Fingerprinting Smartphone Accelerometers with Traditional Classifiers and Deep Learning Networks

Adriana Berdich, Patricia Iosif, Camelia Burlacu, Alfred Anistoroaei and Bogdan Groza
Faculty of Automatics and Computers, Politehnica University of Timisoara, Romania
Email: adriana.berdich@aut.upt.ro, {patricia.iosif, camelia.burlacu}@student.upt.ro, {alfred.anistoroaei, bogdan.groza}@aut.upt.ro

***Abstract*—Fingerprinting smartphones using their accelerometers has several applications, including activity recognition, driving style classification and device to device authentication. In this work, we study accelerometer-based smartphone fingerprinting. We gather data from mobile devices placed together to record identical vibrations. Then, we extract time domain features, which we use to train multiple traditional machine learning algorithms based on statistical properties of the data. Finally, we use the raw data in a more complex Convolutional Neural Network and compare the results. To make the investigations more challenging, we discuss fingerprinting both distinct and identical smartphones and reach an accuracy close to 100% with several traditional classifiers.**

***Index Terms*—smartphone fingerprinting, accelerometer, time domain features, machine learning, CNN**

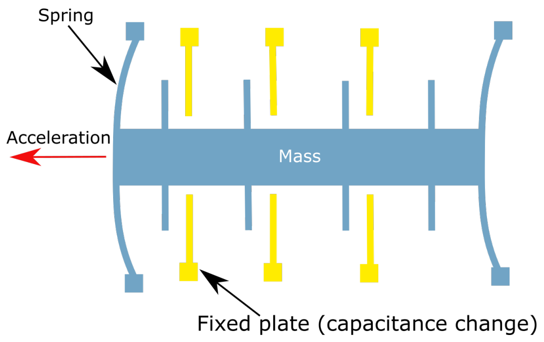


Fig. 1. Accelerometer sensor

I. Introduction

Nowadays, smartphones are likely the most popular gadget and their performance has drastically increased. They are also equipped with several sensors that allow the user to complete everyday tasks more swiftly and efficiently. The most crucial activity which is frequently performed with a smartphone is device-to-device authentication. In recent years, authentication based on sensor fingerprints has been increasingly explored.

Several approaches for fingerprinting smartphones based on their sensors have been proposed in the literature, focusing on microphones [1], [2], loudspeakers [3], [4], [5], gyroscopes [6], etc. Regardless of the sensor type, sensor characteristics may be unstable due to influences by the environment which make identification more challenging. In this work, we discuss smartphone fingerprinting based on accelerometer imperfections. Each smartphones is equipped with MEMS (Micro-Electromechanical Systems) accelerometer sensors, with the schematic suggested in Figure 1, which have unpredictable characteristics due to chemical, physical and geometrical imperfections introduced during manufacturing.

Accelerometers have been proposed for pairing mobile devices almost two decades ago [7], [8]. Still, only a few works were concerned with smartphone fingerprinting based on accelerometer data. The authors of [9] proposed a method for accelerometer fingerprinting based on 8 time domain features and 10 frequency domain features. Later, the authors from [10] used 8 time domain features along with a thresholding method to fingerprint accelerometers for device authentication. More recently, the authors from [11] used 10 time domain features and 10 frequency domain features with a combination of one-class and multi-class classifiers. Accelerometers and microphones were independently classified in [12].

Several works used multiple sensors for smartphone fingerprinting. The accelerometer, gyroscope and camera were used in [13], while the authors from [14] used the accelerometer, gyroscope, magnetometer and microphone. Many works used only the accelerometer and gyroscope for device fingerprinting, i.e., [15], [16], [17], [18]. The accelerometer, camera, loudspeaker and wireless transmitter were employed in [19] for device identification. In [20], once again, data from several sensors were used, i.e., accelerometer, gyroscope, orientation, magnetic field, rotation vector, linear acceleration, and gravity. Software fingerprinting based on the accelerometer and gyroscope was discussed in [21] and [22].

Several works used accelerometer sensors for other objectives. A survey on driver behavior detection using accelerometer sensors was conducted in [23]. The same topic was discussed in [24] and [25]. Driver identification based on data acquired from the accelerometer was discussed in [26] and [27]. Driving style recognition and driver classification based on motion sensors was proposed in [28] and [29]. The authors of [30] proposed a method for transportation mean detection based on accelerometer and GPS data. Also, [31] and [32] discuss transportation mode recognition using data from accelerometers. Real-time pothole detection using accelerometer data was discussed in [33]. In [34], a method for road condition monitoring using the accelerometer and GPS sensors was proposed. In-vehicle seat detection using smartphone motion sensors, i.e., accelerometer and gyroscope, was proposed in [35]. Gait recognition based on accelerometer data is discussed in [36].

The authors of [37] develop a system for detecting activity at metro stations based on smartphone sensors. Gait recognition using accelerometer data and adapted Gaussian mixture models was proposed in [38]. Activity recognition using smartphone sensors, including the accelerometer, was discussed in [39]. The authors from [40] present a method for tracking metro rides using the accelerometer. The work in [41] proposes a system for device to device authentication based on accelerometer data collected in different transportation modes, i.e, car, train, tram, bike, walk and shaking. Device-to-device pairing based on accelerometer data is also proposed in [42], [43] and [44].

II. Results with traditional classifiers

In this section, we discuss data collection, analyze collected signals from the accelerometer sensors, extract several time domain features and use traditional machine learning algorithms to classify smartphones.

A. Data collection

For data collection, we developed an Android application that acquires data from the phone's accelerometer at a sampling rate of 10ms. The data is collected simultaneously from all smartphones in an environment with vibrations. Then, the acquired data was saved in a text file to be used for signal analysis. Each file contains data collected for approximately 40 minutes, i.e., around 240.000 samples. As accelerometer data spans on 3-axes, i.e., X, Y, and Z, we first compute the square for each axis, sum the results and then extract the square root as follows in order to make the measurements independent from the phone orientation: $a = \sqrt{a_X^2 + a_Y^2 + a_Z^2}$.

To perform the classification, we split the data collected from each smartphone into 200 signals with 1.000 samples, i.e., 10 seconds each. Then, we randomly split the signals into training and testing data, starting from 20% training and 80% testing up to 80% training and 20% testing, with an increment step of 10%. Next, we process and analyze the signals and apply several traditional machine learning algorithms, i.e., NN (Wide Neural Network), Ensemble, KNN (K-Nearest Neighbors), SVM (Support Vector Machine), and Decision Tree. In the next section, we will also train a more complex CNN (Convolutional Neural Network) and finally compare the results.

To make the fingerprinting process more challenging, we use the 5 identical and 5 different smartphones to collect data and test the machine learning algorithms. These devices are five different smartphones: LG Optimus P700 (A), Samsung Galaxy S7 (B), Samsung Galaxy A21s (C), Samsung Galaxy J5 (D) and Allview V1 Viper I (E), and then the following five identical Samsung J5 (labels F to J) .

B. Signal processing and machine learning algorithms

In Figure 2 we depict the concept overview. From each signal, we extract the following time-domain features:

1) Kurtosis - the tailedness of a signal,

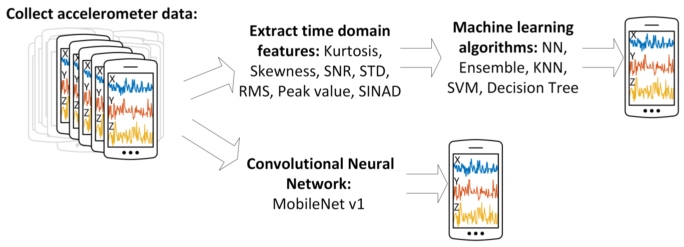
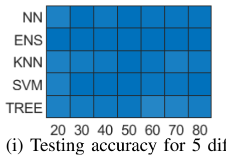
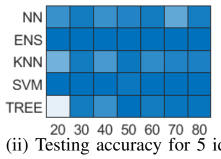

Collect accelerometer data:
X
Y
Z
Extract time domain features: Kurtosis, Skewness, SNR, STD, RMS, Peak value, SINAD
Machine learning algorithms: NN, Ensemble, KNN, SVM, Decision Tree
Convolutional Neural Network: MobileNet v1

Fig. 2. Concept overview



	20	30	40	50	60	70	80
NN	0.975	1.0	0.987	1.0	0.991	0.992	0.993
ENS	1.0	1.0	1.0	1.0	1.0	1.0	1.0
KNN	0.975	0.983	0.987	1.0	0.991	0.978	0.98
SVM	0.975	1.0	1.0	1.0	0.991	0.992	1.0
TREE	0.975	0.983	0.987	0.99	0.966	0.971	0.981

(i) Testing accuracy for 5 different accelerometers



	20	30	40	50	60	70	80
NN	0.950	0.983	0.950	0.970	0.983	0.914	0.981
ENS	1.0	1.0	1.0	1.0	1.0	1.0	1.0
KNN	0.900	0.983	0.937	0.990	0.958	0.971	0.975
SVM	1.0	1.0	1.0	0.990	1.0	1.0	1.0
TREE	0.800	0.983	0.950	1.0	0.983	0.992	0.993

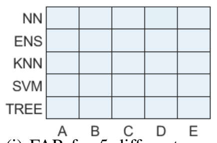
(ii) Testing accuracy for 5 identical accelerometers

Fig. 3. Testing accuracy as heatmap (left) and numerical values

2) Skewness - the asymmetry of the signal around the mean value,
3) SNR (Signal-to-Noise Ratio), i.e., the ratio between the power of the signal and the power of the noise,
4) STD (Standard deviation) - the square root of the variance of the signal,
5) RMS (Root-Mean-Square) - the mean of the squares along several samples from the signal,
6) Peak value - the maximum value of the signal,
7) SINAD (Signal to Noise and Distortion Ratio) - the ratio between the power of the signal and that of the noise and distortions.

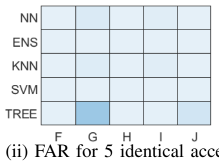
The extracted time domain features are finally sent as input to several classification algorithms to learn the smartphones characteristics based on their accelerometers. We use the following five machine learning algorithms that are available in Matlab:

1) NN (Wide Neural Network) is a simple neural network which contains an input layer of 7 neurons, a fully connected layer that has 100 neurons, a rectified linear unit (ReLU) to activate the fully connected layer, another fully connected layer with 5 outputs that correspond to the 5 smartphones and, finally, a Softmax function,
2) ENS (Ensemble - Subspace Discriminant) has a subspace dimension of 7, which corresponds to the number of time domain feature extracted from the accelerometer signals and requires 30 learning cycles,
3) KNN (K-Nearest Neighbors) uses the Euclidean distance and 2 neighbors to classify the smartphones,
4) SVM (Support Vector Machine) uses the linear kernel function, one-vs-one coding method, one box constraint level and the automatic kernel scale,



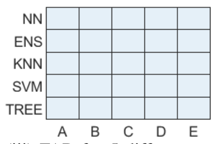
	A	B	C	D	E
NN	0	0	0	0	0
ENS	0	0	0	0	0
KNN	0	0	0	0	0
SVM	0	0	0	0	0
TREE	0	0	0	0	0

(i) FAR for 5 different accelerometers for 20% training



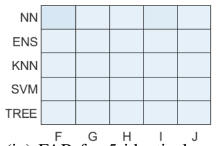
	F	G	H	I	J
NN	0	0.046875	0	0.023	0
ENS	0	0	0	0	0
KNN	0	0.039	0	0.023	0
SVM	0	0	0	0	0.007
TREE	0.007	0.320	0	0.007	0.085

(ii) FAR for 5 identical accelerometers for 20% training



	A	B	C	D	E
NN	0	0	0	0.007	0
ENS	0	0	0	0	0
KMN	0	0	0	0	0
SVM	0	0	0	0	0
TREE	0	0	0	0	0

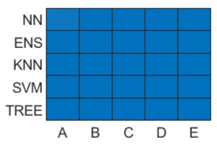
(iii) FAR for 5 different accelerometers for 80% training



	F	G	H	I	J
NN	0.062	0	0	0.031	0
ENS	0	0	0	0	0
KNN	0	0	0	0	0
SVM	0	0	0	0	0
TREE	0	0	0	0	0

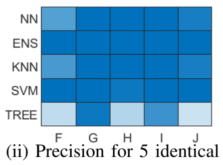
(iv) FAR for 5 identical accelerometers for 80% training

Fig. 4. FARs as heatmap (left) and numerical values (right)



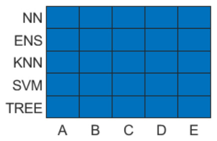
	A	B	C	D	E
NN	0.968	1.0	1.0	1.0	1.0
ENS	1.0	1.0	1.0	1.0	1.0
KNN	1.0	1.0	1.0	1.0	1.0
SVM	1.0	1.0	1.0	1.0	1.0
TREE	1.0	1.0	1.0	1.0	1.0

(i) Precision for 5 different accelerometers for 20% training



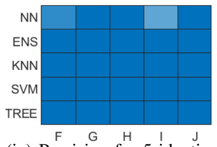
	F	G	H	I	J
NN	0.781	1.0	1.0	1.0	0.937
ENS	1.0	1.0	1.0	1.0	1.0
KNN	0.812	1.0	1.0	1.0	0.937
SVM	1.0	1.0	0.968	1.0	1.0
TREE	0.500	0.968	0.531	0.843	0.468

(ii) Precision for 5 identical accelerometers for 20% training



	A	B	C	D	E
NN	1.0	1.0	1.0	1.0	1.0
ENS	1.0	1.0	1.0	1.0	1.0
KNN	1.0	1.0	1.0	1.0	1.0
SVM	1.0	1.0	1.0	1.0	1.0
TREE	1.0	1.0	1.0	1.0	1.0

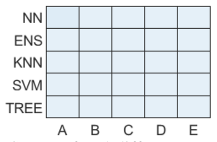
(iii) Precision for 5 different accelerometers for 80% training



	F	G	H	I	J
NN	0.875	1.0	1.0	0.75	1.0
EMS	1.0	1.0	1.0	1.0	1.0
KNN	1.0	1.0	1.0	1.0	1.0
SVM	1.0	1.0	1.0	1.0	1.0
TREE	1.0	1.0	1.0	1.0	1.0

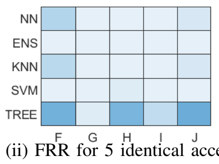
(iv) Precision for 5 identical accelerometers for 80% training

Fig. 6. Precision as heatmap (left) and numerical values (right)



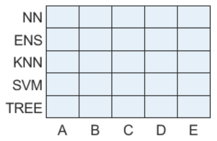
	A	B	C	D	E
NN	0.031	0	0	0	0
ENS	0	0	0	0	0
KNN	0	0	0	0	0
SVM	0	0	0	0	0
TREE	0	0	0	0	0

(i) FRR for 5 different accelerometers for 20% training



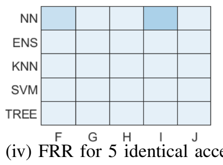
	F	G	H	I	J
NN	0.218	0	0	0	0.062
ENS	0	0	0	0	0
KNN	0.187	0	0	0	0.062
SVM	0	0	0.031	0	0
TREE	0.500	0.031	0.468	0.156	0.531

(ii) FRR for 5 identical accelerometers for 20% training



	A	B	C	D	E
NN	0	0	0	0	0
ENS	0	0	0	0	0
KNN	0	0	0	0	0
SVM	0	0	0	0	0
TREE	0	0	0	0	0

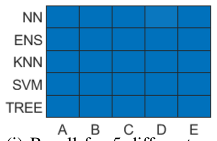
(iii) FRR for 5 different accelerometers for 80% training



	F	G	H	I	J
NN	0.125	0	0	0.250	0
ENS	0	0	0	0	0
KNN	0	0	0	0	0
SVM	0	0	0	0	0
TREE	0	0	0	0	0

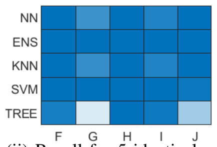
(iv) FRR for 5 identical accelerometers for 80% training

Fig. 5. FRRs as heatmap (left) and numerical values (right)



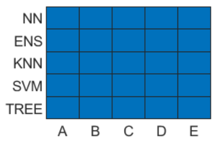
	A	B	C	D	E
NN	1.0	1.0	1.0	0.969	1.0
ENS	1.0	1.0	1.0	1.0	1.0
KNN	1.0	1.0	1.0	1.0	1.0
SVM	1.0	1.0	1.0	1.0	1.0
TREE	1.0	1.0	1.0	1.0	1.0

(i) Recall for 5 different accelerometers for 20% training



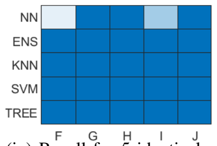
	F	G	H	I	J
NN	1.0	0.842	1.0	0.914	1.0
ENS	1.0	1.0	1.0	1.0	1.0
KNN	1.0	0.864	1.0	0.914	1.0
SVM	1.0	1.0	1.0	1.0	0.969
TREE	0.941	0.430	1.0	0.964	0.576

(ii) Recall for 5 identical accelerometers for 20% training



	A	B	C	D	E
NN	1.0	1.0	1.0	1.0	1.0
ENS	1.0	1.0	1.0	1.0	1.0
KNN	1.0	1.0	1.0	1.0	1.0
SVM	1.0	1.0	1.0	1.0	1.0
TREE	1.0	1.0	1.0	1.0	1.0

(iii) Recall for 5 different accelerometers for 80% training



	F	G	H	I	J
NN	0.777	1.0	1.0	0.857	1.0
ENS	1.0	1.0	1.0	1.0	1.0
KNN	1.0	1.0	1.0	1.0	1.0
SVM	1.0	1.0	1.0	1.0	1.0
TREE	1.0	1.0	1.0	1.0	1.0

(iv) Recall for 5 identical accelerometers for 80% training

Fig. 7. Recall as heatmap (left) and numerical values (right)

5) TREE (Decision Tree) has the maximum number of splits set to 100 and uses the Gini's diversity index split criterion.

C. Results

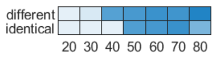
In Figure 3, we show the mean testing accuracy for each classifier and the percentage of training data that was used. In Figure 3 (i), we show the accuracy for different smartphones, while in Figure 3 (ii), we display it for identical smartphones. Overall, the results are favorable for each training percentage. In case of some classifiers, as expected, the accuracy is higher for different smartphones than for identical ones. However, we achieve 100% accuracy with the ENS classifier for all training percentages in both scenarios, i.e., identical and different smartphones. The SVM classifier likewise produces good results, with accuracy ranging from 99% to 100% in both cases. The sole exception is when 20% of the data was used for training, in the case of different smartphones. Accuracy is only 97.5% in this case, however, this is expected given the limited amount of training data. KNN achieves an accuracy of 97.5% to 100% for different smartphones and 90% to 99% for identical smartphones. Similarly, NN yields an accuracy between 97% and 100% for different smartphones and between 91% and 98.3% for identical smartphones. TREE's accuracy ranges from 97.5% to 99% for different smartphones and from 80% to 100% for identical smartphones. With a few outliers, the testing accuracy is more than 97.5%.

In Figures 4, 5, 6, 7, we depict the FAR (False Acceptance Rate), FRR (False Rejection Rate), as well as the precision and recall for each classifier and each smartphone. Since there is no significant variation in results for different training percentages, we chose to show the FAR, FRR, accuracy, and recall only at 20% and 80% training in what follows. These figures show the metrics for different smartphones (i) and identical smartphones (ii) at 20% training. Then, they show the metrics for different smartphones (iii) and identical smartphones (iv) at 80% training. The FAR for all classifiers and all smartphones is zero in the case of 20% training for different smartphones. The FRR is also zero, except for smartphone A, which has a FRR of 3.1% when NN is used. Regarding the precision and recall, the results are 100% except when NN is used. In this case, smartphone A has a precision of 96.8%, and smartphone D has a recall of 96.9%. For identical smartphones, the results for FAR, FRR, accuracy and recall are somewhat worse at 20% training. The exception is ENS, which yields zero FAR and FRR, as well as 100% precision and recall. The FAR for all other classifiers is between 0% and 8.5%, with the exception of smartphone G, when TREE is utilized. In this case, the FRR is 32%. The FAR ranges from 0% to 53.1%, while precision ranges from 53.1% to 100%, and recall ranges from 43% to 100%. As expected, the results are better in the case of 80% training. Except for one phone in case of different device and two phones phones in case of identical devices when the FAR is less than 6.2% and the FRR less than 25% with the NN classifier. For the rest of the classifiers, the FAR and FRR are zero in both situations, i.e., different and identical smartphones. For all classifiers, with different smartphones, both the precision and recall are 100% when 80% was applied. In the case of identical smartphones, devices F and I show the lowest precision and recall, just above 75% with NN. For the rest of of the classifiers, in case of 80% training, the precision and recall are 100% for all devices.

Overall, the ENS classifier performs better than the rest across all training percentages, but the other classifiers also provide good results.

III. Results with deep learning classifiers

This section depicts the architecture of the selected CNN along with the results for each training percentage.



	20%	30%	40%	50%	60%	70%	80%
Different	0.384	0.812	0.925	0.937	0.949	0.943	0.971
Identical	0.297	0.472	0.684	0.93	0.939	0.95	0.893

Fig. 8. Testing accuracy as heatmap (left) and numerical values (right)

A. *MobileNet applied to one-dimensional data*

To further explore the generalization capabilities of our data set, we attempt to train a deep convolutional neural network solely on raw data. For each example in the data set, the previously extracted 7 features are replaced with 1000 acceleration values, i.e., the number of samples in each signal. This simultaneously increases the complexity of our feature space while also removing some of the bias and data loss that occurs during pre-processing. Therefore, as a result of feature space expansion, deep neural networks are used.

The selected CNN, MobileNet-v1 [45], was originally intended as a lightweight, mobile-oriented computer vision architecture. To adapt the network to our specific use case of 1000 features and 1 input channel, we use a TensorFlow[1] implementation[2] that replaces 2D-convolution with its 1D counterpart. The authors of the original paper prioritized efficiency, therefore, they used depthwise separable convolution [46] to reduce the number of floating point operations. This implies that for each block, we use not one but two convolutional layers: a depthwise convolution and a pointwise convolution. The former is applied channel-by-channel, with no further mixing. Therefore, after each depthwise layer, the number of channels remains unchanged. Pointwise convolutions always use a kernel of size 1 and are intended to merge the channels from the preceding layer.

Concretely, the CNN consists of 27 convolutional layers and 1 fully-connected layer placed on top of one another. This corresponds to 1 standard-convolution stem block, followed by 13 depthwise-separable convolution blocks and 1 final dense layer. Each convolution is followed by batch normalization [47] and a ReLU activation function. The stem block lowers the number of features by half while increasing the number of channels from 1 to 64. As data travels down the network, the number of features is lowered by half until it hits 32, and the number of channels eventually reaches 2048. The network has a total of 7,980,613 trainable parameters.

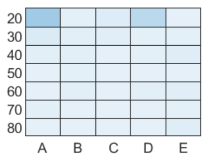
To train the network, we used the multi-class cross entropy loss and the Adam optimizer with a learning rate of 0.001. Training lasted 100 epochs, with the same seven training-testing ratios from 20% to 80% used in section II.

B. *Results*

In Figure 8 we show the accuracy for different and identical smartphones, for all training percentages. As expected, in general, the accuracy is increasing with training percentages, with few exceptions. Also, the accuracy for identical smartphones is lower than for different smartphones. In Figure 9 (i) we depict the FAR for different smartphones and in Figure 9 (ii)

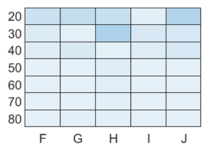
[1]https://www.tensorflow.org/

[2]https://github.com/Sakib1263/MobileNet-1D-2D-Tensorflow-Keras



	A	B	C	D	E
20%	0.300	0.006	0.036	0.190	0.001
30%	0.046	0.012	0.019	0.006	0.006
40%	0.013	0.008	0.002	0.006	0.026
50%	0	0	0	0.002	0.002
60%	0	0.009	0.001	0.010	0.017
70%	0.008	0.001	0	0.001	0.007
8%	0	0	0	0.001	0.029

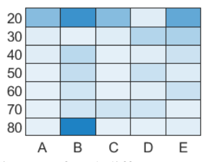
(i) FAR for 5 different accelerometers



	F	G	H	I	J
20%	0.110	0.152	0.131	0.010	0.226
30%	0.044	0.005	0.244	0.055	0.069
40%	0.027	0.048	0.006	0.032	0.069
50%	0	0	0.029	0.001	0.003
60%	0	0	0.021	0.003	0
70%	0	0.001	0.011	0.001	0
80%	0	0.0091	0.011	0.001	0

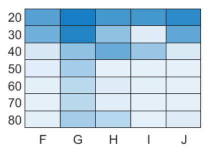
(ii) FAR for 5 identical accelerometers

Fig. 9. FAR for different and identical smartphones as heatmap (left) and numerical values (right)



	A	B	C	D	E
20%	0.413	0.738	0.416	0.026	0.573
30%	0.027	0	0.029	0.219	0.252
40%	0.025	0.188	0.006	0.026	0.111
50%	0.001	0.149	0	0.123	0.037
60%	0	0.093	0.001	0.025	0.118
70%	0	0.088	0.098	0.093	0.008
80%	0.002	0.867	0	0.007	0.020

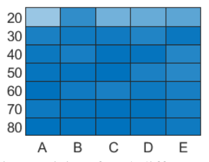
(i) FRR for 5 different accelerometers



	F	G	H	I	J
20%	0.623	0.900	0.700	0.710	0.800
30%	0.5092	0.850	0.378	0.037	0.580
40%	0.064	0.378	0.543	0.334	0.050
50%	0.018	0.282	0.003	0.001	0
60%	0.006	0.188	0.035	0.005	0
70%	0	0.194	0.024	0	0
80%	0.010	0.292	0.200	0	0.035

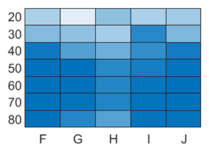
(ii) FRR for 5 identical accelerometers

Fig. 10. FRR for different and identical smartphones as heatmap (left) and numerical values (right)



	A	B	C	D	E
20%	0.334	0.784	0.500	0.548	0.596
30%	0.884	0.931	0.928	0.855	0.953
40%	0.946	0.896	0.981	0.974	0.837
50%	0.970	0.977	0.995	0.882	0.824
60%	0.999	0.928	0.887	0.940	0.896
70%	0.940	0.980	0.990	0.988	0.907
80%	0.996	0.988	0.993	0.968	0.931

(i) Precision for 5 different accelerometers

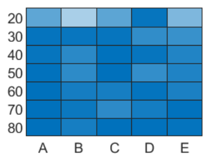


	F	G	H	I	J
20%	0.253	0.020	0.382	0.254	0.299
30%	0.427	0.373	0.283	0.825	0.448
40%	0.934	0.634	0.517	0.776	0.920
50%	0.998	1.0	0.816	0.900	0.986
60%	0.999	1.0	0.834	0.966	1.0
70%	0.998	0.984	0.855	0.983	1.0
80%	0.994	0.831	0.624	0.977	0.987

(ii) Precision for 5 identical accelerometers

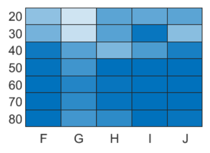
Fig. 11. Precision for different and identical smartphones as heatmap (left) and numerical values (right)

we depict the FAR for identical smartphones for all training percentages. For different smartphones, the values are between 0% and 30.09%. In case of identical smartphones the minimum values are also 0%, but the maximum values are 24.4%. In Figure 10 we depict the FRR for different (i) and identical (ii) smartphones. For different smartphones, the FRR is between 0% and 86.7%, while for identical smartphones, the values are similar, between 0% and 90%. In terms of precision, for different smartphones, in Figure 11 (i) the values are



	A	B	C	D	E
20%	0.586	0.264	0.594	0.972	0.450
30%	0.974	0.958	0.970	0.781	0.750
40%	0.972	0.813	0.993	0.972	0.852
50%	0.990	0.800	1.0	0.777	0.863
60%	1.0	0.903	0.900	0.977	0.880
70%	1.0	0.947	0.802	0.908	0.990
80%	0.997	0.900	1.0	0.992	0.977

(i) Recall for 5 different accelerometers



	F	G	H	I	J
20%	0.376	0.1	0.600	0.578	0.600
30%	0.490	0.140	0.621	0.964	0.400
40%	0.934	0.622	0.456	0.665	0.894
50%	0.982	0.718	1.0	0.997	1.0
60%	0.994	0.847	0.965	0.996	1.0
70%	1.0	0.798	0.976	1.0	1.0
80%	0.990	0.708	0.8	1.0	0.965

(ii) Recall for 5 identical accelerometers

Fig. 12. Recall for different and identical smartphones as heatmap (left) and numerical values (right)

between 33.4% and 99.9%, while for identical smartphones, in Figure 11 (ii), the values are between 2% and 99.9%. The recall for different smartphones, as shown in Figure 12 (i), is between 26.4% and 100%. In case of identical smartphones, as shown in Figure 12 (ii), the minimum values is 10% and the maximum values is 100%.

IV. Conclusion

In this work, we discuss smartphone fingerprinting based on their accelerometer sensors, using several machine learning classifiers, i.e., NN, Ensemble, KNN, SVM, and Decision Trees on 7 time domain features, i.e., Kurtosis, Skewness, SNR, STD, RMS, peak value, and SINAD, or, alternatively, just the raw accelerometer data in case of a CNN. Our data set contained examples from five identical and different smartphones and reached a maximum accuracy of 100% for identification, when the Ensemble classifier was used. The CNN performed much worse when low percentages of training data were used and the results improved significantly with more training data, still not necessarily outperforming the traditional classifiers. Consequently, the results show that traditional machine learning approaches applied on statistical properties of accelerometer data can give better results than some more complex deep learning architectures in case of fingerprinting smartphones based on accelerometer sensors, especially when limited training data is available.

Acknowledgement. This paper was financially supported by the Project "Network of excellence in applied research and innovation for doctoral and postdoctoral programs / InoHubDoc", project co-funded by the European Social Fund financing agreement no. POCU/993/6/13/153437.

References

[1] A. Berdich, B. Groza, E. Levy, A. Shabtai, Y. Elovici, and R. Mayrhofer, "Fingerprinting smartphones based on microphone characteristics from environment affected recordings," *IEEE Access*, vol. 10, 2022.

[2] Y. Lee, J. Li, and Y. Kim, "Micprint: acoustic sensor fingerprinting for spoof-resistant mobile device authentication," in *Proceedings of the 16th EAI International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services*, 2019, pp. 248–257.

[3] T. Qin, R. Wang, D. Yan, and L. Lin, “Source cell-phone identification in the presence of additive noise from cqt domain,” *Information*, vol. 9, no. 8, p. 205, 2018.

[4] Z. Zhou, W. Diao, X. Liu, and K. Zhang, “Acoustic fingerprinting revisited: Generate stable device id stealthily with inaudible sound,” in *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, 2014, pp. 429–440.

[5] A. Berdich, B. Groza, R. Mayrhofer, E. Levy, A. Shabtai, and Y. Elovici, “Sweep-to-unlock: Fingerprinting smartphones based on loudspeaker roll-off characteristics,” *IEEE Transactions on Mobile Computing*, 2021.

[6] J. Tian, J. Zhang, X. Li, C. Zhou, R. Wu, Y. Wang, and S. Huang, “Mobile device fingerprint identification using gyroscope resonance,” *IEEE Access*, 2021.

[7] D. Bichler, G. Stromberg, M. Huemer, and M. Löw, “Key generation based on acceleration data of shaking processes,” in *UbiComp 2007: Ubiquitous Computing: 9th International Conference, UbiComp 2007, Innsbruck, Austria, September 16-19, 2007. Proceedings 9*. Springer, 2007, pp. 304–317.

[8] R. Mayrhofer and H. Gellersen, “Shake well before use: two implementations for implicit context authentication,” *Adjunct Proc. Ubicomp 2007*, 2007.

[9] S. Dey, N. Roy, W. Xu, R. R. Choudhury, and S. Nelakuditi, “Accelprint: Imperfections of accelerometers make smartphones trackable.” in *NDSS*. Citeseer, 2014.

[10] T. Van Goethem, W. Scheepers, D. Preuveneers, and W. Joosen, “Accelerometer-based device fingerprinting for multi-factor mobile authentication,” in *International Symposium on Engineering Secure Software and Systems*. Springer, 2016, pp. 106–121.

[11] Z. Ding and M. Ming, “Accelerometer-based mobile device identification system for the realistic environment,” *IEEE Access*, vol. 7, pp. 131 435–131 447, 2019.

[12] H. Bojinov, Y. Michalevsky, G. Nakibly, and D. Boneh, “Mobile device identification via sensor fingerprinting,” *arXiv preprint:1408.1416*, 2014.

[13] I. Amerini, P. Bestagini, L. Bondi, R. Caldelli, M. Casini, and S. Tubaro, “Robust smartphone fingerprint by mixing device sensors features for mobile strong authentication,” *Electronic Imaging*, vol. 2016, no. 8, pp. 1–8, 2016.

[14] I. Amerini, R. Becarelli, R. Caldelli, A. Melani, and M. Niccolai, “Smartphone fingerprinting combining features of on-board sensors,” *IEEE Transactions on Information Forensics and Security*, vol. 12, no. 10, pp. 2457–2466, 2017.

[15] A. Das, N. Borisov, and M. Caesar, “Exploring ways to mitigate sensor-based smartphone fingerprinting,” *arXiv preprint: 1503.01874*, 2015.

[16] ——, “Tracking mobile web users through motion sensors: Attacks and defenses.” in *NDSS*, 2016.

[17] A. Das, N. Borisov, and E. Chou, “Every move you make: Exploring practical issues in smartphone motion sensor fingerprinting and countermeasures.” *Proc. Priv. Enh. Tech.*, vol. 2018, no. 1, pp. 88–108, 2018.

[18] X.-Y. Li, H. Liu, L. Zhang, Z. Wu, Y. Xie, G. Chen, C. Wan, and Z. Liang, “Finding the stars in the fireworks: Deep understanding of motion sensor fingerprint,” *IEEE/ACM Transactions on Networking*, vol. 27, no. 5, pp. 1945–1958, 2019.

[19] K. Ren, Z. Qin, and Z. Ba, “Toward hardware-rooted smartphone authentication,” *IEEE Wireless Com.*, vol. 26, no. 1, pp. 114–119, 2019.

[20] T. Hupperich, H. Hosseini, and T. Holz, “Leveraging sensor fingerprinting for mobile device authentication,” in *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*. Springer, 2016, pp. 377–396.

[21] J. Zhang, A. R. Beresford, and I. Sheret, “Factory calibration fingerprinting of sensors,” *IEEE Transactions on Information Forensics and Security*, vol. 16, pp. 1626–1639, 2020.

[22] ——, “Sensorid: Sensor calibration fingerprinting for smartphones,” in *2019 IEEE Symposium on Sec. and Priv.* IEEE, 2019, pp. 638–655.

[23] N. Kalra and D. Bansal, “Analyzing driver behavior using smartphone sensors: a survey,” *Int. J. Electron. Electr. Eng*, vol. 7, no. 7, pp. 697–702, 2014.

[24] P. Singh, N. Juneja, and S. Kapoor, “Using mobile phone sensors to detect driving behavior,” in *Proceedings of the 3rd ACM Symposium on Computing for Development*. ACM, 2013, p. 53.

[25] J. Yu, Z. Chen, Y. Zhu, Y. J. Chen, L. Kong, and M. Li, “Fine-grained abnormal driving behaviors detection and identification with smartphones,” *IEEE transactions on mobile computing*, vol. 16, no. 8, pp. 2198–2212, 2017.

[26] M. Enev, A. Takakuwa, K. Koscher, and T. Kohno, “Automobile driver fingerprinting,” *Proceedings on Privacy Enhancing Technologies*, vol. 2016, no. 1, pp. 34–50, 2016.

[27] P. Phumphuang, P. Wuttidittachotti, and C. Saiprasert, “Driver identification using variance of the acceleration data,” in *Computer Science and Engineering Conference, 2015 International*. IEEE, 2015, pp. 1–6.

[28] M. Van Ly, S. Martin, and M. M. Trivedi, “Driver classification and driving style recognition using inertial sensors,” in *Intelligent Vehicles Symposium (IV), 2013 IEEE*. IEEE, 2013, pp. 1040–1045.

[29] D. A. Johnson and M. M. Trivedi, “Driving style recognition using a smartphone as a sensor platform,” in *2011 14th International IEEE Conf. on Intl. Transp. Sys.* IEEE, 2011, pp. 1609–1615.

[30] T. Feng and H. J. Timmermans, “Transportation mode recognition using GPS and accelerometer data,” *Transportation Research Part C: Emerging Technologies*, vol. 37, pp. 118–130, 2013.

[31] S. Wang, C. Chen, and J. Ma, “Accelerometer based transportation mode recognition on mobile phones,” in *Wearable Computing Systems (APWCS), 2010 Asia-Pacific Conference on*. IEEE, 2010, pp. 44–46.

[32] S. Reddy, M. Mun, J. Burke, D. Estrin, M. Hansen, and M. Srivastava, “Using mobile phones to determine transportation modes,” *ACM Transactions on Sensor Networks (TOSN)*, vol. 6, no. 2, p. 13, 2010.

[33] A. Mednis, G. Strazdins, R. Zviedris, G. Kanonirs, and L. Selavo, “Real time pothole detection using android smartphones with accelerometers,” in *Distributed Computing in Sensor Systems and Workshops (DCOSS), 2011 International Conference on*. IEEE, 2011, pp. 1–6.

[34] K. Chen, M. Lu, X. Fan, M. Wei, and J. Wu, “Road condition monitoring using on-board three-axis accelerometer and GPS sensor,” 2011.

[35] Z. He, J. Cao, X. Liu, and S. Tang, “Who sits where? Infrastructure-free in-vehicle cooperative positioning via smartphones,” *Sensors*, vol. 14, no. 7, pp. 11 605–11 628, 2014.

[36] M. Muaaz and R. Mayrhofer, “Accelerometer based gait recognition using adapted gaussian mixture models,” in *Proceedings of the 14th International Conference on Advances in Mobile Computing and Multi Media*, ser. MoMM ’16. New York, NY, USA: Association for Computing Machinery, 2016, p. 288–291.

[37] A. Mongia, V. M. Gunturi, and V. Naik, “Detecting activities at metro stations using smartphone sensors,” in *2018 10th Intl. Conf. on Communication Systems & Networks*. IEEE, 2018, pp. 57–65.

[38] M. Muaaz and R. Mayrhofer, “Accelerometer based gait recognition using adapted gaussian mixture models,” in *Proceedings of the 14th International Conference on Advances in Mobile Computing and Multi Media*, 2016, pp. 288–291.

[39] X. Su, H. Tong, and P. Ji, “Activity recognition with smartphone sensors,” *Tsinghua science and tech.*, vol. 19, no. 3, pp. 235–249, 2014.

[40] J. Hua, Z. Shen, and S. Zhong, “We can track you if you take the metro: Tracking metro riders using accelerometers on smartphones,” *IEEE Transactions on Information Forensics and Security*, vol. 12, no. 2, pp. 286–297, 2016.

[41] B. Groza, A. Berdich, C. Jichici, and R. Mayrhofer, “Secure accelerometer-based pairing of mobile devices in multi-modal transport,” *IEEE Access*, vol. 8, pp. 9246–9259, 2020.

[42] B. Groza and R. Mayrhofer, “Saphe: simple accelerometer based wireless pairing with heuristic trees,” in *Proceedings of the 10th Intl. Conf. on Advances in Mobile Computing & Multimedia*, 2012, pp. 161–168.

[43] M. Fomichev, J. Hesse, L. Almon, T. Lippert, J. Han, and M. Hollick, “Fastzip: Faster and more secure zero-interaction pairing,” in *Proceedings of the 19th Annual International Conference on Mobile Systems, Applications, and Services*, 2021, pp. 440–452.

[44] M. Fomichev, F. Álvarez, D. Steinmetzer, P. Gardner-Stephen, and M. Hollick, “Survey and systematization of secure device pairing,” *IEEE Communications Surveys & Tutorials*, vol. 20, no. 1, pp. 517–550, 2018.

[45] A. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, “Mobilenets: Efficient convolutional neural networks for mobile vision applications,” 04 2017.

[46] L. Sifre and S. Mallat, “Rigid-motion scattering for texture classification,” 2014. [Online]. Available: https://arxiv.org/abs/1403.1687

[47] S. Ioffe and C. Szegedy, “Batch normalization: Accelerating deep network training by reducing internal covariate shift,” 02 2015.