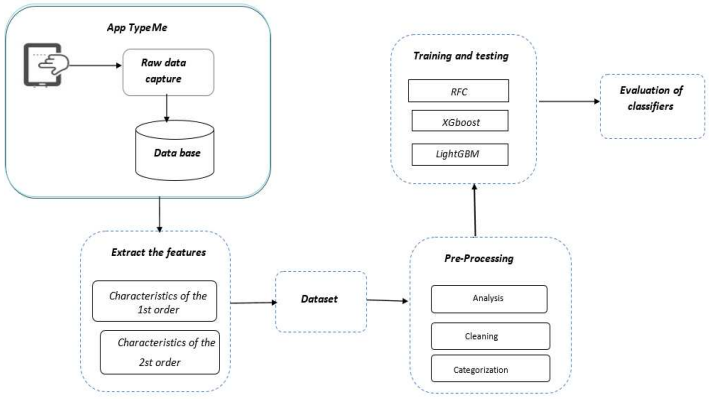

Figure 1: Architecture of the proposal

To carry out the first module, we had to develop an android application to capture human characteristics when typing texts. To do this, we invited 25 subjects to interact with the application by typing the target information "vida.nova" in 10 sessions and for each session they had to provide 50 typing samples with a rest interval of 1 (one) to 2 (two) days between sessions to better capture the degree of variability of each subject. For each keystroke, the application was responsible for storing 3 (three) features, 2 (two) based on time and 1 (one) representing the key. The time instants, Tp (press time) and Ts (release time) are captured according to the keystroke listeners incorporated into the application, so for the defined target information a total of 18 instants are captured for each repetition, divided into 9 Tp and 9 Ts. Figure 2 illustrates when the press and release times are captured.

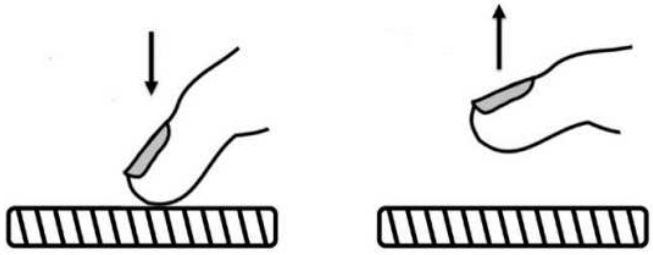

Figure 2: Pressing and releasing moments, adapted from [9]

After obtaining the raw data, the features were extracted. This stage was divided into two, namely: extraction of the 1st order characteristicsa where we extracted key duration (D), consecutive key press latency (PP), consecutive key release latency (SS) and consecutive key press-release latency (PS) and for the extraction of the 2nd order characteristicsa we calculated averages of the 1st order characteristics which together form the final dataset. In the pre-processing stage, we analyzed the entire dataset, checking that the dataset contained no null values, the number of samples per user and general information about the dataset, such as the number of rows and columns. For the cleaning stage, we decided to eliminate all users who contained less than 250 typing samples in their individual sample set and moved on to the categorization stage, which consisted of transforming the various classes representing the users' names into categories or labels using Scikit Learn's Label Encoding function. For the Training and Testing module, the scikit-learn 1.3.0 library was used to create three (3) models: Random Forest Classifier (RFC), eXtreme Gradient Boosting (XGBoost) and Light Gradient Boosting Machine (LightGBM). To this end, 80%

of the data was used for training and 20% for testing. The 3 (three) models were then built with the help of scikitlearn's GridSearch search grid using 5 (five) cross-validation folds.

V. RESULTS OF THE EXPERIMENTS

This section presents the results of the experiments carried out, starting with the mobile application developed, the data set collected and the results of the models trained.

A. Mobile application

The application developed consists of six (6) screens, four (4) available to the user and two (2) available to the administrator. The initial screen allows the user to enter their name and click on the "register" button if it is their first time, or they can enter their name and click on the "log in" button to have direct access to the characteristics collection screen if they have a record in the database. Figures 3 (a and b) illustrate all the application screens.

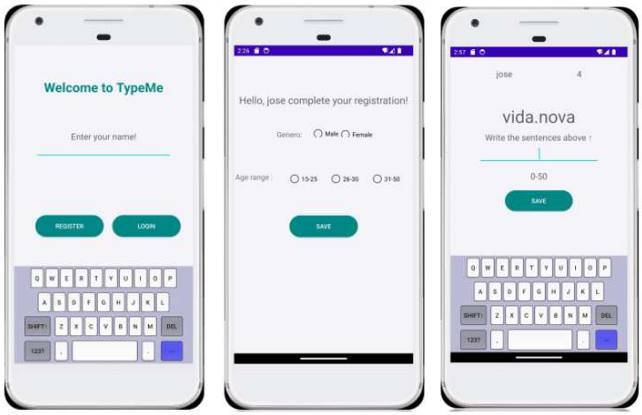

Figure 3-a: Developed android application

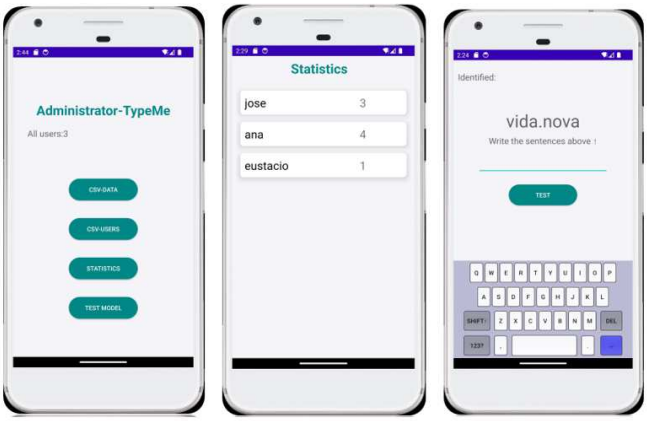

Figure 3-b: Developed android application

B. Dataset

The raw data set captured by the mobile application contains 105687 (one hundred and five thousand, six hundred and eighty-seven) rows and 6 columns that have not undergone any special treatment, organized in a csv file generated by the mobile application. The columns are structured as follows: user name, session identifier, repetition, the key representing the key pressed, the press and release times Tp and Ts. Table 1 illustrates an example of how the raw data set is structured.

Table 1: Raw data set

Name	idSession	Repetition	Key	TP	TS
Ana	1	1	V	4917513	4917662
Ana	1	1	I	4917824	4917930
Ana	1	1	D	4918682	4918788
Ana	1	1	A	4920034	4920150
Ana	1	1	.	4921042	4921169
Ana	1	1	N	4923102	4923229
Ana	1	1	O	4925171	4925331
Ana	1	1	V	4926147	4926274
Ana	1	1	A	4926951	4927057

After extracting the characteristics, we obtained a data set with 11743 (eleven thousand seven hundred and forty-three) samples structured as shown in table 2 and distributed as shown in figure 4.

Table 2: Data set with extracted characteristics

Name	idSession	Repetition	Duration.v	PP.v_i	SS.v_i	PS.v_i	...	Average_PP	Average_PS	Average_SS
Ana	1	1	0.149	0.311	0.268	0.417	...	0.12725	1.17975	1.301625
Ana	1	2	0.192	0.783	0.718	0.91	...	0.14475	1.122125	1.257375
Ana	1	3	0.138	0.31	0.311	0.449	...	0.1395	1.59625	1.737
Ana	1	4	0.159	0.782	0.761	0.92	...	0.148875	1.191375	1.33625

The initial idea was that each user would take 50 samples per session, for a total of 10 established sessions. However, not all users completed the sessions, which resulted in a dataset that was not completely balanced. Figure 4 illustrates the distribution of samples per subject.

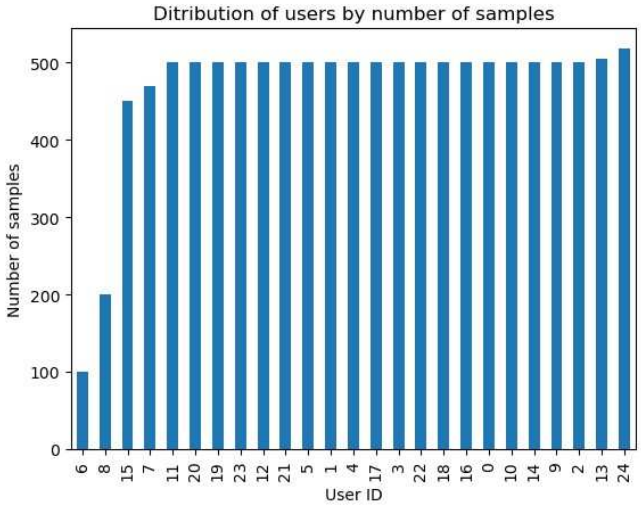

Figure 4: Number of samples per subject

C. Model training and testing results

As mentioned above, after passing the data through the pre-processing stage, we trained the models with 80% of the data and tested them with 20%. To build the models, we used the Gridsearchcv function with 5 cross-validation folds. Below we detail the training process for each of the models.

- Training the Random Forest Classifier model

The Random Forest Classifier model was built using the GridSearchCv function. This model was trained with a data set containing 9154 typing samples and to evaluate its performance it was tested with a never-before-seen data set with 2289 samples. Figure 5 illustrates the average cross validation accuracy score achieved by GridSearchCv in training in relation to the number of runs for different

hyperparameter configurations. The average score achieved was 78.9% accuracy.

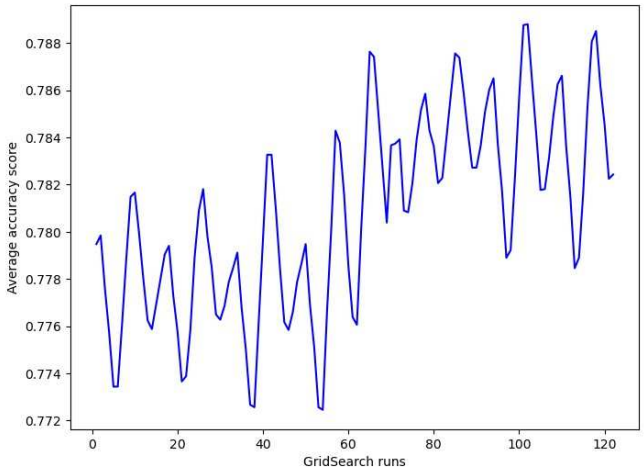

Figure 5: Average accuracy score in the cross-validation of the Random forest model

- Training the eXtreme Gradient Boosting model

The eXtreme Gradient Boosting model was also built using the GridSearchCv function. The model was trained with a dataset containing 9154 typing samples and, to evaluate its performance, it was tested with a dataset containing 2289 samples. Figure 6 illustrates the average cross-validation accuracy score achieved by GridSearchCv in training in relation to the number of runs for different hyperparameter configurations. The average score achieved for the best hyperparameter configuration was 84.05% accuracy.

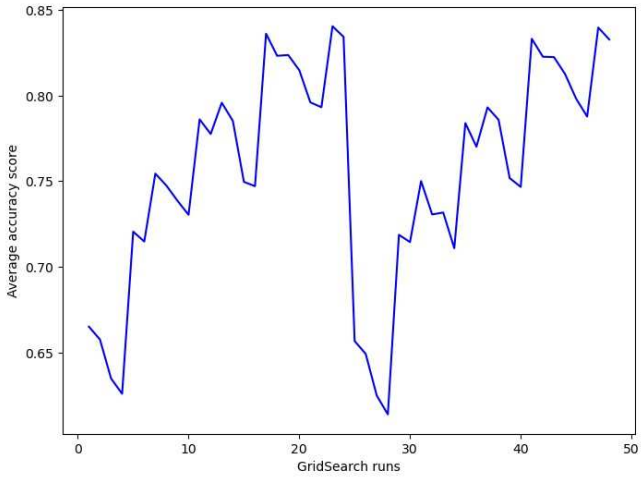

Figura 6: Average accuracy score in the cross-validation of the XGBoost model

- Training the Light Gradient Boosting Machine model

The Light Gradient Boosting Machine model, similar to the previous models mentioned, was also built using the GridSearchCv function. It was also trained with a dataset containing 9154 typing samples and to evaluate its performance it was tested with a never-before-seen dataset of 2289 samples. Figure 7 illustrates the average cross-validation accuracy score achieved by GridSearchCv in training in relation to the number of runs for different hyperparameter configurations. The average score achieved was 84.88% accuracy.

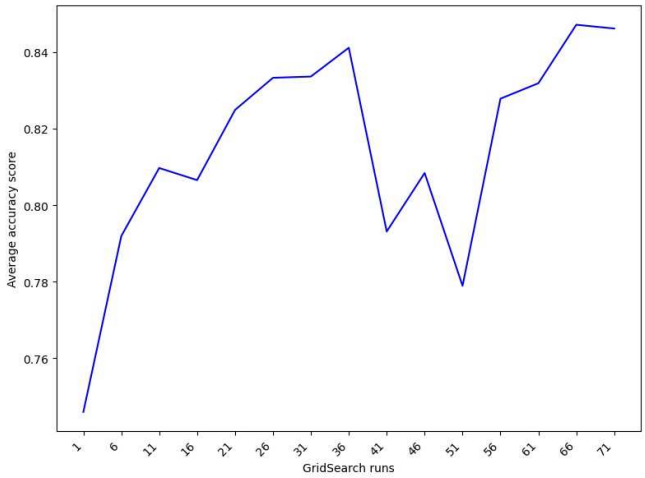

Figura 7: Average accuracy score in the cross-validation of the LightGBM model

Having developed the models, we moved on to the testing stage, the results of which are shown in Table 3. It is important to note here that as a further testing stage we developed an API where we integrated our best model and communicated with the android application in order to carry out tests in scenarios that represent real contexts in which smartphones are used in our day-to-day lives.

Table 3: Performance of the models in the test set

	Metrics			
Models	Accuracy (%)	Precision (%)	Recall (%)	F1-score (%)
Random Forest	81.00	81.16	81.00	80.94
XGBoost	**86.33**	**86.47**	**86.33**	**86.34**
LightGBM	86.06	86.16	86.06	86.03

After checking that the XGBoost model performed better than the others with the test data, we integrated this model with the developed API and linked it to the Android application (see figure 5). This integration enables tests to be carried out in a real environment, allowing the model's performance to be verified in practical situations.

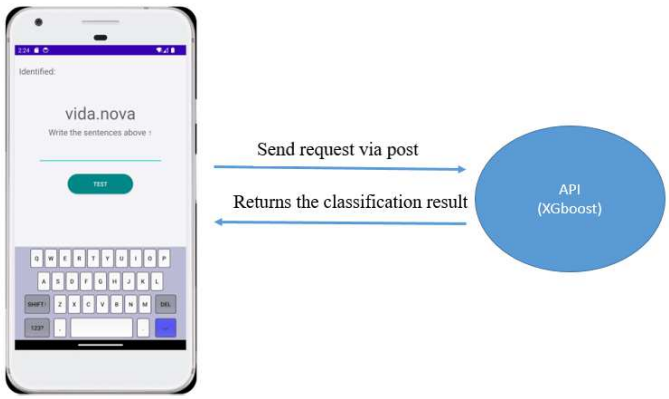

Figure 5: Communication between application and API

For this stage, we randomly selected 10 of the 25 subjects who took part in the data collection. These subjects contributed 20 typing samples each, totaling 200 samples for

the 10 users. The samples were sent to the API as the subjects typed in the target information, making it possible to make predictions in real time. The results of these tests are detailed in Table 4, providing a clear view of the model's performance in a real-use environment.

Table 4: Performance of XGboost in real-world tests

	Metrics			
Model	Accuracy (%)	Precision (%)	Recall (%)	F1-score (%)
XGBoost	90.00	**90.50**	90.00	89.87

The performance of the models in the tests with the test data and with the API developed was evaluated using the following metrics: Accuracy, Precision, Recall and F1-score, in order to verify the efficiency of each architecture trained to solve the problem in question in this research.

VI. DISCUSSION OF RESULTS

The general assessment of the performance of these models should be looked at from different points of view, some aspects should be highlighted such as the discrepancy in false positive rates between the different models for each of the users can be attributed to various reasons, including the specific characteristics of these models and the nature of the data, as it was observed that most of the users who have outstanding false positive rates remain for the different models. In this section, we discuss the factors that influenced the models' classification errors. However, after a rigorous and in-depth analysis, we found that the false positive rates found in the various models are related to the following factors:

- Variation in typing dynamics: typing dynamics vary between users, but can also vary within the same user over time. Changes in speed, key pressure and the user's own consistency affect the model's accuracy.
- Anomalous behavior: users with anomalous typing behavior (significant and unexpected deviations from a user's standard or typical behavior) can be more challenging to identify correctly. If these users have typing patterns that don't fit well with the characteristics used by the model, this can lead to false positives.
- Environmental variations: environmental factors, such as the location where the smartphone is being used, can influence typing dynamics. For example, typing on the move, in different positions or in noisy environments can introduce variations. The same aspect was also reported in [9].
- Device changes: this is related to the user's own ability to use several smartphones, as less skilled users are unable to maintain their behavioral pattern, especially on a smartphone they are not used to. In this particular study, we only used one (1) smartphone for data collection. However, for users who are less skilled in using smartphones, this aspect can have high implications for their behavioral pattern, leading to false positive rates.

In short, when evaluating the performance of a model trained to identify users based on the dynamics of text typing, especially on smartphones, one must take into account not only the user's own behavior but also the sensitivity of the data characteristics to the trained model, bearing in mind that each machine learning algorithm has its own philosophies and techniques.

VII. LIMITATIONS AND FUTURE WORK

A. Limitations

During the course of the research, we came across some limitations that may have influenced the solution reached in this study:

- Responses to touch features on smartphones: Some smartphones on which the preliminary version of the developed application was installed did not respond properly to touch events (finger pressure and area occupied by the finger) due to the capacity and sensitivity of the smartphone's touch screen. This aspect forced the research to use only temporal features in the raw data, since practically all smartphones responded well to this feature, and it is a cheap feature in terms of smartphone hardware requirements.
- Use of only one smartphone in the data collection process: This aspect had an influence on the length of the data collection process and may also have influenced the extreme variation in the typing behavior of some users and consequently made it difficult to classify the trained models correctly

B. Recommendations and future work

After evaluating the results achieved in this research, some points were found that need to be improved in order to obtain better results than those obtained.

- Building a more robust data set by adding new features and combining temporal and tactile features in order to carry out various experiments and build a more robust behavioral biometric model;
- Carry out studies using Deep Learning techniques in order to dispense with the process of manually extracting the characteristics from the raw data;
- Implement adaptation mechanisms in the trained models, since biometric behavioral characteristics undergo small changes over time;
- Carry out studies with a larger number of subjects

VIII. REFERENCES

[1] L. A. Gabralla, "Dense Deep Neural Network Architecture for Keystroke Dynamics Authentication in Mobile Phone," *Advances in Science, Technology and Engineering Systems,* vol. 5, pp. 1-2, 2020.

[2] P. Bours e S. Mondal, "A study on continuous authentication using a combination of keystroke and mouse biometrics," *Neurocomputing,* vol. 230, pp. 1-22, 2017.

[3] T. Dee, I. Richardson e A. Tyagi, "Continuous Transparent Mobile Device Touchscreen Soft Keyboard Biometric Authentication," *32nd International Conference on VLSI Design and 2019 18th International*

Conference on Embedded Systems (VLSID), pp. pp 1-2, 2019.

[4] M. Frank, R. Biedert, E. Ma, I. Martinovic e D. Song, "Touchalytics: On the Applicability of Touchscreen Input as a Behavioral Biometric for Continuous Authentication," *IEEE TRANSACTIONS ON INFORMATION FORENSICS AND SECURITY,* vol. 8, pp. 1-2, 2013.

[5] M. Antal, L. Z. Szabo e I. Laszlo, "Keystroke dynamics on Android Platform," em *8th International Conference interdisciplinarity in Engineering*, Romania, 2015.

[6] S. Krishnamoorthy, L. Rueda, S. Saad e H. Elmiligi, "Identification of User Behavioral Biometrics for Authentication Using Keystroke Dynamics and Machine Learning," em *2nd International Conference on Biometric Engineering and Applications*, 2018.

[7] N. Benjapatanamongkol e P. Bhattarakosol, "A Preliminary Study of Finger Area and Keystroke Dynamics Using Numeric Keypad With Random Numbers on Android Phones," Zhejiang University, Thailand, 2019.

[8] R. M. L. Filho, "Autenticação de alunos utilizando biometria comportamental em ambientes Juiz On-line," Manaus, 2022.

[9] P. S. Teh, N. Zhang, S. Tan, Q. Shi, W. H. Khoh e R. Nawaz1, "Strengthen user authentication on mobile devices by using user's touch dynamics pattern," em *Journal of Ambient Intelligence and Humanized Computing (2020) 11:4019–4039*, 2019.