Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 225 (2023) 1272–1281

Procedia Computer Science

www.elsevier.com/locate/procedia

27th International Conference on Knowledge-Based and Intelligent Information & Engineering Systems (KES 2023)

Evaluation of Deep Learning Models for Continuous Authentication Using Behavioral Biometrics

Utku Uslu[a], Özlem Durmaz İncel[b], Gülfem Işıklar Alptekin[a,*]

[a]Galatasaray University, Ciragan Cad, No:36, Istanbul, 34348, Turkey
[b]Bogazici University, Bebek, 34342, Turkey

Abstract

The traditional method to authenticate users on mobile devices or applications requires usernames and passwords. This approach authenticates the user only at the entry point of an application without providing continuous authentication during the whole session. This paper explores the use of behavioral biometrics, which involves tracking the unique movements of a user while interacting with a device, for continuous authentication on a mobile banking application. As a methodology, we use binary classification and explore the performance of deep learning algorithms. A dataset is collected from 45 participants using a mobile banking application in Turkey. We train four different types of deep architectures, including Multilayer Perceptron (MLP), LSTM, bi-directional LSTM, and convolutional LSTM. The dataset includes data from both touch screens and motion sensors. The results of the experiments reveal that MLP and the convolutional-LSTM algorithms achieve the best performance on raw data from both motion sensors and touch screens. Accuracy rates are over 99.85%, and FAR, FRR, and EER are below 0.5%.

Keywords: Continuous authentication; deep learning; behavioral biometrics; mobile banking

1. Introduction

Authenticating users on mobile devices is mainly based on the "something the user knows or owns" paradigm, which is also known as the knowledge-based authentication method [1, 2]. This type of authentication method includes usernames, passwords, and tokens, and it has been the most popular authentication system due to its simplicity. Users often use simple passwords or leave oily residues on the screen when drawing patterns that enable imposters to easily access the device and application contents [3]. As an alternative authentication approach, continuous authentication using behavioral biometrics is emerging [1]. It involves tracking the unique movements of users and detecting their patterns to identify them during a whole session while interacting with the device. This method usually uses

* Corresponding author. Tel.: +90-212-227-4480 ; fax: +90-212-227-5557.
E-mail address: gisiklar@gsu.edu.tr

1877-0509
10.1016/j.procs.2023.10.115

the dynamics of keystrokes, gait analysis, mouse movements, and touch gestures. The accuracy (ACC) rates of behavioral biometric authentication are often lower than the traditional methods [2]; however, they can be used as a complementary solution besides traditional schemes.

Mobile banking is one of the application areas where authentication is critical. Mobile banking applications use either a login name/password or physical biometrics as an authentication mechanism. Moreover, for certain transactions, a one-time password is used as a second-factor authentication. These mechanisms do not continuously authenticate the user during a session. If the user forgets to log out or an attacker captures the device, access to the user's account is not challenging.

This paper explores the use and performance of different deep learning (DL) algorithms to authenticate users on a mobile banking application with behavioral biometrics. The dataset was collected in our previous work [4]. It includes 45 participants performing 5 different mobile banking transactions in 3 different positions. The application is currently used by customers of a local bank in Turkey. For data collection, it is modified to collect touch screen (x-y coordinates on the screen), finger pressure, and sensor data from the motion sensors (gyroscope, accelerometer, magnetometer) to track the hand movements of a user. We consider the authentication process as a classification problem, whether the subject is the actual user or not, i.e., the imposter.

In our previous work [4], we explored the performance of traditional classification algorithms using 9 different classifiers. The biggest difference in this paper is that we did not evaluate the performance with DL algorithms. Besides we did not do feature engineering instead, we worked on raw data in our previous paper. In this paper, EER values are found to be lower than the ones in [4]. This work explores the authentication performance with four different DL algorithms: MLP, LSTM, bi-directional LSTM, and convolutional-LSTM. As mentioned in [5], handcrafted statistical features present low discriminative power, while DL algorithms can model the raw data better. We created three different datasets from the original one: 1) data with features [4], 2) data including the raw data from the touch screen, 3) data including both the raw sensor and the touch screen data. We trained these algorithms using different parameters to determine the best values and performed hyperparameter optimization. We used the following metrics to measure the performance of the models: false acceptance rate (FAR), false rejection rate (FRR), equal error rate (EER), true positive rate (TPR), and accuracy rate (ACC). We can summarize the contributions of this paper as follows:

- Although there are studies using DL algorithms for continuous authentication [7, 8, 9, 10, 12], they are mostly not associated with a real application. This paper uses a dataset collected from a real-life mobile banking application.
- We investigate the authentication performance with different DL algorithms and compare their performance both on data with extracted features and raw data.
- We show that higher performance can be achieved with the DL algorithms on raw data instead of a dataset with extracted features.

The rest of the paper is organized as follows: Section 2 summarizes related studies applying DL algorithms in continuous authentication. In Section 3 we present our proposed methodology for training and testing the models. In Section 4, we evaluate the performance of different DL architectures, and finally, in Section 5, we draw the conclusions and discuss how this work can be extended in future studies.

2. Related Work

In our previous work related to the continuous authentication domain [4], the same dataset was used. In that study, 9 different classifiers (support vector machine (SVM), MLP, KNN, decision tree, naive Bayes, one-class SVM, random forest, and ensemble algorithms) were trained. It was observed that the best average recognition performance was 3.5% (in terms of EER) with the SVM classifier. In that study, the performance was not evaluated with DL algorithms, and the raw data was not used. Instead, 126 features were extracted.

A comparison of related studies that utilize DL algorithms for continuous authentication is summarized in Table 1. CNN, LSTM, and MLP were frequently utilized, but their performances were not compared on the same dataset. Convolutional LSTM was not evaluated. The performance results in terms of TPR are generally higher than 90%,

Table 1. Comparisons with related studies.

Reference	Data	# of participants	Classifier	Performance
[4]	Sensor, touch (Dakota)	45	SVM, RF, MLP, KNN, decision tree, naive Bayes	TPR=99%, EER=3.5%
[7]	Motion sensors	20	CNN and SVM	Accuracy=95.01%
[8]	Sensor	84	LSTM	FRR=6.67%, FAR=0.95%
[9]	Touch	41	Deep Neural Network	Accuracy=94.2%, EER=3.46%
[10]	Sensor, touch (HMOG)	100	MLP	Accuracy=88%, EER=15%
[12]	Sensor, touch (HMOG)	100	Siamese CNN + SVM	Accuracy=97.8%
[13]	Sensor, touch (HMOG)	100	Two-stream CNN	Accuracy=90.04%, EER=5.14%
This study	Sensor, touch (Dakota)	45	CNN, LSTM, CNN+LSTM	Accuracy=99.87%, EER<0.5%

similar to our research results. We achieve FAR, FRR, and EER values of lower than 0.5% when the raw sensor and touch data are used. In our previous study [4], EER values were obtained as 3.5%, which was quite higher compared to the results of this paper.

3. Authentication Methodology

In this section, we explain the steps of our methodology (Figure 1): i) pre-processing, ii) feature extraction (optional), iii) training, iv) testing, and performance evaluation.

3.1. Dataset

The dataset was collected from 45 participants who use a mobile banking application. The currently used banking application was modified to log data from the touch screen and the motion sensors (gyroscope, accelerometer, magnetometer) on the phone. 5 popular transactions performed by the bank customers were identified, and all the users performed the same transactions during the data collection process. These transactions are as follows:

- T1: Account and credit card balance control on the dashboard
- T2: Account search on account list and balance control
- T3: Money transfer from one account to another
- T4: Foreign exchange buy operation
- T5: Credit card debt payment

Since users may use an application in different postures in real-life, they were also asked to repeat the same 5 transactions in 3 different postures:

- P1: Phone in hand and sitting
- P2: Phone in hand and standing
- P3: Phone on the table and sitting

Although different postures (i.e. walking) can be included, we think other postures are not very common when using a mobile banking application. The 45 subjects were mainly undergraduate/graduate students and faculty members at the university. There were also bank employees aged between 18 and 42. The data size is 1.9 GB (5 transactions in 3 different postures, which means 15 transactions per participant). Further details about the dataset can be found in [4].

For the experiments of this paper, we created 3 sub-datasets. The first one is constructed by extracting 126 features from the raw data. We aimed to compare the findings of this work with the previous results. 8 basic features are extracted from the sensor data: mean, standard deviation, median, minimum, maximum, variance, kurtosis, and skewness. These features are extracted from all 3 axes (x, y , z) of each motion sensor and the magnitude value from the 3 axes, which is the square root of the sums of squares of the values in each axis. 96 features are extracted from

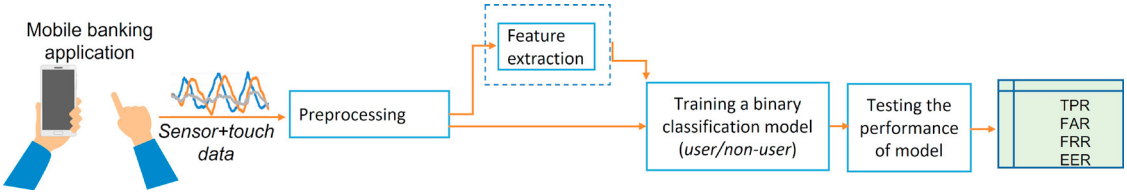

Fig. 1. Framework of authentication methodology.

the sensor readings. 30 more features are extracted from the touch screen readings, such as distance, the median of current pressure, and average velocity. We cannot give the complete list here due to space limitations, but explanations can be found in [4]. As the second sub-dataset, we used the raw touch screen readings, excluding the sensor data. As the third sub-dataset, we used the whole raw data from touch screen readings and sensor readings.

In the pre-processing step, we applied min-max normalization since not every feature and data type was in the same range. A binary classification approach is used, and data from a particular user are labelled as "user", and the data from others are labelled as "non-user". In this case, the class labels are imbalanced, with a ratio of 1:44. Since we have a majority of non-user class, the classifiers learn the non-user class better. In the literature [14], under-sampling or over-sampling approaches are used for this problem. Since we have limited data, instead of using under-sampling, we preferred an over-sampling method, namely "synthetic minority oversampling (SMOTE)" technique [15]. In SMOTE, an instance from the minority class is randomly selected, and its k-nearest minority-class neighbours are selected. A synthetic instance is created by randomly choosing one of the k nearest neighbours and connecting the instance and the neighbour instance to form a line segment [14]. We used $k = 2$ in our evaluations. We apply SMOTE only to the training data.

3.2. Training and Test Metrics

After these pre-processing steps, a model is trained for each user on the training data after partitioning the dataset into train and test sets. We used the split approach with a 2/3 ratio. In model creation, we experiment with MLP, LSTM, or convolutional-LSTM as the DL architectures, and we explain the details of the algorithms along with their parameters in Section 3.3. The metrics for evaluating the algorithm performances are selected as (Equations 1 to 4):

- FAR (false acceptance rate): Rate of unauthorized persons who are incorrectly accepted.
- FRR (false rejection rate): Rate of authorized persons who are incorrectly rejected.
- EER (equal error rate): The common value when FAR and FRR are equal.
- TPR (true positive rate): Rate of authorized persons who are correctly accepted.
- ACC: Accuracy of the model.

$$FAR = FP/(FP + TN) \tag{1}$$

$$FRR = FN/(FN + TP) \tag{2}$$

$$ACC = (TP + TN)/(TP + TN + FP + FN) \tag{3}$$

$$TPR = TP/(TP + FN) \tag{4}$$

In these equations, TP are the true positives (positive predictions that are really positive), FN are the false negatives (negative predictions, i.e., the imposter, that are actually positive, i.e., the user), FP are the false positives (positive predictions that are actually negative) and TN are the true negatives (negative predictions that are really negative).

3.3. Classifiers: Deep Learning Architectures

This section explains DL architectures and the associated hyperparameters. The MLP algorithm is a class of feed-forward artificial neural networks. This algorithm was evaluated in [4], but with fewer layers. In this paper, we in-

creased the layers to deepen the artificial neural network to observe its impact on authentication. We used the Scikit-Learn $MLPClassifier$ implementation in the evaluations.

The LSTM algorithm is a recurrent neural network (RNN) architecture. LSTM's main advantage is the existence of feedback connections; therefore, it can process data sequences efficiently. We changed the epoch number, activation function, dropout, and batch size parameters to find the best-performing model. The LSTM algorithm is implemented using the Keras library. Bi-directional LSTM works like the LSTM algorithm. Nonetheless, the amount of cells needed is double the number of cells in LSTM because LSTM only stores information from the past, but the bi-directional LSTM stores both information from the past and the future, so it also works backwards.

Besides, the CNN algorithm is used in combination with LSTM. Even though the CNN algorithm is primarily used in image classification, it does not require much pre-processing, making it easier to implement. In the CNN algorithm, filters are used to create a filter map to determine the features. Filter sizes are given as parameters. In the implementation (ConvLSTM2D library in Keras is used), the convolutions of the CNN are performed in the LSTM cell. Instead of LSTM interpreting the output from a CNN model, the convolutional LSTM uses convolutions as an input into the LSTM units [16].

3.4. Hyperparameters

The algorithms have the number of layers, the number of epochs, the learning rate, the batch size, the activation function, and the dropout rate as the list of parameters. Apart from these, the input (the data) is one of the crucial factors affecting the success of a model. In this section, we briefly summarize the function of these parameters, and in Section 4, we present the values selected for these hyperparameters.

Epoch is a complete run of the algorithm on the training data set. The higher the epoch number, the better the ML algorithm learns; however, one should be careful about overfitting. *Hidden layers* are mathematical functions and produce output specific to an expected result. DL algorithms are differentiated from classical artificial neural networks with a higher number of layers. Increasing the number of epochs or the number of hidden layers can increase the model's performance, but as the number of epochs or hidden layers increases, the time required to train a model also increases. In addition, there is always a risk of overfitting, which means that as the model works more on the training data, it starts to memorize the data and, therefore, may perform worse on the test data, hence cannot generalize. It is, therefore, essential to find the optimal number of epochs and hidden layers. The *dropout rate* is a parameter used to solve the problem of overfitting. When an epoch ends, depending on the dropout rate, some of the outputs generated from that epoch are ignored to prevent the learning dataset from being stored. The dropout rate is between 0 and 1, where 1 means all outputs are rejected, and 0 means all outputs are kept.

The *learning rate* is the speed at which the ML algorithm learns and updates the weights while building a model, and it is a part of the optimization process. A low learning rate can lead to a lengthy training process that could get stuck, while a high learning rate can cause weights to update too often; thus, it may cause the model to converge very quickly, even to a sub-optimal solution causing an unstable training process. *Batch size* is the number of samples passed before the weights are updated. A smaller batch size significantly increases training time, while a large batch size may lead to less precision. The *activation function* helps the artificial neural network to learn complex patterns in the data. The function takes the output of a previous cell and converts it to some form, then feeds the output into the next cell. Different activation functions compute the output differently. For example, the *Softmax* activation function is more efficient in solving multi-class classification problems, while the *Sigmoid* activation function is more efficient in solving binary classification problems. Usually, the *Relu* function is used by default in DL algorithms and is widely used in internal layers.

4. Performance Analysis

The authentication process is a binary classification operation. Among the data of 45 users, there is 1 user and 44 non-users data for each user. To resolve the imbalance of user and non-user data, we use SMOTE in the training dataset. We also want to label non-users successfully. Thus, to better assess the performance of the algorithms, we observe not only ACC and TPR but the error rates, including FAR, FRR, and EER. Not only on average, but the ACC must be high for all users.

Table 2. Average results with MLP on data with extracted features.

Activation Function	Learning Rate	# of Hidden Layers	FAR	FRR	EER	TPR	ACC
TanH	Constant	10	1.486	16.095	8.791	83.904	98.234
ReLu	Constant	10	0.527	15.523	8.025	84.476	99.189
ReLu	Adaptive	10	0.485	13.767	7.126	86.232	99.267
ReLu	Adaptive	12	0.552	14.253	7.403	85.746	99.193
ReLu	Adaptive	15	0.509	6.274	3.392	93.725	99.381

4.1. Dataset 1 - Evaluation with Extracted Features

After extracting the features, we trained the DL algorithms. There is no need to extract features while working with DL algorithms; however, we aimed to compare the findings of this work with the results reported previously on the same dataset where features were extracted [4].

4.1.1. MLP

As presented in Table 2, in each case, the ACC values are over 98%. Increasing the number of hidden layers usually increases ACC and TPRs and decreases FRR and FAR. At first, the ACC of some users (those with id's: 5, 80, 82, 84, 85, 86, 88, 97, 102) are observed to be lower than 98%. That also means their FARs are generally higher than 2%, unlike other users. After changing the activation function to ReLU, the overall ACC increased. We can say that the ReLU activation function performs better than tanh. The overall FAR also decreased. However, for some users, we still observed TPRs to be lower than 90% and FRRs over 10%. We then changed the learning rate to be adaptive, and this change also lowered the error rates.

When we analyze the results per person, we observe that FAR values for all users are below 4%, while FRR values are below 15%, except for 6 users. This means that the number of false negatives (incorrectly rejecting the user) is high, and their models may further need to be optimized.

4.1.2. LSTM

In all instances, the overall ACC were over 98% (Table 3). LSTM ACC rates are generally lower than the one of MLP, whereas TPR and FRR results are better than MLP. There were 9 users whose accuracy rates were lower than 98%. As the next step, we changed the activation function to the sigmoid. It is a suitable approach for binary classification since it returns either 0 or 1. We also lowered the number of epochs since training with 50 epochs took a long time. This time ACC rates were observed to be lower than before. However, the number of users with TPRs lower than 90% and FRRs over 10% decreased. We lowered the dropout rate to allow the model to remember more from previous states. The overall ACC and FAR improved this time, whereas FRR and TPR worsened. Next, we lowered the number of epochs to 15 to see the effect of time on the learning process. The overall ACC declined as expected. Surprisingly, the overall TPR increased, whereas the overall FRR decreased. This may show that even though a higher number of epochs results in better ACC, it does not mean that TPR and FRR successes will increase.

Table 3. Average results with LSTM and bi-directional LSTM on the data with extracted features

Architecture	Activation Function	Number of Epochs	Dropout Rate	Batch Size	FAR	FRR	EER	TPR	ACC
LSTM	Softmax	50	0.5	4	1.029	8.311	4.670	91.688	98.842
LSTM	Sigmoid	25	0.5	4	1.861	8.173	5.017	91.826	98.003
LSTM	Sigmoid	25	0.2	4	1.159	9.256	5.208	90.743	98.696
LSTM	Sigmoid	15	0.2	4	1.823	6.007	3.915	93.992	98.011
LSTM	Sigmoid	30	0.2	16	1.627	8.701	5.164	91.298	98.244
Bi-directional LSTM	Sigmoid	25	0.2	4	0.937	7.492	4.214	92.507	98.943

Table 4. Average result with the Convolutional-LSTM algorithm on the data with extracted features

Activation Function	Number of Epochs	Dropout Rate	Batch Size	FAR	FRR	EER	TPR	ACC
Softmax	25	0.5	4	0.962	11.498	6.230	88.501	98.836
Softmax	25	0.5	16	0.952	12.388	6.670	87.611	98.836
Sigmoid	25	0.5	16	1.059	12.032	6.545	87.967	98.744
Sigmoid	40	0.5	16	0.823	12.528	6.676	87.471	98.965
Sigmoid	20	0.5	8	0.960	8.694	4.827	91.305	98.845

Finally, we changed the batch size to 16 and the epoch number to 30 to observe the relation between them. The results did not change significantly. The ACC increased slightly, whereas TPR and FRR performances declined. However, ACC rates for two users were lower than 90%, performing worse than the other cases. Even though we want to improve overall scores, a decrease in individual scores is not preferred. After applying grid search, we get the following parameters for LSTM: the number of epochs: 25, activation function: sigmoid, dropout rate: 0.2, and batch size:4. As the next step, we experiment with bi-directional LSTM, using the same set of parameters from the grid-search analysis (presented as the last row of Table 3). Bi-directional-LSTM showed better results in every metric with an ACC of 98.94%, TPR of 92.5%, FRR of 7.49%, and FAR of 0.93% than simple LSTM. When we analyzed the results individually, there are only 3 users with an ACC lower than 98%. Hence, this is one of the best results achieved at the individual level.

4.1.3. Convolutional LSTM

As presented in Table 4, Convolutional LSTM's overall ACC rates are better than LSTM also at the individual level. However, the performance is less than the LSTM algorithm on TPR and FRR. At first, there were 8 users with ACC rates lower than 98%. The overall ACC was 98.83%. However, as there are 12 users with TPR of lower than 90%, and FRR of over 10%, the overall TPR and FRRs were not satisfying. Then, we changed the batch size from 4 to 16 to determine the performance level by upgrading the weights less frequently. The number of users with ACC of lower than 90% was 7, which did not change much. However, the number of users with TPR of lower than 90% and FRR of over 10% increased dramatically to 20. Then, we changed the activation function to sigmoid, which performs better on binary classification problems. The overall ACC slightly decreased; meanwhile, nearly the same TPR and FRR were obtained. But, the number of users with lower TPR, namely less than 90% and higher FRR with 10%, decreased to 13. This means individually, the error rates decreased. Furthermore, we tried different epoch and batch numbers. It turns out that having a smaller batch size is better because even though the overall ACC did not differ much and the number of users with an ACC of lower than 98% were the same, performance on TPR and FRR improved. The last experiment was the only time the algorithm's overall TPR was over 90%, and overall FRR was below 10%.

The best results with the convolutional-LSTM algorithm are obtained with the number of epochs of 20 and batch size of 8. Even though its overall ACC is not the best, the general performance exceeds the others in other metrics. The number of users with ACC lower than 98% did not differ much between different parameters, and these users were generally the same. This may mean that it is challenging to identify these users from others.

To summarize, a comparison of the algorithms with the best-performing hyperparameter values in terms of error rates is visualized in Figure 2 (left-most part, columns named as DATASET-1). MLP and LSTM algorithms perform better than the others. While LSTM achieves a slightly smaller FRR, MLP is better regarding EER and FAR. Bi-directional LSTM and convolutional LSTM achieve lower FAR than LSTM; however, their performances in FRR and EER are slightly higher than simple LSTM. When we compare these findings with the results presented in [4], again, MLP was the best-performing classifier, and similar values are reported on the dataset with features.

4.2. Dataset 2 - Evaluation on Raw Touch Data

In this section, we work on raw data and investigate the performance only on touch data before applying the algorithms on both sensor and touch screen data. Again, we apply MLP, LSTM, and convolutional-LSTM algorithms on touch (scroll) data to determine if we can authenticate users only by the data extracted from touch screen movements.

Table 5. Authentication performance with raw data

Data	Algorithm	FAR	FRR	EER	TPR	ACC
Touch screen	MLP	1.115	2.621	1.8684	97.378	98.845
	LSTM	4.484	6.058	5.271	93.941	95.472
	Bi-LSTM	6.032	7.561	6.796	92.438	93.981
	Conv-LSTM	2.406	3.064	2.735	96.935	97.564
Touch screen + Sensors	MLP	0.129	0.028	0.079	99.971	99.872
	LSTM	0.394	0.031	0.212	99.969	99.612
	Bi-LSTM	0.337	0.122	0.230	99.877	99.720
	Conv-LSTM	0.137	0.336	0.237	99.663	99.856

The results are presented in Table 5 in the first rows named as *Touch screen*. The parameters of the classifiers are as follows: i) MLP: Activation: Relu, Learning Rate: Adaptive, Hidden Layer Size: 15, ii) LSTM: Epoch: 10, Activation: Sigmoid, Dropout: 0.2, Batch Size: 16, iii) Bi-LSTM: Epoch: 10, Activation: Sigmoid, Dropout: 0.2, Batch Size: 16, iv) Conv-LSTM: Epoch: 10, Activation: Sigmoid, Dropout: 0.5, Batch Size: 16. Firstly, MLP is the best-performing algorithm applied on touch screen data. If we investigate the ACC rates individually, we see that no user had an ACC lower than 95%. Also, there are 3 users having TPR lower than 90% and FRR 10%. These numbers show that scroll data alone cannot provide authentication to some users as compared to using the sensors we discuss in Section 4.3. LSTM algorithm did not perform well compared to others. 6 users have ACC of lower than 90%. Convolutional-LSTM performed better than LSTM. There were no fluctuations between TPR and FRRs, like in the LSTM algorithm. Overall, we can say that even though the MLP algorithm performed the best on scroll data, neither of the algorithms successfully authenticated all users. Therefore, in the following section, we explore the performance by also using the sensor data.

4.3. Dataset 3 - Evaluation on Raw Sensor and Touch Data

We trained all four algorithms with the optimal parameters that we obtained from the previous experiments that was generated from raw data (in rows named as *touch screen+sensors* of Table 5). Again, the same set of parameters in Section 4.2 are used. Compared with using only touch data, the MLP algorithm on raw data performed better. The overall ACC is 99.87%. The TPR of the algorithm is 99.97%, the MLP algorithm on raw data is successful in authenticating all users. LSTM algorithm on raw data also performs quite satisfactorily even though its overall ACC is slightly lower than MLP. Individually, 40/45 users' ACC rates are over 99%. TPR and FRRs of LSTM are reasonable; it successfully authenticates all users since there is no single user with significantly worse results than others. The bi-directional LSTM algorithm achieved slightly better results in ACC than LSTM. According to the overall TPR and ACC rate, bi-directional LSTM can be said to be successfully authenticating users using raw behavioral biometric data. Finally, the convolutional-LSTM algorithm has given slightly better results in ACC compared to LSTM and bi-directional LSTM. It successfully authenticates all users.

A comparison of the algorithms with the best-performing hyperparameters on 3 datasets is summarized in Figure 2. As expected, DL performed better on raw data, since the number of instances in raw data is larger than those with extracted features. Essential characteristics may be lost during feature extraction. Overall, all of these algorithms had FAR, FRR, and EER values below 0.5% when data from both touch screens and sensors were utilized (Dataset-3). When only touch screen data is used (Dataset-2), we can achieve lower error rates than working on data with features (Dataset-3), but the addition of sensor data significantly improves the results (Dataset-3). Mainly, MLP and Convolutional LSTM algorithms achieve the minimum error rates and the maximum TPR and ACC.

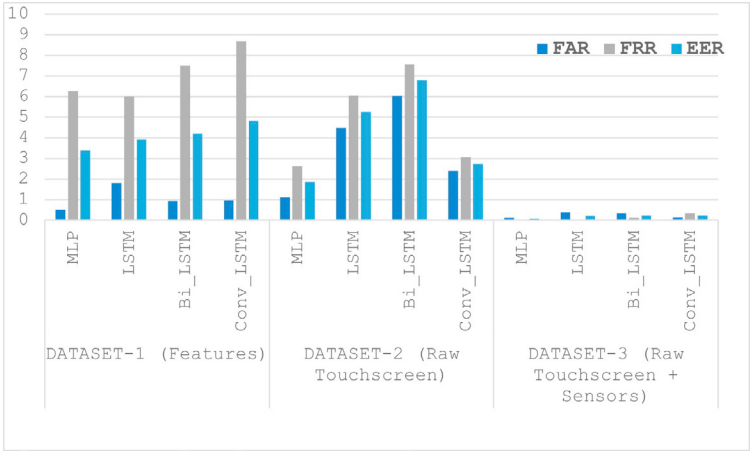

Fig. 2. Comparison of the algorithms on raw data with the best hyperparameter values.

5. Conclusion and Discussion

In this paper, we applied DL algorithms to authenticate users with their behavioral biometrics. The best data set to perform the authentication process is the raw data set, as all four DL algorithms' best results were obtained by working on this data set. In the following, we summarize our findings and report possible directions for future research.

- On-device implementation: It may be argued that running DL algorithms is challenging on a mobile phone, and these algorithms may not be utilized in a real implementation. Here, a solution could be to run the training and inference on a cloud platform, or there are certain techniques and platforms, such as TensorFlowLite[17], to optimize the DL algorithms to run on mobile devices [5]. In the second option, training can again be performed offline. As future work, we plan to collect more data and implement resource-aware versions of algorithms for smartphones.
- Performance: The best-performing algorithms are MLP and Convolutional LSTM with ACC rates of over 99.85%, and FAR, FRR, and EER of below 50%.
- Limitations and possible extensions: The dataset was obtained from two different phone models and 45 users. As future work, these can be extended. When the number of users increases, biometric diversity within the negative class may impact the results. However, the algorithms can be optimized accordingly. For some users, FRR values were higher than other users, and further optimization may be required. Instead of looking at each scroll event, authenticating over a session and using smoothing methods, such as majority voting can be used [4]. As future work, we will use F-score to measure the performance of the models, which may be appropriate for imbalanced datasets like the one in this paper.
- Circumventing DL-based authentication models: Continuous authentication on mobile banking applications analyze unique movements captured by mobile phone sensors. DL models are often employed to identify users based on these interaction patterns. However, there are potential ways to circumvent these models. One method is to perturb the sensor data by adding small adverserial noise to confuse the model. Generative adverserial networks can generate synthetic sensor data that closely mimics the actual user's movement patterns. Another technique involves mimicry attacks, in which an attacker aims to imitate the unique sensor-based movements of the targeted user. This can be achieved by collecting data on the user's interaction patterns and using it to create realistic synthetic data that closely mimic their movements. The attacker can then use this data to impersonate the user and bypass continuous authentication mechanisms. The proposed models assume that all information received from the mobile phone is secure and cannot be accessed from outside. DL models have certain limitations, which can be exploited to circumvent their continuous authentication capabilities. If a model has been overfitted to its training data, it may struggle to generalize to new user movements. Attackers can exploit this weakness by introducing new interaction patterns the model has not encountered. It is crucial

for developers to be aware of these techniques and develop robust countermeasures to ensure the security of their applications by using additional measures such as multi-factor authentication.

Acknowledgment

This research has been financially supported by the Galatasaray University Research Fund, project number: FBA-2022-1073, and also by the Bogazici University Research Fund, project number: 19301P. We thank Secil Gunay, Yasemin Akan, Okan Engin Basar and Yunus Barlas for their support in the data collection process.

References

[1] Stylios, I., Kokolakis, S., Thanou, O., and Chatzis, S. (2021). Behavioral biometrics & continuous user authentication on mobile devices: A survey. Information Fusion, 66, 76-99.
[2] Bailey, K. O., Okolica, J. S., and Peterson, G. L. (2014). User identification and authentication using multi-modal behavioral biometrics. Computers & Security, 43, 77-89.
[3] Patel, V. M., Chellappa, R., Chandra, D., and Barbello, B. (2016). Continuous user authentication on mobile devices: Recent progress and remaining challenges. IEEE Signal Processing Magazine, 33(4), 49-61.
[4] Incel, Ö. D., Günay, S., Akan, Y., Barlas, Y., Basar, O. E., Alptekin, G. I., and Isbilen, M. (2021). DAKOTA: Sensor and Touch Screen-Based Continuous Authentication on a Mobile Banking Application. IEEE Access, 9, 38943-38960.
[5] Wang, C., Xiao, Y., Gao, X., Li, L., & Wang, J. (2021). A Framework for Behavioral Biometric Authentication using Deep Metric Learning on Mobile Devices. IEEE Transactions on Mobile Computing.
[6] Barlas, Y., Basar, O. E., Akan, Y., Isbilen, M., Alptekin, G. I., and Incel, O. D. (2020, September). DAKOTA: Continuous authentication with behavioral biometrics in a mobile banking application. In 2020 5th International Conference on Computer Science and Engineering (UBMK) (pp. 1-6). IEEE.
[7] Zhu, T., Weng, Z., Chen, G., and Fu, L. (2020). A hybrid deep learning system for real-world mobile user authentication using motion sensors. Sensors, 20(14), 3876.
[8] Abuhamad, M., Abuhmed, T., Mohaisen, D., and Nyang, D. (2020). AUToSen: Deep-learning-based implicit continuous authentication using smartphone sensors. IEEE Internet of Things Journal, 7(6), 5008-5020.
[9] Al-Dori, Aws Saood Mohamed. "Touchscreen-based Smartphone Continuous Authentication System (SCAS) using Deep Neural Network." Turkish Journal of Computer and Mathematics Education (TURCOMAT) 12.11 (2021): 2382-2391.
[10] Volaka, H. C., Alptekin, G., Basar, O. E., Isbilen, M., and Incel, O. D. (2019). Towards continuous authentication on mobile phones using deep learning models. Procedia Computer Science, 155, 177-184.
[11] Sitová, Z., Šeděnka, J., Yang, Q., Peng, G., Zhou, G., Gasti, P., and Balagani, K. S. (2015). HMOG: New behavioral biometric features for continuous authentication of smartphone users. IEEE Transactions on Information Forensics and Security, 11(5), 877-892.
[12] Centeno, M. P., Guan, Y., and van Moorsel, A. (2018, June). Mobile based continuous authentication using deep features. In Proceedings of the 2nd International Workshop on Embedded and Mobile Deep Learning (pp. 19-24).
[13] Li, Y., Hu, H., Zhu, Z., and Zhou, G. (2020). SCANet: sensor-based continuous authentication with two-stream convolutional neural networks. ACM Transactions on Sensor Networks (TOSN), 16(3), 1-27.
[14] He, H., and Ma, Y. (Eds.). (2013). Imbalanced learning: foundations, algorithms, and applications.
[15] Chawla, N. V., Bowyer, K. W., Hall, L. O., and Kegelmeyer, W. P. (2002). SMOTE: synthetic minority over-sampling technique. Journal of artificial intelligence research, 16, 321-357.
[16] Sainath, T. N., Vinyals, O., Senior, A., and Sak, H. (2015, April). Convolutional, long short-term memory, fully connected deep neural networks. In 2015 IEEE international conference on acoustics, speech and signal processing (ICASSP) (pp. 4580-4584). IEEE.
[17] Abadi M, et al. Tensorflow: A system for large-scale machine learning. In: 12th USENIX symposium on operating systems design and implementation (OSDI 16); Savannah, GA, USA; 2016. pp. 265-283.